Robust Information Extraction with Perceptrons

Mihai Surdeanu
Technical University of Catalonia
surdeanu@lsi.upc.edu

Massimiliano Ciaramita
Yahoo! Research Barcelona
massi@yahoo-inc.com

Abstract

We present a system for the extraction of entity and relation mentions. Our work focused on robustness and simplicity: all system components are modeled using variants of the Perceptron algorithm (Rosemblatt, 1858) and only partial syntactic information is used for feature extraction. Our approach has two novel ideas. First, we define a new large-margin Perceptron algorithm tailored for class-unbalanced data which dynamically adjusts its margins, according to the generalization performance of the model. Second, we propose a novel architecture that lets classification ambiguities flow through the system and solves them only at the end. The system achieves competitive accuracy on the ACE English EMD and RMD tasks.

1 Introduction

Within the Information Extraction (IE) community the Automatic Content Extraction (ACE)[1] program provides an evaluation platform that is currently the *de facto* standard for the evaluation of IE systems. The work presented in this paper falls within the scope of two important tracks of the ACE program: (a) Entity Mention Detection (EMD), which evaluates the identification and classification of entity mentions, and (b) Relation Mention Detection (RMD), which involves the extraction of binary relation mentions between ACE entities. Figure 1 shows a sample text containing three ACE entity mentions and two relation mentions. As an example, the noun phrase headed by "building" is the mention of an entity of type `FACILITY` and subtype `Building-Grounds`. The relation mentions can be symmetrical, which hold no matter the order of the two arguments, and asymmetrical, where the argument order is important; e.g., between "building" and "Marines" there is a symmetrical relation of type `PHYSICAL` and subtype `Located`, whereas between "building" and "Shatra" there is an asymmetrical relation of type `PART-WHOLE` and subtype `Geographical`.

This paper describes a system for the extraction of both entity and relation mentions. The methods presented are evaluated on the English ACE corpus but all the algorithms introduced are language independent. The approach proposed in this paper has several novel points:

- All learning tasks in the proposed system are implemented using variants of the Perceptron Algorithm (PA). Furthermore, we introduce a new large-margin PA tailored for unbalanced data. We show that in the RMD task the algorithm performs better than both Support Vector Machines (SVM) and regular Perceptron.
- We use a novel strategy to mitigate errors in early stages of the system, such as entity mention classification. If entity classification ambiguities are detected (with a dedicated learning-based component) we let them

[1]http://www.nist.gov/speech/tests/ace/

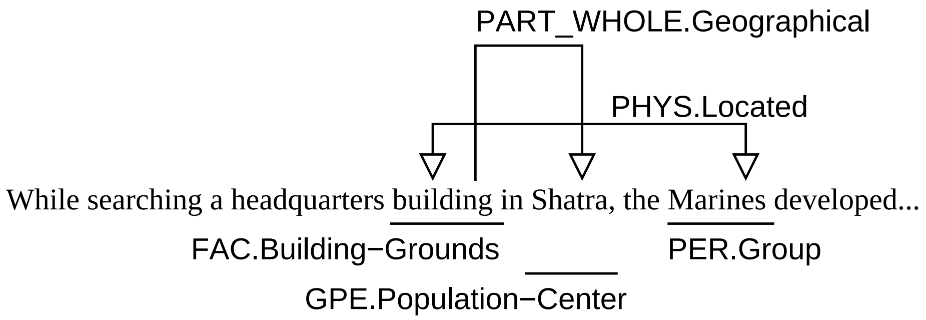

Figure 1: Sample text annotated with ACE entity and relation mentions.

trickle through the other learning components (i.e., RMD) and solve them only at the end using an approximated-inference algorithm.

Our system obtains competitive results on the two tasks, and especially on RMD, where both of the above issues are fully exploited. We see these results as very encouraging considering that: (a) we use minimal syntactic analysis of the text (i.e., only part-of-speech (POS) tagging and chunking), (b) in the learning components we use only linear kernels with a simple feature space, and (c) we do not use any form of co-reference.

The paper is organized as follows. Section 2 overviews the architecture of the full system. The EMD system is detailed in Section 3. The ambiguity detection system for entity classification is introduced in Section 4. Section 5 describes the RMD component including the novel Perceptron algorithm. Section 6 contains the empirical analysis of the system and Section 7 concludes the paper.

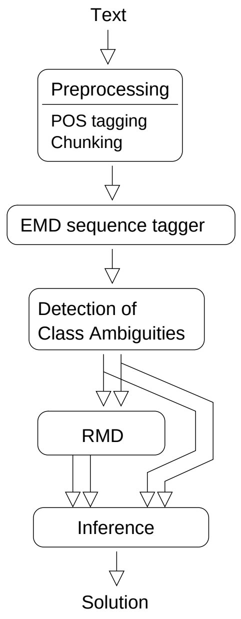

Figure 2: System architecture. The double lines indicate ambiguities in entity or relation extraction.

2 Architecture

Figure 2 shows the IE architecture proposed in this paper. The system execution flow starts with a preprocessing step where the text is tokenized, POS-tagged, and basic syntactic phrases, i.e., chunks, identified. For POS tagging we use the TnT tagger (Brants, 2002)[2]. For syntactic analysis we use an in-house chunker based on the YamCha toolkit[3] trained on the Penn TreeBank.

