Bidirectional Sequence Classification for Named Entities Recognition

Andrea Gesmundo

Univ Geneva, Dept of Computer Science and Dept of Linguistic,
route de Drize 7, 1227 Carouge, Switzerland
andrea.gesmundo@unige.ch

Abstract. With this paper is presented a system for Named Entities Recognition, based on the Perceptron Algorithm. In the proposed framework, the order of the inference is not forced into a monotonic behavior (left-to-right), but is learned together with the parameters of the local classifier. The system tested on the task of Italian NER at EVALITA 2009 obtained the second position, with an F1 measure of 81.46%.

Key words: Guided Learning, Perceptron, NER.

1 Introduction

Named Entities Recognition is a subtask of Information Extraction. The goal of this task is to locate and classify chunks of words that identify names of persons, organizations, geo-political entities or locations. NER is defined as a chunking task, but following the general guidelines of [1] we can solve it as a tagging task. The system described in this paper carries out NER as a tagging task applying a semi-supervised training approach. In particular we extend the Guided Learning (GL) framework presented in [2]. This approach is more complex than supervised learning. The system can learn the parameters for the local classifier from gold standard labels, but has no indications on the order of inference.

Compared to others approaches, GL shows some advantages, it does not suffer from the label bias problem [3]. Basing the learning algorithm on the Perceptron scheme allows one to keep a low system complexity and moderate execution time, without sacrificing learning capability and quality of the results. Compared to others systems that use a Perceptron algorithm, like [4], GL introduces a bidirectional search strategy. Instead of forcing the order of the tagging in a left-to-right fashion, any tagging order is allowed. It follows a easiest-first approach and incorporates the learning of the order of inference in the training phase. In this way right-context and bidirectional-context features can be used at little extra cost. Following [5] we apply a voting system between multiple data representation of text chunks, this technique consists of training one instance of the system for each version of the same corpus represented with different chunks representations, and then merging the predictions obtained from the systems with a voting method.

As shown by the results obtained in Evalita 2009 NER Shared Tasks and in [2] [7] and [8], GL is a framework that can be adapted to a variety of tagging tasks, ensuring state of the art results and short training times.

2 Bidirectional Guided Classification

In this section we present the Inference Algorithm and the Training Algorithm.

2.1 Inference Algorithm

As input to the Inference Algorithm we have a sequence of tokens $t_1 t_2 \cdots t_n$. For each token t_i, we have to assign a label $l_i \in L$, with L being the label set. A subsequence $t_i \cdots t_j$ is called a span, and is denoted $[i, j]$. To each span s are associated one or more hypotheses, composed by a sequence of length $|s|$ over L.

The labels located at the boundaries of an hypothesis sequence are used as context for labeling tokens outside the span s. In our case a trigram model is used, so when choosing the label for the token t_i we can use the two boundary labels (l_{i+1}, l_{i+2}) of the right span $[i+1, j]$ if this has already been tagged. Similarly, we can use the two labels (l_{i-2}, l_{i-1}) as left context for the current tagging operation in the case that the left span $[k, i-1]$ is available. We will refer to the left two label as the left interface I_{left}, and to the right two labels as the right interface I_{right}.

We denote the boundaries of a span s with $b = (I_{left}, I_{right})$, b contains the labels relevant for the tagging of neighboring tokens. We partition the hypotheses associated with span s into sets compatible with the same boundaries b. For each span s we use a table M_s indexed by all possible b, so that $M_s(b)$ is the set of all hypotheses associated with s that are compatible with I_{left} and I_{right}.

For a span s, we denote the associated top hypothesis with:

$$h_s^* = \underset{h \in M_s(b), \forall b: M_s(b) \neq \emptyset}{\operatorname{argmax}} V(h) \tag{1}$$

where V is the score function of a hypothesis.

Spans are started and grown by means of tagging actions. Three kinds of actions are available: it is possible to start a new span by labeling a token with no context, or expand an existing span by labeling an adjacent token, or merging two spans by labeling the token between them. In this last case the two originating spans would be subsequences of the resulting span, and the labeling action of the token between the spans will use both right and left context information.

For each hypothesis h associated with a span s, we maintain its most recent tagging action $a(h)$, and the hypotheses, if any, that have been used as left context $h_L^*(h)$ and right context $h_R^*(h)$.

