Models Cascade for Tree-Structured Named Entity Detection

Marco Dinarelli
LIMSI-CNRS / Orsay Cedex, France
marcod@limsi.fr

Sophie Rosset
LIMSI-CNRS / Orsay Cedex, France
rosset@limsi.fr

Abstract

Named Entity Recognition (NER) is a well-known Natural Language Processing (NLP) task, used as a preliminary processing to provide a semantic level to more complex tasks. In this paper we describe a new set of named entities having a multi-level tree structure, where base entities are combined to define more complex ones. This definition makes the NER task more complex than previous tasks, even more due to the use of noisy data for the annotation: transcriptions of French broadcast data. We propose an original and effective system to tackle this new task, putting together the strengths of solutions for sequence labeling approaches and syntactic parsing via cascading of different models. Our system was evaluated in the 2011 Quaero named entity detection evaluation campaign and ranked first, with results far better than those of the other participating systems.

1 Introduction

Named Entity Detection is a well-known NLP task used to extract semantic information in other more complex tasks such as Relation Extraction (Doddington et al., 2004) or Question Answering (Voorhees, 2001). After its definition in MUC-6 (Grishman and Sundheim, 1996), the NER task evolved increasing its complexity. Current definitions provide a fine-grained semantic information level with a broad coverage (Sekine, 2004). This has increased the interests in developing named entity detection systems.

In this paper we describe a new set of named entities defined recently(Grouin et al., 2011). These named entities have a multilevel tree structure where components are combined to define more complex and general entity structures. This definition increases significantly the complexity of the NER task, even more due to the type of data used for the annotation: manual and automatic transcriptions of French broadcast data.

Given such a definition, it is not possible to tackle the task with traditional sequence labeling approaches. At the same time, solutions able to reconstruct tree structures from flat sequences, like syntactic parsing solutions, may have serious limitations, due to the noisy data used. In order to solve these problems, we studied and implemented a two-stage system that put together the strengths of the two approaches: i) a complex linear-chain Conditional Random Field (CRF) model, integrating a huge number of features, takes the noisy data as input and generates the corresponding sequence of components; ii) a syntactic parsing model based on Probabilistic Context Free Grammar (PCFG) reconstructs the tree-structured named entities.

We describe our solution showing that it is equivalent, in the processing steps, to a syntactic parsing analysis solution, and that it can better handle noisy data. Additionally, given the amount of data used and the relatively slow training time of CRF models when using large amount of data and features, we describe a procedure for incremental CRF model training that solves this particular efficiency problem. The system we propose ranked first at the 2011 Quaero NER evaluation campaign (Galibert et al., 2011), with results far better than those of the other participating systems, though results on automatic transcriptions are largely affected by ASR system errors.

The remainder of the paper is organized as follows: in the next section we describe the new set of named entities defined and the data used for annotation. In section 3 we describe the system we propose, and implemented for the 2011 Quaero Named Entity Recognition evaluation campaign. In section 4 we detail the experimental setup, and

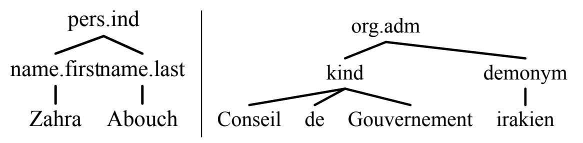

Figure 1: Examples of structured named entities defined within the Quaero project. Left: Iraqi Governing Council

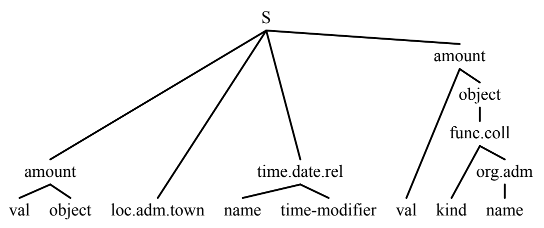

Figure 2: An example of named entity tree corresponding to entities of a whole sentence. Tree leaves, corresponding to sentence words have been removed to keep readability.

describe and comment the results. Finally in section 5 we draw our conclusions and propose some perspectives for future work.

2 Towards Tree Structured Named Entities

Named Entity Recognition was first defined as recognizing proper names (Coates-Stephens, 1992). Since MUC-6 (Grishman and Sundheim, 1996) named entities are proper names falling into three major classes: persons, locations and organizations. There are some propositions to sub-divide these entities into fine-grained classes. For example, politicians for the person class (Fleischman and Hovy, 2002) or cities for the location class (Fleischman, 2001). Some are sometimes added like product (Bick, 2004; Galliano et al., 2009).