The next component identifies the boundaries of entity mentions and for each extracted mention it detects its entity type and subtype. We model all these operations jointly using a sequence tagger that assigns a Begin/Inside/Outside (BIO) label to each of tokens in the document word sequences. The BIO labels are extended with a concatenation of the entity type and subtype. For example, the label `B-FAC-Plant` indicates that the corresponding token begins a mention of an entity of type `FACILITY` and subtype `Plant`. The sequence tagger uses the PA for sequence learning (Collins, 2002), which optimizes the choice of labeling globally at sentence level (cf. Section 3).

The next component detects ambiguities in the assignment of entity types and subtypes. The motivation for the inclusion of this component is that the

[2] `http://www.coli.uni-saarland.de/~thorsten/tnt`
[3] `http://chasen.org/~taku/software/yamcha`

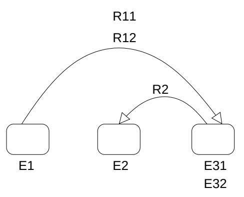

Figure 3: Sentence with an ambiguous solution: entity E3 and relation R1 each have two possible labels.

EMD tagger performs well for the detection of mention boundaries but less well in classifying them, we present an empirical analysis of the EMD tagger in Section 6.3. When there are ambiguities in entity classification this module lets several entity classes pass through to RMD. We implement this operation as a re-classification task for each entity mention detected in the previous step. Classification ambiguities are detected with a beam heuristic: for every entity we accept all classes generated with a probability within a certain threshold of the top class's probability. This classifier is implemented with the averaged PA of (Freund and Shapire, 1999). We use a separated instance of the same classifier to detect the type of each entity mention, i.e., nominal (`NOM`), pronominal (`PRO`), or name (`NAM`). We describe this classifier in Section 4.

We model the RMD task as a classification problem. That is, every pair of entity mentions is a possible relation. The candidate relation is a negative example if no actual relation exists between the two entities, or a positive example otherwise. Positive examples are labeled with a relation class that concatenates the relation type, subtype, and direction. This approach yields a very unbalanced sample space where the ratio of negative to positive examples is very large (e.g., more than 13 to 1 in the ACE training corpus). To address this problem we propose a new PA tailored for such class-unbalanced scenarios. We detail this algorithm in Section 5 and show that it outperforms both the averaged PA and SVM in Section 6.3. The output of this component is a beam-based set of multiple relation classes when the corresponding relation is ambiguous.

The outcome of this process can be highly ambiguous: each detected entity or relation mention is possibly assigned to more than one class. Figure 3 gives an example where one entity mention and one relation mention are assigned to two possible classes. The last system component implements the inference mechanism necessary to identify the best solution which will be the final system output. The algorithm works in two steps:

1. *Candidate generation*. For each sentence we generate all possible candidates. For example, the candidates generated for the output shown in Figure 3 are: {`R11(E1, E31), R2(E31, E2)`}, {`R11(E1, E32), R2(E32, E2)`}, {`R12(E1, E31), R2(E31, E2)`}, {`R12(E1, E32), R2(E32, E2)`}. Note that in this step we consider only a subset of all possible candidates since the previous beam-based filters eliminate many entity and relation classes unlikely to be correct. This inference strategy falls in the category of approximated inference rather than exact inference.

2. *Candidate search*. We search for the best solution by picking the sentence candidate that has the highest confidence and is consistent with the ACE domain constraints. As an example, according to the definition of ACE relations, a `PHYS.Located` relation may occur only between a `PER` entity and a `FAC`, `LOC`, or `GPE` entity. We compute the confidence in a sentence candidate with **E** entities and **R** relations with the following formula:

$$conf(\mathbf{E},\mathbf{R}) = \lambda_e \sum_{i=1}^{|\mathbf{E}|} p(E_i) + \lambda_r \sum_{i=1}^{|\mathbf{R}|} p(R_i) \tag{1}$$

   where p is the probability of the corresponding class and λ_e and λ_r are parameters indicating the confidence assigned to the entity and relation classification models (the larger the better). Since the Perceptron does not output probabilities we convert the model raw activations to true probabilities using the *softmax* function (Bishop, 1995).

The proposed architecture is closest in spirit to the work of (Roth and Yih, 2004). There are however two significant differences between our work and theirs. First, ours uses approximated inference whereas (Roth and Yih, 2004) use exact inference implemented with a Constraint Satisfaction

(CS) model. Their approach is guaranteed to find the overall best solution, but it suffers the cost of searching through a very large candidate space, i.e, all possible candidates are generated, involving an additional software module (the CS software). Second, the EMD and RMD in (Roth and Yih, 2004) are disjoint and independent, whereas in our implementation the RMD classifier uses as features the output of the second entity classifier. We show in Section 6.3 that feeding the EMD output to RMD is beneficial, even if it is ambiguous.

3 Entity Mention Detection as Sequential Tagging

We take a sequence labeling approach to learning a model for detecting entity mentions. The objective is to learn a function from input vectors, i.e., the observations from labeled data, to response variables, i.e., the entity labels. Previous work on POS tagging, shallow parsing, NP-chunking and NER has shown that performance can be significantly improved by optimizing the choice of labeling over whole sequences of words, rather than individual words. To model sequential labeling we adopt the Perceptron-trained Hidden Markov Model (HMM) originally proposed in (Collins, 2002).

3.1 Approach

HMMs define a probabilistic model for observation/label sequences. The joint model of an observation/label sequence $(\mathbf{x}, \mathbf{y})$, is defined as:

$$P(\mathbf{y}, \mathbf{x}) = \prod_i P(y_i|y_{i-1})P(x_i|y_i), \tag{2}$$

where y_i is the ith label in the sequence and x_i is the ith word. A common variant involves modeling the conditional distribution of label sequences given observation sequences.