We can now define the score function for hypotheses in a recursive fashion:

$$V(h) = V(h_L^*(h)) + V(h_R^*(h)) + U(a(h)) \tag{2}$$

The score of the current tagging action $U(a(h))$ is added to the score of the top hypotheses that might have served as context and have been merged in the new hypothesis. The score of the labeling action $U(a(h))$ is computed through a linear combination of the weight vector w and the feature vector of the action $f(a(h))$:

$$U(a(h)) = w \cdot f(a(h)) \tag{3}$$

To reduce the search space explored during the inference algorithm we apply a beam search strategy. The beam width B determines the maximum number of boundaries b maintained for each span s. The value of B is given as input, as the weight vector w used to compute the score of an action.

The algorithm works using two groups of spans: P is the list of accepted spans, and Q is the a queue of candidate spans. Q can contain new spans of length one or extension of spans previously accepted and at the current time located in P.

At the beginning of the inference algorithm P is initialized with the empty set, and Q is filled with candidate spans $[i, i]$ for each token t_i, and for each label $l \in L$ assigned to t_i we set:

$$M_{[i,i]}((l, l)) = \{i \rightarrow l\} \tag{4}$$

where $i \rightarrow l$ represent the hypothesis consisting of the action with no context which assigns label l to w_i. This provides the set of starting hypotheses.

The loop of the algorithm repeatedly selects a candidate span s' from Q, s' is the span with the highest action score, so we pick the span that represents the next tagging action we are most confident about.

Now we use s' to update P and Q. First we update P, adding s' and removing the spans included in s'. Then let S be the set of spans removed from P. We update Q removing each span which takes one of the spans in P as context, and replace it with a new candidate span taking s' as new context.

The algorithm terminates when P contains a single span covering the whole token sequence and Q becomes empty.

The loop is guaranteed to terminate since at each iteration a span is expanded or added in P, and considering that P cannot have overlapping spans we can conclude that the number of iterations needed is linear with the size of the token sequence.

2.2 Learning Algorithm

In this section we describe the Guided Learning Algorithm, used to learn the weight vector w with a Perceptron-like algorithm.

For the training a set of token sequences $\{T_1, T_2, \cdots, T_m\}$ is provided as input. to each token sequence $T_r = (t_1, t_2, \cdots t_n)$ is paired a gold standard label sequence of the same length $L_r = (l_1, l_2, \cdots, l_n)$. At the beginning of the learning algorithm we initialize P and Q as we do in the inference algorithm. Then we iterate selecting the candidate span s' for the next labeling action from Q like

in the inference algorithm. If the s' top hypothesis match on the gold standard, we update P and Q as in the inference algorithm. Otherwise, we update the weight vector w by promoting the features of the gold standard, and demoting the features of the action of the candidate top hypothesis, like in the Perceptron algorithm. Then we undo the last labeling action by replacing the elements in Q with a new list of candidates containing all the possible spans based on the context spans in P, and computing the new scores with the updated weight vector w.

In our implementation we have used the Averaged Perceptron [4] and Perceptron with margin [6].

3 Experiments

In this section we are going to describe the setting chosen for the final experiment for the Evalita-09 NER shared task, we also report and discuss the results.

3.1 Setting

In the setting of our best system we set beam width $B = 3$, as threshold between speed and accuracy. As external resources we used gazetteers of names of cities and geographical locations (11000), generic proper names and surnames plus names of italian politicians and famous persons (49000), organizations and banks names, italian political parties (14000). Among the set of features used, we distinguish between context features and lexical features.

Context features are meant to capture the information of the surrounding words, POS and NER labels, We report the feature templates used in our best model in Table 1 .

Table 1. Templates for context features: 1) single word features, 2) couple of words features, 3) left context features, 4) right context features, 5) bidirectional features, 6) bidirectional features using POS tags.

1	$[w_0]$, $[w_{-2}]$, $[w_{-1}]$, $[w_1]$, $[w_2]$
2	$[w_{-1},w_0]$, $[w_0,w_1]$
3	$[n_{-1}]$, $[n_{-2},n_{-1}]$, $[n_{-2},n_{-1},w_0]$, $[n_{-1},w_0]$, $[n_{-2}]$, $[n_{-2},w_0]$
4	$[n_1]$, $[n_1,n_2]$, $[n_1,n_2,w_0]$, $[n_1,w_0]$, $[n_2]$, $[n_2,w_0]$
5	$[n_{-1},n_1]$, $[n_{-1},n_1,w_0]$
6	$[p_0, p_{-1}]$, $[n_{-2}, n_1, p_{-1}]$, $[n_{-2}, n_1, p_{-2}, p_{-1}, p_1]$

To select features that use the POS information we applied a semi-automatic method for feature template selection. These features are reported in Table 1 line 6. Adding these features led to a relative F_1 improvement of 1.1% on the development set.

