Semi-Supervised Neural Networks for Nested Named Entity Recognition *

Jinseok Nam
Knowledge Engineering Group
Department of Computer Science, TU Darmstadt, Germany
Knowledge Discovery in Scientific Literature
German Institute for Educational Research, Germany
nam@cs.tu-darmstadt.de

Abstract

In this paper, we investigate a semi-supervised learning approach based on neural networks for nested named entity recognition on the GermEval 2014 dataset. The dataset consists of triples of a word, a named entity associated with that word in the first-level and one in the second-level. Additionally, the tag distribution is highly skewed, that is, the number of occurrences of certain types of tags is too small. Hence, we present a unified neural network architecture to deal with named entities in both levels simultaneously and to improve generalization performance on the classes that have a small number of labelled examples.

1 Introduction

Named Entity Recognition (NER) is an important natural language processing (NLP) task that aims at assigning a class label to a word such as person, location, organization and so on. In contrast to the traditional NER where a classifier assigns only a single named entity (NE) for elements in text, the GermEval 2014 dataset (Benikova et al., 2014b) allows for elements to have two NEs at most. For example, "TU Darmstadt" is not only considered as an *organization*, but "Darmstadt" can be also tagged as a *location*. The dataset consists of sentences sampled from Leipzig Corpora Collection (LCC) (Quasthoff et al., 2006) publicly available for download.[1]

[1] http://corpora.uni-leipzig.de/download.html

Recently, neural networks (NNs) have succeeded in various NLP tasks including NER (Collobert et al., 2011). Thus, we build a neural network architecture solving the nested NER problem in a semi-superivsed way by making use of a large number of unlabelled sentences from LCC.

2 Background

2.1 NER using Neural Networks

Collobert et al. (2011) proposed a unified neural network architecture, namely SENNA, on which we build an architecture for nested NER.

Consider a sentence $t = \{w_1, w_2, \cdots, w_{N_t}\}$ of length N_t in which each word w_i is associated with its target y_i, which has one of C possible tags. The inputs to SENNA are the concatenated vector representations for the words in the sentence. The vector representations can be drawn from a matrix $\mathbf{L} \in \mathbb{R}^{d \times |V|}$ where d is the dimension of the vectors and $|V|$ is the number of words in our vocabulary. While it is possible to define another feature matrix that we want to learn such as capitalization features $\mathbf{L}^{(caps)}$ as well as the word features $\mathbf{L}^{(w)}$, for simplicity, we only consider the word features as $\mathbf{L}$ in this Section.

Assuming that we wish to tag a word w_i and let k_w be the width of a window. The vector representations of word w_i and of words surrounding w_i in a window are drawn from $\mathbf{L}$, then concatenated to form $\mathbf{x}_i = \{\mathbf{L}_{\cdot w_n}\}_{n=-\lfloor k_w/2 \rfloor + i}^{\lfloor k_w/2 \rfloor + i} \in \mathbb{R}^{d \cdot k_w}$ where $\lfloor x \rfloor$ denotes the largest integer not greater than x. If n is less than 1 or greater than N_t, a special *padding* word is used instead. In turn, the input $\mathbf{x}_i$ is passed to a non-linear function to obtain a hidden representation

$$\mathbf{h}_i = f\left(\mathbf{W}^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}\right) \quad (1)$$

where the function $f : \mathbb{R} \rightarrow \mathbb{R}$ is an element-wise transfer function, e.g., *sigmoid*, *tanh*, and *ReLU*,

$\mathbf{W}^{(1)} \in \mathbb{R}^{F \times (d \cdot k_w)}$ is a matrix of weights that link input units to hidden units, and $\mathbf{b}^{(1)} \in \mathbb{R}^F$ is a vector of biases for the hidden layer. The hidden representation $\mathbf{h}_i$ is, then, fed forward to the output layer to yield the prediction scores $\hat{\mathbf{y}}_i \in \mathbb{R}^C$ of the tags for the given local context

$$\hat{\mathbf{y}}_i = \mathbf{W}^{(2)}\mathbf{h}_i + \mathbf{b}^{(2)} \tag{2}$$

where $\mathbf{W}^{(2)} \in \mathbb{R}^{C \times F}$ are the weights between hidden and output units, each of which corresponds to a tag, and $\mathbf{b}^{(2)} \in \mathbb{R}^C$ are the biases for the output layer.