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|x_i, y_{i-1}). \tag{3}$$

Discriminative approaches to sequence labeling (McCallum et al., 2000; Lafferty et al., 2001; Collins, 2002; Altun et al., 2003) have several advantages over generative models, such as not requiring questionable independence assumptions, optimizing the conditional likelihood directly and employing richer feature representations.

The learning task can be framed as learning a discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow I\!R$, on a training data of observation/label sequences, where F is linear in a feature representation Φ defined over the joint input/output space

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle. \tag{4}$$

Φ is a global feature representation, mapping each $(\mathbf{x}, \mathbf{y})$ pair to a vector of feature counts $\Phi(\mathbf{x}, \mathbf{y}) \in I\!R^d$, where d is the total number of features. This vector is given by

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} \sum_{j=1}^{|\mathbf{y}|} \phi_i(y_{j-1}, y_j, \mathbf{x}). \tag{5}$$

Each individual feature ϕ_i extracts a morphological or contextual feature, and the dependencies between consecutive labels. The features used are described in detail below in Section 3.2. Given an observation sequence $\mathbf{x}$, we make a prediction by maximizing F over the entity sequence variable:

$$f_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}). \tag{6}$$

This involves computing the Viterbi decoding, with respect to the parameter vector $\mathbf{w} \in I\!R^d$, whose complexity is linear in the size of the sequence. To estimate $\mathbf{w}$ we use the sequence perceptron algorithm (Collins, 2002). The perceptron minimizes the error rate, without involving normalization factors and provides a very simple method. The performance of Perceptron-trained HMMs has proven competitive on a number of tasks; e.g., in shallow parsing, where the Perceptron performance is comparable to that of Conditional Random Field models (Sha and Pereira, 2003), We mitigate the tendency to overfit of the perceptron by regularizing the model by means of averaging, straightforwardly extending Collins' method, summarized in Algorithm 1.

3.2 Features

We used the following combination of spelling/morphological and contextual features. For each observed word x_i in the data ϕ extracts the following features:

1. **Words:** $x_i,\ x_{i-1},\ x_{i-2},\ x_{i+1},\ x_{i+2}$;

Algorithm 1: Hidden Markov Average Perceptron

```
input : $\mathcal{S} = (\mathbf{x}_i, y_i)^N; \mathbf{w}^0 = \vec{0}$
for $t = 1$ to $T$ do
    choose $\mathbf{x}_j$
    compute $\hat{\mathbf{y}} = f_{\mathbf{w_t}}(\mathbf{x_j})$
    if $\hat{\mathbf{y}} \neq \mathbf{y}_j$ then
        $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Phi(\mathbf{x}_j, \mathbf{y}_j) - \Phi(\mathbf{x}_j, \hat{\mathbf{y}})$
output: $\mathbf{w} = \frac{1}{T}\sum_t \mathbf{w}_t$
```

2. **First sense:** supersense baseline prediction for x_i, $\mathsf{fs}(x_i)$;

3. **Combined (1) and (2):** $x_i + \mathsf{fs}(x_i)$;

4. **Pos:** $\mathsf{pos_i}$ (the POS of x_i), $\mathsf{pos_{i-1}}$, $\mathsf{pos_{i-2}}$, $\mathsf{pos_{i+1}}$, $\mathsf{pos_{i+2}}$, $\mathsf{pos_i}[0]$, $\mathsf{pos_{i-1}}[0]$, $\mathsf{pos_{i-2}}[0]$, $\mathsf{pos_{i+1}}[0]$, $\mathsf{pos_{i+2}}[0]$, $\mathsf{pos\_comm_i}$ if x_i's POS tags is "NN" or "NNS" (common nouns), and $\mathsf{pos\_prop_i}$ if x_i's POS is "NNP" or "NNPS" (proper nouns);

5. **Word shape:** $\mathsf{sh}(x_i)$, $\mathsf{sh}(x_{i-1})$, $\mathsf{sh}(x_{i-2})$, $\mathsf{sh}(x_{i+1})$, $\mathsf{sh}(x_{i+2})$, where $\mathsf{sh}(x_i)$ is as described below. In addition $\mathsf{sh_i} = \mathsf{low}$ if the first character of x_i is lowercase, $\mathsf{sh_i} = \mathsf{cap\_brk}$ if the first character of x_i is uppercase and x_{i-1} is a full stop, question or exclamation mark, or x_i is the first word of the sentence, $\mathsf{sh_i} = \mathsf{cap\_nobrk}$ otherwise;

6. **Previous label:** entity label y_{i-1}.

Words (1) are morphologically simplified using the "morph" function provided by WordNet (Fellbaum, 1998). The first sense feature (2) is a coarse-grained WordNet sense predicted for x_i by the baseline model described in (Ciaramita and Altun, 2006). POS features of the form $\mathsf{pos_i}[0]$ extract the first character from the POS label – a coarse POS tag. Word shape features (5) are regular expression-like transformation in which each character c of a string s is substituted with X if c is uppercase, if lowercase, c is substituted with x, if c is a digit it is substituted with d and left as it is otherwise. In addition each sequence of two or more identical characters c is substituted with $c*$. For example, for s = "Merrill Lynch& Co.", $\mathsf{sh(s)} = \mathsf{Xx} * \mathsf{Xx} * \&\mathsf{Xx}$..