Recently some extensions of named entity have been proposed. For example, (Sekine, 2004) defined a complete hierarchy of named entities containing about 200 types.

A well-known task of named entity detection is the one proposed for the CoNLL shared task 2003, described in (Tjong Kim Sang and De Meulder, 2003), where only four named entities were involved: *Person*, *Organization*, *Location* and *Other*. The latter was used for proper names belonging to any other entity different from the first three types.

After this task, named entity detection has been refined to include other entities, e.g. describing time expressions, events, quantity, currency etc. Current named entity detection tasks provide a fine-grained semantic representation level of the lexical surface form, sometimes comparable to other semantic representations like those used in Semantic Role Labeling (Carreras and Marquez, 2005) or Spoken Language Understanding (De Mori et al., 2008). The increasing complexity of named entities definition reflects the change of needs in computer science activities, that in turn is the consequence of the tremendous growth of the amount of information to deal with nowadays.

The set of named entities used in this work has been recently defined in (Grouin et al., 2011) and presents an important difference with respect to previous sets. Beyond the presence of subtypes, named entities have a tree structure. For example, the *Organization* type can be characterized by the subtypes *administrative* or *enterprise* giving the named entities *org.adm* and *org.ent* starting from the entity *org*. Each entity is composed of at least one component, which trigger in any case an entity type. The component allows to characterize more precisely the semantic content of the entity. For example the entity *pers*, used to describe persons, has, among others, the components *name.first*, for first name, and *name.last*, for last name. The presence of either *name.first* or *name.last* (or both) triggers the entity *pers*. The results of the composition is a tree-structured named entity. Two examples of structured named entities are shown in figure 1. In addition we report a named entity tree in figure 2, where words, corresponding to tree leaves, have been removed to keep readability, and it was generated starting from the sentence:

> **90 personnes** toujours présentes **à Atambua** c'est là qu'**hier matin** ont été tués **3 employés du haut commissariat des Nations unies** aux réfugiés, le **HCR**[1]

Here words realizing entities have been highlighted in bold. More details on the definition of this new set of named entities can be found in (Grouin et al., 2011), where also statistics on inter-annotators agreements are reported.
Given this structured named entities definition, the corresponding NER task is significantly more complex than previous equivalent tasks, where the entity structure was flat. Moreover, the complexity of the task is increased also by the type of data used for the annotation, that is manual and automatic transcriptions of French broadcast data. In

[1]90 people still present in Atambua is where yesterday morning killed three employees of the United Nations High Commissioner for Refugees UNHCR.

Quaero	training		dev	
# sentences	43,251		112	
	words	entities	words	entities
# tokens	1,251,432	245,880	2,659	570
# vocabulary	39,631	134	891	30
# components	–	133662	–	971
# components dict.	–	28	–	18
# OOV rate [%]	–	–	17.15	0

Table 1: Statistics on the training and development sets of the Quaero corpus.

Quaero	test BN		test BC	
# sentences	1704		3933	
	words	entities	words	entities
# tokens	32945	2762	69414	2769
# vocabulary		28		28
# components	–	4128	–	4017
# components dict.	–	21	–	20
# OOV rate [%]	3.63	0	3.84	0

Table 2: Statistics on the test set of the Quaero corpus, divided in Broadcast News (BN) and Broadcast Conversations (BC)

particular the data have been collected not only from French radio channels, but also from a North-African French-speaking radio channel. This aspect introduces complexity due to different expressions used from French non-native speakers as well as to their particular accents and sometimes vocabulary.

The training data are the same used for the ESTER2 evaluation campaign (Galliano et al., 2009). The transcriptions have been re-annotated with the named entities described above.

The corpus will be referred in this work as `Quaero`. Description of training, development and test data are reported in table 1 and 2. In particular the test set is constituted by different kind of data: transcriptions of broadcast news, and broadcast conversations. the merge of the first two types. An interesting point in table 1 and 2 is the Out-of-Vocabulary (OOV) rate of dev and test sets.

3 Tree-Structured Named Entity Detection System

Given the definition of named entities described in previous section, we consider that, even if possible, the best solution cannot be a sequence labeling approach as largely used in standard named entity detection task (Tjong Kim Sang and De Meulder, 2003). In contrast, an approach coming from syntactic parsing would be possible. In that case syntactic parse trees could be replaced by named entity trees like the one shown in figure 2[2].