As lexical features for the current word we consider: the presence of special characters of symbols like digits or '-'; the prefixes and suffixes up to length of 9 characters; the capitalization of the first letter or of the whole word, in relation with the capitalization of context words. We observed that treating the capitalization of the first word of the sentence separately from the other words of the sentence resulted in a relative F_1 improvement of 1.2% on the development set.

Following [5] we applied a voting system between multiple data representation of text chunks. We generated 5 versions of the corpus, one for each chunks representation (IOB1, IOB2, IOE1, IOE2, O+C). Then we trained one instance of our system on each of the five version of the corpus. As a final step we produced a prediction of the test set from each of the five versions of the system, and we merged the predictions with a majority vote. Differently from [5] we associated the votes to chunks instead of the single labels. In the merged prediction we kept the chunks with at least four votes out of five, using this technique we recorded a relative F_1 improvement of 0.8% on the development set.

3.2 Results

The results reported in this section are those obtained on the Evalita 2009 I-CAB corpus, consisting of news stories taken from an Italian newspaper.

The participating systems are evaluated on the Precision, Recall and F_1 measures. We submitted 2 outputs, the first for the system with no external resources, and the second with the use of gazetteers. The results are reported in Table 2, and Table 3 for the system with gazetteers.

Table 2. Results for the system with no external resources.

	Precision	Recall	F_1
GPE	82.37%	76.03%	79.07
LOC	68.29%	35.90%	47.06
ORG	78.13%	55.16%	64.67
PER	87.85%	76.96%	82.04
Overall	83.92%	69.79%	76.21

Table 3. Results for the system with the gazetteers.

	Precision	Recall	F_1
GPE	83.00%	83.73%	83.36
LOC	68.48%	40.38%	50.81
ORG	80.41%	63.69%	71.08
PER	91.03%	84.06%	87.41
Overall	86.06%	77.33%	81.46

We can notice that the Recall is usually lower than Precision. When gazetteers were used we observed relative F_1 improvement of 6.89%, moreover the ratio Recall/Precision increased from 0.83% up to 0.90%. We observed also that the identification for the classes GPE and PER reached a good level, while the LOC had the lower score in both the versions. On a common Desktop (equipped with a Core2 Duo CPU at 2.66GHz) the 8 rounds of training were completed in 1 hour and a half, and the tagging of the 4136 sentences of the test set took less then 2 minutes. During the training phase 500k features were generated.

4 Conclusion

In this paper we extended the work on the Guided Learning approach, adapting it to a new task, and applying new features. We successfully participated at the Evalita 2009 NER shared task, achieving the second position in the final rank. In related works, described in [2] [7] and [8], we applied this approach to a variety of tasks (POS tagging, NER, NP chunking) reaching state of the art results with moderate execution time. With this work, we have further proved the validity of Guided Learning.

Acknowledgments. We thank Giorgio Satta and Libin Shen for help and advice.

References

1. Ramshaw, L., and Marcus, M.: Text chunking using transformation-based learning. In: Proceedings of the third ACL workshop on very large corpora (1995)
2. Shen, L., Satta, G., Joshi, A., K.: Guided learning for bidirectional sequence classification. In: Proceedings of the 45th annual meeting of the association of computational linguistics (2007)
3. Bottou, L.: Une approche théorique de l'apprentissage connexionniste: Applications à la reconnaissance de la parole. Ph.D thesis, Université de Paris XI (1991)
4. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of EMNLP-2002 (2002)
5. Shen, H., Sarkar, A.: Voting between multiple data representations for text chunking. In: Proceedings of the eighteenth meeting of the canadian society for computational intelligence (2005)
6. Krauth, W., and Mézard, M.: Learning algorithms with optimal stability in neural networks. Journal of Physics A, vol. 20, pp. 745–752 (1987)
7. Gesmundo, A.: Elaborazione del linguaggio naturale basata su features bidirezionali. Master thesis, Università di Padova (2007)
8. Gesmundo, A.: Bidirectional sequence classification for Part of speech tagging. In: Proceedings of EVALITA 2009 (2009)