To benefit from higher-order feature representations, after extracting each observation vector, we apply an additional feature map, Φ^2. This extracts all second order features of the form $x_i x_j$; i.e., $\mathbf{\Phi}^2(\mathbf{x}) = (x_i, x_j)^{(d,d)}_{(i,j)=(1,1)}$. This feature map is equivalent to adopting a polynomial kernel function of degree 2 in a dual model. Training a dual model with large datasets is impractical, due to the fact that it is not possible to cache the full Kernel matrix. Instead, using a second order map in the primal model, inflates considerably the feature space (we find more than 10 million features) but makes training still considerably faster than in the dual model[4].

4 Entity Classification as Ambiguity Detection

As mentioned in Section 2 the task of this component is to reclassify all the entity mentions detected by the EMD sequence tagger in order to detect ambiguities, i.e., entity mentions that are assigned several classes with close probabilities.

4.1 Approach

We implement the entity classifier using the standard averaged PA. See (Freund and Shapire, 1999) for details on this algorithm. We converted the raw activations generated by this algorithm to true probabilities (required by the beam filter) using the *softmax* function (Bishop, 1995). An important difference between this classifier and the previous EMD sequence tagger is that this classifier works at entity level rather than word level. This setting allows us to generate more complex features (cf. below).

4.2 Features

The features used for entity classification are essentially n-grams of the words inside or in the immediate context of the given mention. We list these features in Table 1. The $token$ function extracts the word, lemma, and POS tag of a given token. The $tokens$ function constructs unigrams and bigrams of words, lemmas, and POS tags for a given sequence of tokens. We apply these two functions to the head word of the current mention (usually the last word in the mention), the words inside the entity, the entity left context (the context size spans two words), and

[4]In practice it is sufficient to consider pairs with $i \leq j$.

$token$(entity head word)
WordNet SuperSense of head word
BBN class of head word
$tokens$(entity inside words)
$tokens$(entity left context)
$tokens$(entity right context)
true if entity is known person name
true if entity is known location

Table 1: Feature types used for entity classification.

the entity right context. As additional features we use the WordNet SuperSense of the entity head word (extracted using the tagger described in (Ciaramita and Altun, 2006), but without the additional second-order feature map), the BBN class of the entity head word (extracted using the same tagger, but trained on the BBN Entity corpus[5]), and two Boolean flags which indicate if the current mention is a known person or location name.

5 Relation Mention Detection with Perceptron with Dynamic Uneven Margins

As previously mentioned, a key feature of the ACE RMD problem is the large unbalance between positive and negative examples in the data. To address this issue, we propose a new large-margin PA where the margins are: (a) different for positive and negative examples to model the unbalance in the data, and (b) adjusted on-line according to the generalization performance of the model. We call this algorithm the Perceptron Algorithm with Dynamic Uneven Margins (PADUM). We detail the algorithm next.

5.1 Approach

Our approach is based on two observations:

(a) Maximum or large margin classifiers exhibit good generalization performance. This observation was the motivation behind SVM (Cristianini and Shawe-Taylor, 2000). (Krauth and Mezard, 1987) define a new PA called Perceptron Algorithm with Margins (PAM), which learns large-margin classifiers by doing a more conservative parameter update in training. Unlike the PA, the PAM performs vector updates not only when the prediction is incorrect, but also when the model is not confident enough, i.e., the predicted margin is smaller than a constant τ. The PAM converges more slowly than the PA but the classifier learned is guaranteed to have a large margin.

(a) Treat positive and negative examples differently in unbalanced data. (Li et al., 2002) discuss that for data where the ratio of positive to negative examples is very small it is more important to classify correctly a positive example than a negative one. (Li et al., 2002) introduce a variation of the PAM, called Perceptron Algorithm with Uneven Margin (PAUM), which uses two margin parameters, one for positive examples, τ_{+1}, and another for negative examples, τ_{-1} (typically $\tau_{+1} \gg \tau_{-1}$). Intuitively, the PAUM gives more importance to positive than negative examples by learning classifiers with margins that are larger for the former class of examples. They showed that the PAUM has similar theoretical properties as the PAM, but it outperforms other on-line algorithms and SVM for highly-unbalanced scenarios.

PADUM is a direct descendant of the PAUM and is motivated by the fact that tuning PAUM's margin parameters $\tau_{\pm 1}$ is both important and difficult. For example, modeling the ACE RMD problem with a one-versus-rest approach yields 33 binary classifiers, each requiring a separate manual tuning process for the $\tau_{\pm 1}$ parameters. Setting incorrect values for $\tau_{\pm 1}$ yields several undesired side effects. For example, a value too small for τ_{+1} means that the PAUM acquires too few positive examples and the resulting model fails to generalize well. This is the typical behavior of the PA, which has $\tau_{\pm 1} = 0$. On the other hand, setting a value too large for τ_{+1} signifies that the PAUM acquires two many positive examples in **w**, with the effect that the model is too eager in predicting positive examples. This yields a classifier with an excessive bias towards recall in detriment of precision.

Instead of relying on static margin parameters, the PADUM has a built-in tuning process for the $\tau_{\pm 1}$ parameters based on the following intuition:

> *The margin parameters $\tau_{\pm 1}$ are inversely proportional with the classifier general-*

[5] BBN Pronoun Co-reference and Entity Type Corpus, Linguistic Data Consortium (LDC) catalog number LDC2005T33.

ization performance for positive/negative examples.