A first problem with this choice is constituted by the noisy data involved in our task. Indeed, our preliminary evaluations of a classic algorithm used successfully for syntactic parsing, resulted in quite poor results. Note also that most syntactic parsing solutions, are designed to perform parsing using Part-of-Speech (POS) tags, annotated upon words, as tree leaves, instead of words. This solution introduces a good generalization over surface forms, allowing algorithms to deal with relatively noisy data, although the same solutions proved to be much less effective on automatic transcriptions. Moreover, this solution is effective for syntactic parsing, since syntactic constituents are directly related to POS tags, but it would not be reasonable for named entity detection, since the generalization introduced by POS tags over lexical surface forms would make it impossible to discriminate between different entities. Indeed, most of the named entities syntactic heads are known to be nouns. Additionally, since our named entity detection task must be performed also on automatic transcriptions containing ASR mistakes, a solution designed for syntactic parsing and adapted to be applied directly on lexical surface forms would be not robust enough, due to OOV words and ASR mistakes that alter significantly the syntactic structure of the input sentence. Our idea is thus to split the annotation process in two phases:

1. In the first step we use a model robust to noisy input and Out-of-Vocabulary words in order to annotate entity components, i.e. components of named entities that can be annotated directly on words (tree leaves in figure 2).
2. In the second step we use a model for syntactic parsing in order to reconstruct the entire entity trees.

The first step is a sequence segmentation and labeling task, thus any model suitable for this kind of problems can be adopted. Given its characteristics and its success in sequence labeling tasks, we adopt CRF (Lafferty et al., 2001) for the first step. For the second step we adopt a Probabilistic Context-Free Grammar (Booth and Thomson, 1973; Krenn and Samuelsson, 1997). The next two subsections describe these two models.

3.1 Linear-Chain Conditional Random Fields

CRFs have been proposed for the first time for sequence segmentation and labeling tasks in (Lafferty et al., 2001). This model belongs to the family of exponential or log-linear models. Its main characteristics are the possibility to include a huge number of features, like the Maximum Entropy

[2] where words are not shown for readability reason

model, but computing global conditional probabilities normalized at sentence level, instead of position level. In particular this last point results very effective since it solves the label bias problem, as pointed out in (Lafferty et al., 2001).

Given a sequence of N words $W_1^N = w_1, ..., w_N$ and its corresponding sequence of named entities $E_1^N = e_1, ..., e_N$, CRF trains the conditional probabilities

$$P(E_1^N|W_1^N) = \frac{1}{Z}\prod_{n=1}^{N}\exp\left(\sum_{m=1}^{M}\lambda_m \cdot h_m(e_{n-1}, e_n, w_{n-2}^{n+2})\right) \quad (1)$$

where λ_m are the training parameters. $h_m(e_{n-1}, e_n, w_{n-2}^{n+2})$ are the feature functions capturing conditional dependencies of entities and words. Z is a probability normalization factor in order to model well defined probability distribution:

$$Z = \sum_{\tilde{e}_1^N}\prod_{n=1}^{N} H(\tilde{e}_{n-1}, \tilde{e}_n, w_{n-2}^{n+2}) \quad (2)$$

where $\tilde{e}_{n-1}$ and $\tilde{e}_n$ are the entities hypothesized for the previous and current words, $H(\tilde{e}_{n-1}, \tilde{e}_n, w_{n-2}^{n+2})$ is an abbreviation for $\sum_{m=1}^{M} \lambda_m \cdot h_m(e_{n-1}, e_n, w_{n-2}^{n+2})$.

Two particular effective implementations of CRFs have been recently proposed. One is described in (Hahn et al., 2009) and uses a margin based training criteria for probabilities estimation. The other is described in (Lavergne et al., 2010) and has been implemented in the software *wapiti*[3]. The latter solution in particular trains the model using two different regularization parameters at the same time: Gaussian prior, also known as *l2* regularization and used in many software to avoid over fitting; and Laplacian prior, also known as *l1* regularization (Riezler and Vasserman, 2010), which has the effect to filter out features with very low scores. These two regularization parameters are used together in the model implementing the so-called *elastic net* regularization (Zou and Hastie, 2005):

$$l(\lambda) + \rho_1\|\lambda\|_1 + \frac{\rho_2}{2}\|\lambda\|_2^2 \quad (3)$$

λ is the set of parameters of the model introduced in equation 1, $l(\lambda)$ is the minus-logarithm of equation 1, used as loss function for training CRF. $\|\lambda\|_1$ and $\|\lambda\|_2$ are the *l1* and *l2* regularization, respectively, while ρ_1 and ρ_2 are two parameters that can be optimized as usual on development data or with cross validation.