In other words, if the classifier has good performance, the PADUM converges faster by decreasing the values of $\tau_{\pm 1}$, which reduces the number of model updates. If the classifier does not generalize, the PADUM maintains large values for $\tau_{\pm 1}$, which means that the algorithm continues to learn until the classifier learned has sufficiently large margins. We quantify the generalization performance of the classifier using its classification error rate on the training set. We measure the generalization performance separately for positive and negative examples to address the unbalance in the data.

Figure 2 summarizes the algorithm. For simplicity here we assume that each sample **x** is already expanded to its feature vector. The PADUM works on a training sample $\mathcal{Z}$ and learns a set of weighted prediction vectors $(\mathbf{w}_k, c_k)$, where the weight c_k indicates how many iterations has the $\mathbf{w}_k$ model survived unchanged. The $(\mathbf{w}_k, c_k)$ vectors are used to compute the averaged prediction vector **avg**, using the strategy proposed by (Freund and Shapire, 1999). Step *(b)* in the algorithm inner loop is essentially the PAUM: the model is updated when the predicted margin is smaller than the corresponding $\tau_{\pm 1}$ parameter. The essence of the PADUM is step *(c)* where the margin parameters $\tau_{\pm 1}$ are adjusted according to the classification error rate (computed in step *(a)*). In this paper we use a simple linear function to represent the dependencies between $\tau_{\pm 1}$ and $err_{\pm 1}$. Arguably, there are other better functions to model the dependency between $\tau_{\pm 1}$ and $err_{\pm 1}$. This remains to be investigated in future work. The algorithm parameters are: the number of learning epochs T and the initial values of the margin parameters, $\Gamma_{\pm 1}$. Note that tuning of the $\Gamma_{\pm 1}$ values is significantly simpler than tuning the static $\tau_{\pm 1}$ in PAUM. For example, in all our experiments we set Γ_{+1} to the largest acceptable value (1.0 because we work with normalized vectors), and let PADUM adjust the margin parameters on its own.

Algorithm 2: Perceptron Algorithm with Dynamic Uneven Margins

```
input : $\mathcal{Z} = (\mathbf{x}, \mathbf{y}) \in (\mathcal{X} \times \{-1, +1\})^m$,
        $\Gamma_{-1}, \Gamma_{+1} \in \mathbb{R}^+$
        $T, \mathbf{w}_1 = \vec{0}, c_1 = 0, k = 1$
for $j \in \{-1, +1\}$ do
    $\tau_j \leftarrow \Gamma_j$
    visited$_j \leftarrow 0$
    incorrect$_j \leftarrow 0$
for $t = 1$ to $T$ do
    for $i = 1$ to $m$ do
        (a) compute prediction error rate:
        for $j \in \{-1, +1\}$ do
            if $y_i = j$ then
                visited$_j \leftarrow$ visited$_j + 1$
                if $y_i \langle \mathbf{w}_k, \mathbf{x}_i \rangle \leq 0$ then
                    incorrect$_j \leftarrow$ incorrect$_j + 1$
                err$_j \leftarrow \frac{\text{incorrect}_j}{\text{visited}_j}$
        (b) update vectors:
        if $y_i \langle \mathbf{w}_k, \mathbf{x}_i \rangle \leq \tau_{y_i}$ then
            $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + y_i \mathbf{x}_i$
            $c_{k+1} \leftarrow 1$
            $k \leftarrow k + 1$
        else
            $c_k \leftarrow c_k + 1$
        (c) update margins:
        for $j \in \{-1, +1\}$ do
            $\tau_j \leftarrow$ err$_j \Gamma_j$
output: avg $= \sum_{i=1}^{k} c_i \mathbf{w}_i$
```

5.2 Features

The features used for the RMD task are inspired from (Zhou et al., 2005; Claudio et al., 2006) and are based only on lexical, morphological, and par-

$tokens$(head words of relation arguments)
$entities$(relation arguments)
$tokens$(words between relation arguments)
$tokens$(chunks between relation arguments)
$path$(chunks between relation arguments)
$tokens$(words in the relation left context)
$tokens$(chunks in the relation left context)
$tokens$(words in the relation right context)
$tokens$(chunks in the relation right context)

Table 2: List of RMD features types.

tial syntactic information. We do not use full syntactic analysis nor any kind of semantic information (outside of the predicted entity classes). We list the feature set in Table 2.

The $tokens$ function constructs unigrams and bigrams of words, lemmas, and POS tags for a given sequence of tokens. We apply this function to the two head words of the relation arguments, the words between/before/after the two relation arguments, and the head words of the chunks between/before/after the arguments. In all experiments reported in this paper we use a context size of 1 word (or chunk) to the left/right of the corresponding relation. The $entities$ function extracts the top N predicted entity classes for the two arguments and constructs all possible combinations between them. Each entity class is expanded in this combination with its position in the list of predicted classes, e.g., 1 for the class with the highest confidence, 2 for the second, etc. The $path$ function constructs two sequences, one of chunk syntactic labels and one of head words, for the sequence of chunks between the two relation arguments.

6 Experiments

In this section we report our results in the official ACE 2007 evaluation and also an analysis of the key features of our system: the PADUM for RMD and the strategy to handle entity classification ambiguities.

6.1 Setup

We trained and tested our IE system on the ACE 2007 English data. The corpus contains 599 files for training, 254 for EMD testing, and 155 for RMD testing (the RMD test corpus is a subset of the EMD test set). The corpus is annotated with 7 entity types subdivided in 44 subtypes, and 6 relation types with 18 subtypes. Since the corpus does not have a development section, we tuned the parameters of our system on the training section using five-fold cross validation.

After tuning, we configured the system with the following parameters. We used $\lambda_e = 1.0$ and $\lambda_r = 0.5$ in the combination stage. Intuitively, this indicates that we trust the EMD system twice as much as the RMD system. This matches our empirical results: the ACE cost-based value for EMD is roughly twice the RMD score. With respect to the beam-based inference, we let the top 20 entity/relation classes enter the combination phase for each entity/relation candidate. To avoid flooding the RMD classifier with entity-based features, we used a different beam filter for the EMD-RMD interaction: for each entity candidate we use the top 2 entity classes if the second-best class is predicted with a probability larger than the top-class probability divided by 100. Otherwise, we use only the top entity class.

We performed little tuning of the PADUM for RMD. We set $\Gamma_{+1} = 1.0$, which is the largest acceptable value since we work with normalized feature vectors, and $\Gamma_{-1} = 0.01$.

6.2 Evaluation Results

Table 8 lists our official ACE score for the EMD evaluation. For brevity we include only the scores for the seven entity types. Tables 4 and 5 list the type and subtype scores for the RMD evaluation. The three tables indicate that our IE system has robust performance on the two tasks: we obtain a cost-based EMD value score of 75.0 (with a value-based F score of 83.3), and a cost-based RMD value score of 33.1 (with a value-based F score of 54.5). We find these results very encouraging: we obtain state-of-the-art RMD results with a very simple architecture and feature space, using only linear kernels in the learning tasks, and without any form of co-reference resolution. The next subsection shows that most of the performance boost is caused by the PADUM, and some by the novel system architecture.

In the EMD task we score above average on the entities that are well represented in the data and for which we have additional features, e.g., `GPE` and `PER`, and not so well for classes with few examples in the data, e.g., `FAC` and `WEA`. Note that we have not used any ACE-specific gazetteers in this paper. The same behavior repeats for RMD: we score badly for relations with very few examples, e.g., `PART-WHOLE.Artifact` or `ORG-AFF.Founder`, and score well for relations that are unambiguous and/or have more examples, e.g., `PER-SOC.Family` or `ORG-AFF.Employment`. These observations highlight the fact that even an algorithm tailored for sparse data such as the PADUM has its limitations.

	Count				Cost (%)						
	Ent	Detection		Rec	Detection		Rec	Value	Value-based		
	Tot	FA	Miss	Err	FA	Miss	Err	(%)	Pre	Rec	F
FAC	719	67	244	212	8.6	25.9	14.4	51.1	72.2	59.7	65.3
GPE	3198	165	385	775	3.6	10.1	10.8	75.6	84.7	79.2	81.8
LOC	422	50	135	152	10.2	22.9	17.3	49.6	68.5	59.8	63.8
ORG	2677	157	475	1119	5.8	16.4	14.1	63.6	77.7	69.5	73.4
PER	10359	560	804	2285	6.9	8.2	1.7	83.2	91.3	90.1	90.7
VEH	413	16	118	95	3.2	25.7	4.7	66.4	89.8	69.6	78.4
WEA	335	21	124	136	10.6	42.0	2.6	44.8	80.8	55.4	65.7
total	18123	1036	2285	4774	5.8	11.8	7.4	75.0	85.9	80.8	83.3

Table 3: EMD scores in the ACE evaluation for the seven entity types.

	Count				Cost (%)						
	Ent	Detection		Rec	Detection		Rec	Value	Value-based		
	Tot	FA	Miss	Err	FA	Miss	Err	(%)	Pre	Rec	F
ART	261	38	157	84	9.1	63.9	2.5	24.5	74.2	33.6	46.2
GEN-AFF	235	28	120	92	9.1	51.5	5.0	34.5	75.6	43.6	55.3
ORG-AFF	503	71	216	237	9.6	45.4	4.0	41.0	78.9	50.6	61.6
PART-WHOLE	354	57	182	110	12.1	48.9	2.2	36.8	77.4	48.9	59.9
PER-SOC	213	24	90	116	5.6	38.5	2.4	53.5	88.0	59.1	70.7
PHYS	428	76	298	113	8.7	69.1	6.2	16.0	62.3	24.7	35.4
total	1994	294	1063	752	9.4	53.5	4.0	33.1	76.1	42.5	54.5

Table 4: RMD scores in the ACE evaluation for the six relation types.

	Count				Cost (%)						
	Ent	Detection		Rec	Detection		Rec	Value	Value-based		
	Tot	FA	Miss	Err	FA	Miss	Err	(%)	Pre	Rec	F
Artifact	14	0	13	1	0.0	92.0	2.4	5.6	70.0	5.6	10.4
Business	63	4	39	24	2.2	63.8	3.4	30.7	85.6	32.8	47.5
Citizen...	171	23	83	73	10.5	49.6	5.7	34.1	73.3	44.6	55.5
Employment	344	61	113	189	12.1	34.8	4.0	49.1	79.1	61.2	69.0
Family	118	19	32	79	8.6	20.9	0.4	70.1	89.7	78.7	83.8
Founder	6	0	5	1	0.0	88.8	3.4	7.8	70.0	7.8	14.1
Geographical	223	33	102	71	10.4	42.0	1.9	45.7	82.1	56.1	66.7
Investor...	8	0	5	3	0.0	57.1	2.9	40.0	93.3	40.0	56.0
Lasting-Personal	32	1	19	13	1.9	50.6	7.8	39.8	81.2	41.6	55.0
Located	382	72	263	102	9.2	68.3	6.6	15.9	61.4	25.1	35.6
Membership	96	8	55	33	6.0	61.3	4.2	28.5	77.2	34.5	47.7
Near	46	4	35	11	4.9	75.2	3.2	16.7	72.8	21.6	33.3
Org-Location	64	5	37	19	5.9	55.6	3.2	35.3	82.0	41.2	54.8
Ownership	15	2	13	2	5.0	87.5	0.0	7.5	71.4	12.5	21.3
Sports-Affiliation	17	0	15	2	0.0	88.4	3.5	8.1	70.0	8.1	14.6
Student-Alum	17	0	10	7	0.0	60.0	7.5	32.5	81.2	32.5	46.4
Subsidiary	117	24	67	38	16.1	58.8	2.9	22.2	66.8	38.3	48.7
User-Owner...	261	38	157	84	9.1	63.9	2.5	24.5	74.2	33.6	46.2
total	1994	294	1063	752	9.4	53.5	4.0	33.1	76.1	42.5	54.5