[3]available at `http://wapiti.limsi.fr`

As explained in (Lavergne et al., 2010), using *l1* regularization is an effective way for feature selection in CRF at training time. Note that other approaches have been proposed for feature selection, e.g. in (McCallum, 2003). In this work we refer to the CRF implementation described in (Lavergne et al., 2010).

3.1.1 Incremental CRF Training

Despite the improvements on CRF implementations, this model remains hard to train on large amount of data when using a reasonable amount of features. Using the data described in section 2 and using features as word prefixes and suffixes, capitalization, punctuation and morpho-syntactic features, our CRF model creates more than 2 billions feature functions. Training such a model is infeasible on current machines.

Exploiting the characteristics of the CRF software *wapiti* and the definition of feature functions, we implemented a procedure for incremental training of CRF models that can be used with an arbitrary number of features. Feature functions $h_m(e_{n-1}, e_n, w_{n-2}^{n+2})$ used in our CRF models have the form:

$$h_{w,e}(e_i, w_i) = \delta(w_i^{'}, w_i) \cdot \delta(e_i^{'}, e_i) \quad (4)$$

where $\delta(.,.)$ is the Kronecker function. This particular function fires when the current word $w_i^{'}$ in the sequence matches w_i and the corresponding entity $e_i^{'}$ matches e_i. When using such simple features, there is usually no limitation on the amount of data that can be used for training. Indeed, more accurate models can be trained using complex features, using also words and entities at previous or next positions, i.e. adjacent tokens. For example:

$$\begin{aligned} h_{w,e}(e_i, w_i) &= \delta(w_{i-1}^{'}, w_{i-1}) \cdot \delta(w_i^{'}, w_i) \\ &\cdot\delta(e_{i-1}^{'}, e_{i-1}) \cdot \delta(e_i^{'}, e_i) \end{aligned} \quad (5)$$

This feature function fires if both words and entities at current and previous positions match. The higher accuracy reached with this type of features is paid at training time with the number of feature function generated in situations like the one reported above, and makes direct CRF model training unfeasible. The solution to this limitation is given observing that:

1. The software we use for CRF model training performs an effective feature selection thanks to *l1* regularization and can reload models for further refinements.

2. Feature functions like 5 fire if and only if both words and both entities at previous and current position matches.

The second point means that if a simple feature matching only the word (and/or entity) at current position is filtered away from *l1* regularization, it doesn't make sense to include in the training any complex feature function involving that word (and/or entity). In order to train our model using all the features, we proceed in three different steps:

1. we train a model with simple feature functions like 4
2. we search the model for simple feature functions corresponding to adjacent words (and/or entities), that in turn corresponds to complex features
3. we retrain the model where we added the complex features found at previous step

In the second step, the features kept in the model are some order of magnitude fewer in number than those generated including directly complex features in step 1, like it can be done with less data. As a consequence, the number of complex features is limited. Moreover, the fact that simple feature functions must correspond to adjacent words (and/or entities) is a further constraint for the number of added features. As we will see in section 4.1, training CRF model with this procedure provides the same prediction accuracy than what we could have training the model directly with all features.

3.2 Structured Named Entities Reconstruction

Models described in this section are well-known solutions for syntactic parsing. We report them to provide a self-contained and complet work, our main contribution in this respect is to have implemented and adapted algorithms to our task.

The model we use for entity tree reconstruction is PCFG (Booth and Thomson, 1973; Krenn and Samuelsson, 1997). There are more accurate models for the same purpose, e.g. the one used in (Charniak and Johnson, 2005). In practice, the first annotation step being carried out with CRF, which provide a high robustness on noisy data, there is no need for complex and expensive models. Moreover PCFG are quite accurate and very fast for parsing (Collins and Koo, 2005).

The input of the CRF model described in the previous section is a sentence like the one reported in section 2[4]. The output is the sequence of entities corresponding to the leaves of the tree in Figure 2, i.e. entity tree components. For example the chunk **Nations unies** (*United Nations*) is annotated by CRF as

org.adm-B{**Nations**} *org.adm-I*{**unies**}

where suffixes -B and -I (for Begin and Inside) are used to have a one-to-one correspondence between words and entities. This makes the NER task a sequence segmentation and labeling problem, without having to deal with alignment issues. From the annotation above it is immediate to reconstruct the annotation

org.adm{**Nations unies**}

Afterwards words are removed and only components are used as input of the PCFG model to reconstruct the entity tree. In our example, the input would then be:

val object name time-modifier val kind name.