Table 5: RMD scores in the ACE evaluation for the 18 relation subtypes.

We believe that in order to achieve truly operational performance one has to look at semi-supervised methods and/or knowledge-rich systems.

With respect to the quantitative performance, all EMD system processes were performed on a 2.4GHz AMD Opteron machine. We applied the second-order feature map without modifications to the original tagger implementation, which is not conceived for handling efficiently tens of millions of features. While the second-order tagger is slow, it could be substantially optimized to achieve better performance. Currently, the second-order tagger takes about 1 hour/epoch to train. The trained system in prediction labels about 50 words/second. The RMD system takes 47 seconds/epoch to train on a Pentium IV computer 3.2GHz and classifies 23,000 words/second (assuming entity mentions are already labeled).

6.3 Analysis

In this subsection we compare: (a) the behavior of the PADUM for RMD against other known learning algorithms, and (b) our architecture with other typical IE systems. All the experiments reported in this subsection were performed on the training corpus using five-fold cross validation.

6.3.1 Analysis of the PADUM for RMD

For a better understanding of PADUM's behavior we compare the PADUM with the regular averaged PA and SVM for the problem of RMD. Note that the averaged PA is a special case of the PADUM, where $\Gamma_{\pm 1} = 0$. We use libsvm[6] for the implementation of the SVM classifier. We configured libsvm with the following parameters: $C = 1.0$; $gamma = 1/k$, where $k = 18$ is the number of categories (i.e., relation subtypes) in the RMD data. We built several other SVM models with various values for the classification penalty costs but saw no improvement in the overall performance. All three algorithms use the same features. To isolate the relation extraction component we performed this experiment using gold entity keys and scored only the relation classification task using standard precision/recall/F_1 measures.

Table 6 summarizes the results of this experiment. For the Perceptron algorithms we report results after one and after five epochs. The table shows that the PADUM performs the best out of the three algorithms. After 5 epochs, the PADUM has a F_1 score approximately 3 points higher than SVM and 1 point higher than the PA. Moreover, the PADUM is the most balanced of the three algorithms. As expected, the PA is precision-biased (with precision 14% higher than recall), whereas SVM (in its default configuration) is recall-biased (with recall 13% higher than precision). On the other hand, the PADUM after 5 epochs has precision and recall scores that are only within 5% of each other. This is proof that the dynamic margin adjustment built in the PADUM is beneficial.

	Precision	Recall	F_1
PADUM, 1 epoch	65.71%	45.48%	53.75
PADUM, 5 epochs	62.96%	56.31%	**59.44**
Avg PA, 1 epoch	**67.94%**	40.28%	50.58
Avg PA, 5 epochs	66.64%	52.19%	58.53
SVM	50.62%	**63.72%**	56.42

Table 6: Comparison of PA, PADUM, and SVM for RMD using gold entity mentions.

	Precision	Recall	F_1
Recognition only	92.39%	87.60%	89.93
Recognition + Classification	77.81%	74.41%	76.07

Table 7: Analysis of EMD scores.

Another advantage of PADUM is its learning speed. For the RMD problem PADUM takes 47 seconds/epoch and usually 5 epochs are sufficient to converge. On the other hand SVM (using libsvm's C-SVC SVM type) takes over 15 hours to learn a RMD model under the same conditions.

6.3.2 Analysis of the Proposed Architecture

In this section we analyze our proposed IE architecture, where the ambiguities in entity and relation classification are left in the system and solved only at the end.

Table 7 and Figure 4 justify the need for ambiguity detection in entity classification. Table 7 shows that the F_1 scores of the EMD system when performing only recognition are significantly higher (approximately 14 F_1 points) than the score of recognition and classification. This indicates that the ma-

[6] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

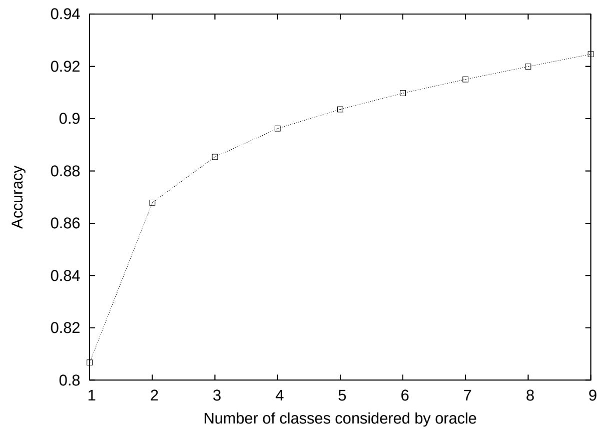

Figure 4: Accuracy of an oracle entity classification system that selects the best class out of the top N.

jor failure point for EMD is mention classification. Furthermore, Figure 4, which shows the accuracy of an oracle entity classification system that selects the best class out of the top N classes output by our entity re-classifier, indicates that the classification accuracy is significantly improved when considering the top two or three classes. For example, the oracle has an accuracy of approximately 89% with the top 3 classes (out of 45 categories: 44 subtypes plus one for the NIL category). This analysis indicates that, although mention classification is the weakest point in EMD, working with the top two or three categories improves significantly the quality of the EMD output. This motivated us in designing the IE system shown in Figure 2.