PCFG production rules are extracted directly from the trees. For example from the tree in Figure 2 the following set of rules is extracted:

S ⇒ *amount loc.adm.town ... org.adm*
amount ⇒ *val object*
time.date.rel ⇒ *name time-modifier*
object ⇒ *func.coll*
func.coll ⇒ *kind org.adm*
org.adm ⇒ *name*

where the first production as been cut to keep readability and corresponds to the children of the tree root *S*. Once the rules have been generated from all trees in the training set, probabilities are estimated with simple maximum likelihood estimation as the probability of a production given the right-hand side (RHS) of the rule. The parsing algorithm using the PCFG generated from entity trees is the *Cocke-Younger-Kasami* (CYK) described in (Johnson, 1998). In order to use this algorithm production rules must be in Chomsky Normal Form (CNF), i.e. rules must have one of the two forms: i) $X_i \Rightarrow X_j X_k$; ii) $X_i \Rightarrow w$, where X are non-terminal symbols and w are terminal symbols. The corresponding probabilities are

$$p_{i \to j,k} = \frac{P(X_i \Rightarrow X_j, X_k)}{P(X_i)} \tag{6}$$

$$p_{i \to w} = \frac{P(X_i \Rightarrow w)}{P(X_i)} \tag{7}$$

[4]***90 personnes*** *toujours présentes* ***à Atambua*** *c' est là qu'* ***hier matin*** *ont été tués* ***3 employés du haut commissariat des Nations unies*** *aux réfugiés , le* ***HCR***

There are well-known algorithms to convert a grammar into CNF, e.g. (Krenn and Samuelsson, 1997). Probabilities are then re-estimated using the Expectation Maximization algorithm called *Inside-Outside* (Krenn and Samuelsson, 1997). The latter is equivalent to the *forward-backword* algorithm for HMM (Rabiner, 1989), where *Inside* and *Outside* variables are used instead of *Forward* and *Backward* variables.

Inside variables $I_i^w(s,t)$ store the probability $P(X_i \Rightarrow^* \mathbf{w}_{s,t} \mid X_i)$, that is the probability of producing the sub-sequence $\mathbf{w}_{s,t} = w_s, ..., w_t$ of the string $\mathbf{w}_{1,T} = w_1, ..., w_T$ from the non-terminal symbol X_i, given the non-terminal symbol X_i, in any number of steps ($\Rightarrow^*$ is the closure of the production symbol $\Rightarrow$). *Outside* variables $O_i^w(s,t)$ store the probability $P(S \Rightarrow^* \mathbf{w}_{0,s}, X_i, \mathbf{w}_{t,T} \mid S)$, that is the probability of producing the subsequence $w_0, ..., w_s, X_i, w_t, ..., w_T$ from the root symbol S, given the root symbol S.

Probability re-estimation consists in computing the quantities $P(X_i \Rightarrow X_j, X_k)$, $P(X_i \Rightarrow w)$ and $P(X_i)$ in terms of *Inside* and *Outside* variables. This give a new estimation of rules probabilities that is used to re-compute *Inside* and *Outside* variables. This procedure is repeated until a convergence criterion is met, e.g. the likelihood doesn't increase significantly.

Given the two annotation steps implemented with CRF and PCFG, our system is in principle equivalent to a single system for syntactic parsing performing a "one-shot" annotation. In particular, solutions like (Charniak and Johnson, 2005) and (Collins and Koo, 2005) use POS tags as tree leaves. This has a two-fold benefit: i) introduces a generalization level over surface forms; ii) provide to the parsing algorithm only the essential information, since POS tags are directly related to syntactic constituents and are sufficient to induce the syntactic structure of a sentence. In our system, the role played by POS tags in syntactic parsing is played by entity components, annotated by CRF.In contrast, for named entity annotation it is not true that components are sufficient to induce the entity tree. Nevertheless, as we will see in next section, this solution does not prevent having good results.

4 Experiments and Results

In this section we first describe the experimental setup, and then we discuss the results. As mentioned in section 3.1, the software used for CRF models is *wapiti*.[5] The procedure for incremental training of CRF models is realized with our own software. We didn't optimize parameters ρ_1 and ρ_2 of the elastic net (see section 3.1), default values lead in most cases to very accurate models. We used a wide set of features in CRF models, in a window of [-2,+2] around the target word:

- A set of standard features like word prefixes and suffixes of length from 1 to 5, plus some *Yes/No* features like *"Does the word start with capital letter ?" "Does the word contain non alphanumeric characters ?"*, etc.
- Morpho-syntactic features extracted from the output of the tool *tagger* (Allauzen and Bonneau-Maynard, 2008)
- Features extracted from the output of the tool *WMatch* (Galibert, 2009; Rosset et al., 2008).

The output provided by *WMatch* contains detailed motpho-syntactic information as well as semantic information at the same level of named entities. Concerning the PCFG model, for preliminary studies we used our own implementation, but for this work we used the much faster implementation described in (Johnson, 1998).[6]

Concerning data, for preliminary studies carried out to validate our incremental training procedure, we used the same data used in the ESTER2 named entity detection evaluation campaign, Thus our results can be directly compared with those reported in (Galliano et al., 2009).

The final results of the 2011 Quaero evaluation campaign are obtained on the data described in section 2. The test data contains transcriptions of both broadcast news and broadcast conversation data. Results are provided on both manual and automatic transcriptions. In the last case, three different ASR systems were used in order to study robustness of the named entity detection systems with respect to different ASR errors and accuracies. These systems are referred to as **ASR1**, **ASR2** and **ASR3** and have word error rates of:

- 16.32%, 18.77%, 24.06% on broadcast news
- 23.34%, 22.99%, 29.18% on broadcast conversations
- 20.96%, 21.56%, 27.44% on the whole test data

[5]available at `http://wapiti.limsi.fr`

[6]available at `http://web.science.mq.edu.au/m̃johnson/Software.htm`

CRF model training		Incremental procedure	
Model features	SER	**Model features**	SER
Words	27.4%	Words+MS+WM unigrams	24.6%
+ MS	26.3%	–	–
+ WM	22.8%	+ Observation bigrams	20.6%
+ MS + WM	20.0%	+ Label bigrams	20.0%

Table 3: Results of CRF models on ESTER2 task obtained with a "normal" training procedure and our incremental procedure. **MS** are morpho-syntactic features, **WM** are features extracted from *WMatch* output

Model	**DEV**	**TEST**
CRF (SER)	24.8%	26.7%
$CYK_{ref.}$ (NER)	6.8%	7.4%
CYK (SER)	30.9%	33.3%

Table 4: Results of preliminary experiments obtained with the CRF model and with PCFG separately

Additionally, since manual transcriptions were provided with punctuation, the **ASR1** output has been automatically annotated with punctuation to try to fit manual transcription conditions.

All results are reported in terms of Slot Error Rate (SER) (Makhoul et al., 1999), which has a similar definition of word error rate for ASR systems, with the difference that correct entity with wrong boundaries and wrong entity with correct boundaries are given half points.

4.1 Results

4.1.1 Evaluation of the incremental procedure

Our incremental procedure was evaluated and results are reported in table 3. We compare two models trained and tested on the ESTER2 data used for the evaluation described in (Galliano et al., 2009). The two models are based on the same features, described in previous section, but use them in two different ways: In the first model, trained with "traditional procedure", the three different type of features (words, features from *tagger* and features from *WMatch*) were integrated in three different steps, using each time unigrams and bigrams on observations and labels. The three steps are mandatory since the total amount of features would not fit into memory.
In the second model, we used directly all type of features, but generating at first only simple features. 3.1.1. In the two other training steps, we added compound features, i.e. bigrams of observations and labels, and we retrained at each step.
As we can see from results in table 3, the two models reach the same final accuracy (a SER of 20.0%), which proves that our incremental training procedure doesn't leave out meaningful features. Additionally, although we don't report training time, we can comment that CRF model training was roughly 10 times faster with our incremental procedure. This is normal since in the second and third steps only compound features corresponding to simple features kept by the first model are added, with the additional constraint that simple features must correspond to adjacent positions. As explained in sub-section 3.1.1, the simple features kept at the end of the training are some orders of magnitude fewer in number than the original features. This limits tremendously the number of compound features added in the other steps. Note also that the results shown in Table 3 are much better than those shown in (Galliano et al., 2009), compared with other solutions.

Average score	Feature type
0.114053	wrd-2
0.0988316	Pre4
0.0914648	Wrd-1
0.084988	wrd-1
0.083699	Suf4
0.0751365	Pre3
0.0745788	WMatch2-2
0.0483077	WMatch1-1
–	...
0.00889771	POS+agree-1
0.00810903	WMatch4-1
0.00789857	POS-2
-0.0022887	POS+type-1
-0.0262062	POS-1
-0.0294334	Suf1
-0.0337793	Pre1

Table 5: Ranks of average score given by the CRF model to feature types

4.1.2 Features relevance

In order to understand features relevance, we report in table 5 feature types ranked by the average score given by the CRF model. Each type correspond to features at any position with respect to the target word, with label unigrams and bigrams. Unigrams are distinguished from bigrams using suffixes *-1* and *-2* respectively. Feature types *wrd* are words converted to lower case, *Wrd* are words kept with original capitalization. Feature types *Pren* are word prefixes of length n, *Sufn* are word suffixes of length n. Features extracted from *WMatch* output are indexed starting from 1. As we can see from the table, morpho-syntactic features (those marked with *POS*) receive quite low scores, especially POS tags. This point validates our intuition about using POS tags for named entity detection, pointed out in section 3. Note that feature types correspond to different layers of information added upon surface level, from less, like prefixes, to more general, like *WMatch* semantic layer. Thus, although some features may have outlier scores, the average score is a good indicator of the relevance of each feature type.