Table 8 compares our architecture with two other popular IE architectures: (a) the pipeline approach, where the EMD and RMD components are sequentially linked and only the top output is propagated between the layers, hence no inference is needed, and (b) the approach of (Roth and Yih, 2004), which combines the EMD and RMD outputs using an inference approach close to ours, but the two components are independently trained without any communication between them. We called our implementation of the latter "pseudo Roth & Yih" because we used the approximated inference described in Section 2 rather than the exact inference introduced in the original article. Table 8 shows that it is important to feed entity information to RMD: the "pseudo Roth & Yih" system, which trains the RMD component without entity class information, has a cost-based value score that is 3.7% lower than our system. On the other hand, the difference between our architecture and the pipeline approach is not that large: our cost-based value score is 0.1% larger for EMD and 0.2% larger for RMD. Nevertheless, our architecture has the additional advantage that all the solutions generated are consistent with the ACE domain constraints, which can not be guaranteed for the pipeline approach.

7 Conclusions

This paper describes a system for the extraction of mentions of entities and binary relations. The main focus behind the development of this system was robustness and simplicity: the system is completely machine learning-based and all learning tasks are developed using variants of the Perceptron algorithm. Furthermore, we use only syntactic information that can be efficiently extracted from text (newswire, blogs, etc): POS tagging and partial syntactic analysis (i.e., chunking).

The paper's contributions include several novel ideas. First, we define a new large-margin Perceptron Algorithm with Dynamic Uneven Margins (PADUM), which is capable of dynamically adjusting its margins in relation to the generalization performance of the learned model. Furthermore, the PADUM manages different margins for positive and negative examples to address the sample unbalance that is common in many learning problems. We show that for the task of relation extraction the PADUM performs significantly better than SVM even though its training time is two orders of magnitude smaller than SVM's.

Second, we propose a novel strategy to mitigate the propagation of errors made in early processing stages; e.g., entity classification. If ambiguities are detected, i.e, the corresponding component is not confident enough in its output, we let several hypotheses flow through the system. All ambiguities are solved only at the end using a simple approximated inference approach. We provided empirical evidence that our approach is better than other traditional IE architectures. Furthermore, our system guarantees a solution that is consistent with the domain constraints.

We evaluated our system within the ACE 2007

	Count				Cost (%)						
	Ent	Detection		Rec	Detection		Rec	Value	Value-based		
EMD	Tot	FA	Miss	Err	FA	Miss	Err	(%)	Pre	Rec	F
This paper	54824	2907	5805	16394	5.2	9.0	6.7	79.1	87.6	84.3	85.9
Pipeline	54824	2907	5805	16406	5.2	9.0	6.7	79.0	87.6	84.2	85.9
Pseudo Roth & Yih	54824	2907	5805	16400	5.2	9.0	6.7	79.1	87.6	84.3	85.9

	Count				Cost (%)						
	Ent	Detection		Rec	Detection		Rec	Value	Value-based		
RMD	Tot	FA	Miss	Err	FA	Miss	Err	(%)	Pre	Rec	F
This paper	8738	1661	4289	3681	12.1	48.7	4.4	34.8	74.0	46.9	57.4
Pipeline	8738	1933	4077	3868	14.0	46.6	4.8	34.6	72.1	48.6	58.1
Pseudo Roth & Yih	8738	1310	4865	3244	9.3	55.9	3.7	31.1	75.6	40.4	52.7

Table 8: Comparison of our architecture with the pipeline approach and (Roth and Yih, 2004).

evaluation. We obtain competitive results on both the EMD and RMD tasks, which is very encouraging considering the simplicity of the proposed approach.

Acknowledgements

This work was partially funded by the European Union project CHIL (IP-506808) and the Spanish Ministry of Science and Technology project TIN2006-15265-C06-05. Mihai Surdeanu is a research fellow within the Ramón y Cajal program of the latter institution.

References

Y. Altun, T. Hofmann, and M. Johnson. 2003. Discriminative learning for label sequence. In *Proceedings of NIPS 2003*.

C. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

T. Brants. 2002. A statistical part-of-speech tagger. In *Proceedings of ANLP 2002*.

M. Ciaramita and Y. Altun. 2006. Broad coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

G. Claudio, A. Lavelli, and L. Romano. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *Proc. of the European Chapter of the Association for Computational Linguistics (EACL)*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with the perceptron algorithms. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

M. Cristianini and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Y. Freund and R.E. Shapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37.

W. Krauth and M. Mezard. 1987. Learning algorithm with optimal stability in neural networks. *Journal of Physics*, 20.

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

Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola. 2002. The perceptron algorithm with uneven margins. In *Proc. of the 19th International Conf. on Machine Learning*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of ICML 2000*.

F. Rosemblatt. 1858. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 68:386–407.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL 2003*.

G. Zhou, J. Su, J. Zhang, and M. Zhang. 2005. Exploring various knowledge for relation extraction. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.