4.1.3 Models for tree-structured named entities

In table 4 we report an evaluation of the two models composing our system. In the first row we report the SER of the CRF model used in the system, taking into account only components, i.e. base entities annotated directly on words. They can be compared for the test set (TEST) with the 20% SER of table 3. We go from a SER of 20% to 26.7%, but it is important to note that the ESTER2 evaluation campaign was performed using only 17 labels, while 26.7% is obtained using 196 entity components. We can thus be satisfied by that result. In order to evaluate the robustness of the CYK algorithm on our task, we computed the Node Error Rate (NER in table 4) on reference components ($CYK_{ref.}$). The CYK module is applied on the reference components instead of those output by the CRF model. The rate of wrong tree nodes with respect to the reference trees is then computed. The NER being under 10%, we can assume that the PCFG model is robust on unseen data. This also confirms the effectiveness of using components directly, instead of the lexical surface forms. Finally we report the results obtained combining the two approaches (CRF+CYK). Errors of the CYK algorithm are summed to errors of CRF, thus we go from a SER of 24.8% and 26.7% on DEV and TEST (row 1, table 4), to 30.9% and 33.3%.

4.1.4 Official results

We report results of the 2011 Quaero named entity detection evaluation campaign (Galibert et al., 2011) in table 6, where **BN** correspond to broadcast news, **BN** to broadcast conversations and **Mrg** to the merge of these two types. Our system is indicated as *CRF+CYK*. The other two participants, *P1* and *P2*, used a system based on CRF and deep syntactic analysis, respectively.[7]

Looking at results we can see that our system outperforms the others in all cases by several points. Nevertheless error rates are quite high, over 50% on ASR output. The complexity of the task must without a doubt be taken into account. It is indeed the first time that structured named entities are handled with an automatic detection. Also the type of data used for the task, i.e. transcriptions of French broadcast data, contributes to increase the task complexity. Despite these results, our approach seems promising since it worked far better than the others in all cases.

[7]There are not more details on the other participant's systems, since they have not been published yet

		Manual	ASR1	ASR1+	ASR2	ASR3
BN	**WER**		16.32%	16.32%	18.77%	24.06%
	P1	42.7%	55.3%	52.7%	58.5%	61.4%
	P2	39.1%	55.6%	54.5%	60.3%	61.8%
	CRF+CYK	29.7%	48.5%	53.8%	52.2%	53.5%
CN	**WER**		23.34%	23.34%	22.99%	29.18%
	P1	55.3%	87.9%	89.9%	78.3%	89.2%
	P2	43.0%	89.3%	83.3%	81.2%	84.1%
	CRF+CYK	37.0%	73.9%	79.0%	66.6%	73.0%
Mrg	**WER**		20.96%	20.96%	21.56%	27.44%
	P1	48.9%	71.4%	71.1%	68.3%	75.2%
	P2	41.0%	72.2%	68.7%	70.7%	72.9%
	CRF+CYK	33.3%	61.1%	66.3%	59.3%	63.2%

Table 6: SER results for the 2011 evaluation campaign on broadcast news (**BN**), broadcast conversation (**BC**) and their merge (**Mrg**) on both manual and automatic transcriptions

Oracles VS Evaluations	**DEV**	**TEST**
CRF+CYK (SER)	30.9%	33.3%
CRF+CYK (OER)	18.6%	21.1%

Table 7: Comparison of Oracle Error Rate (OER) and SER obtained in the evaluation for DEV and TEST sets

We report a comparison of results on manual transcriptions with Oracle Error Rates (OER), i.e. results of our system using the best annotation of CRF among the 10-best hypotheses. This comparison is reported in table 7, and shows that we have a large margin for improvements on our system.

5 Conclusions

In this paper we proposed a system for structured named entity detection. We describe the definition of these structured named entities. The proposed system is based on CRF and syntactic parsing approaches, which combines the effectiveness and robustness of the former with the capability of easily and quickly parsing trees of the latter. Additionally, we proposed an incremental training procedure for CRF model, which showed to be correct and effective and allows to train CRF models with huge number of features. The proposed system participated in the Quaero evaluation campaign and obtained the best results. Although results on ASR output are not satisfactory, taking all results into account the proposed approach seems promising and encourages further studies.

Acknowledgments

This work was partly realized as part of the Quaero Programme, funded by Oseo, French State agency for innovation

References

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of NLL at HLT-NAACL 2003 - Volume 4*, pages 142–147, Morristown, USA.

Ellen M. Voorhees. 2001. The trec question answering track. *Nat. Lang. Eng.*, 7:361–378, December.

X. Carreras and Lluis Marquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling.

R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur. 2008. Spoken language understanding: A survey. *IEEE Signal Processing Magazine*, 25:50–58.

Sylvain Galliano, Guillaume Gravier, and Maura Chaubard. 2009. The ester 2 evaluation campaign for the rich transcription of french radio broadcasts. In *Proceedings of Interspeech*, Brighton, U.K.

Olivier Galibert, Sophie Rosset, Cyril Grouin, Pierre Zweigenbaum, Ludovic Quintard 2011. Structured and Extended Named Entity Evaluation in Automatic Speech Transcriptions In *Proceedings of IJCNLP*, Chang Mai, Thailand.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*, page 363370, Ann Arbor, MI.

Michael Collins and Terry Koo. 2005. Discriminative re-ranking for natural language parsing. *Computational Linguistic (CL)*, 31(1):25–70.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, Williamstown, USA.

T.L. Booth and R.A. Thompson. 1973. Applying Probability Measures to Abstract Languages. In IEEE Transactions on Computers, 22:442–450.

Brigitte Krenn and Christer Samuelsson. 1997. The linguist's guide to statistics - don't panic.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings ACL*, pages 504–513.

Stefan Hahn, Patrick Lehnen, Georg Heigold, and Hermann Ney. 2009. Optimizing crfs for slu tasks in various languages using modified training criteria. In *Proceedings of Interspeech*, Brighton, U.K.

Stefan Riezler and Alexander Vasserman. 2010. Incremental feature selection and l1 regularization for relaxed maximum-entropy modeling.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society B*, 67:301–320.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *19th Conference on Uncertainty in Artificial Intelligence*.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

Olivier Galibert, Ludovic Quintard, Sophie Rosset, Pierre Zweigenbaum, Claire Ndellec, Sophie Aubin, Laurent Gillard, Jean-Pierre Raysz, Delphine Pois, Xavier Tannier, Louise Delger, and Dominique Laurent. 2010. Named and specific entity detection in varied data: The quro named entity baseline evaluation. In *Proceedings of LREC*, La Valletta, Malta.

Olivier Galibert. 2009. *Approches et méthodologies pour la réponse automatique à des questions adaptées un cadre interactif en domaine ouvert*. Ph.D. thesis, Universit Paris Sud, Orsay.

Sophie Rosset, Olivier Galibert, Fuillaume Bernard, Eric Bilinski, and Gilles Adda. 2008. The LIMSI participation to the QAst track. In Working Notes of CLEF 2008 Workshop.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. 2004. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. *Proceedings of LREC 2004*, pages 837–840.

Sam Coates-Stephens. 1992. The analysis and acquisition of proper names for the understanding of free text. *Computers and the Humanities*, 26:441–456.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference - 6: A brief history. In *Proceedings of COLING*, pages 466–471, Copenhagen, Denmark.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of COLING*, pages 1–7.

Michael Fleischman. 2001. Automated subcategorization of named entities. In *Proceedings of the ACL 2001 Student Research Workshop*, pages 25–30.

Eckhard Bick. 2004. A named entity recognizer for danish. In *LREC'04*, Lisbon, Portugal.

Satoshi Sekine. 2004. Definition, dictionaries and tagger of extended named entity hierarchy. In *LREC'04*, Lisbon, Portugal.

Alexandre Allauzen and Hélène Bonneau-Maynard. 2008. Training and evaluation of pos taggers on the french multitag corpus. In *Proceedings of LREC*, Marrakech, Morocco.

John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. 1999. Performance measures for information extraction. In *Proceedings of DARPA Broadcast News Workshop*, pages 249–252.

Cyril Grouin, Sophie Rosset, Pierre Zweigenbaum, Karen Fort, Olivier Galibert, Ludovic Quintard. 2011. Proposal for an Extension or Traditional Named Entities: From Guidelines to Evaluation, an Overview. In *Proceedings of the Linguistic Annotation Workshop (LAW)*.