If we assume that the tag of each word depends only on that word and its context, i.e., $\mathbf{x}_i$, then the probability distribution over $\{w_i, y_i\}$ can be formulated as follows

$$p\left(y_1, \cdots, y_{N_t}, w_1, \cdots, w_{N_t}\right) = \prod_{i=1}^{N_t} p\left(y_i | \mathbf{x}_i; \Theta\right). \tag{3}$$

In order to convert the prediction scores $\hat{y}_{ji}$ of the tag j for the word w_i into probability, we can use the *softmax* function

$$p\left(y_{ji} = 1 | \mathbf{x}_i; \Theta\right) = \frac{\exp \hat{y}_{ji}}{\sum_k \exp \hat{y}_{ki}} \tag{4}$$

where $\Theta = \{\mathbf{L}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$ is a set of parameters. By taking the *log*, our objective, Eq. 4, becomes

$$\max_{\Theta} \sum_{i=1}^{N_t} \sum_{j=1}^{C} \mathbb{I}\left[y_{ji} = 1\right] \left(\hat{y}_{ji} - \log \sum_{k=1}^{C} \exp \hat{y}_{ki} \right) \tag{5}$$

where $\mathbb{I}[\cdot]$ denotes the indicator function which takes 1 when the argument is true, otherwise 0. This is referred to as *word-level* log-likelihood.

Learning Tag Dependencies In *word-level* log-likelihood, tag dependencies are ignored by the assumption that a tag is determined by only its local context. To exploit dependencies between tags, we take tag transition scores $\mathbf{T} \in \mathbb{R}^{C \times C}$ into account. A prediction score for the whole sentence is given by

$$\hat{y}_{[\mathbf{c}]} = \sum_{i=1}^{N_t} \mathbf{W}^{(2)}\mathbf{h}_i + \mathbf{b}^{(2)} + T_{c_i, c_{i-1}} \tag{6}$$

where $[\mathbf{c}]$ denotes a sequence of the tags in the sentence, c_i indicates the tag of the word w_i, and $T_{c_i, c_{i-1}}$ is a transition score from c_{i-1} to c_i. For the case $i = 1$, we also need initial tag scores $T_{c,0} \in \mathbb{R}^C$. The prediction score for the sentence is also transformed to a probability divided by the scores over all possible tag sequences $[\mathbf{k}]$

$$p\left(\{y_i\}_{i=1}^{N_t} | \{\mathbf{x}_i\}_{i=1}^{N_t}; \Theta, \mathbf{T}\right) = \frac{\exp \hat{y}_{[\mathbf{c}]}}{\sum_{[\mathbf{k}]} \exp \hat{y}_{[\mathbf{k}]}}. \tag{7}$$

Similarly, the objective taking transitions between tags into consideration is given by

$$\max_{\Theta, \mathbf{T}} \hat{y}_{[\mathbf{c}]} - \log \sum_{[\mathbf{k}]} \exp \hat{y}_{[\mathbf{k}]} \tag{8}$$

which is referred to as *sentence-level* log-likelihood and this can be addressed efficiently using *recursion*.

2.2 Semi-Supervised Learning

The simplest algorithm for semi-supervised learning is self-training (Rosenberg et al., 2005). In self-training, once a model is trained on labelled data, it is used to predict labels of unlabelled data, then such unlabelled data are provided as if additional labelled examples.

Pseudo-Label (PL) (Lee, 2013) is a semi-supervised learning technique especially for NNs. Unlike self-training, it estimates pseudo-labels, most probable labels of unlabelled data, during training and uses them to update parameters as well as labelled examples. Its purpose is similar to Entropy Regularization (Grandvalet and Bengio, 2005) that minimizes conditional entropy of unlabelled data as a measure of class overlap on the feature space.

3 Semi-Supervised Neural Networks for Nested NER

In contrast to the traditional NER, a word in nested NER can be tagged by multiple NEs. For simplicity, the number of levels is limited to two.

3.1 Jointly Learning Top-level and Nested NEs

In nested NER, a sentence t can be characterized by a sequence of triples $\{w_i, y_i^1, y_i^2\}$ where y_i^1 is the tag of the word w_i in the first level, and y_i^2 for the second level. Note that the tags in both levels are defined over the same set. Figure 1 describes our proposed architecture to tackle nested NER.

The proposed model deals with all NEs in both levels jointly during the learning phase by using an additional feature matrix $\mathbf{L}^{(ne)} \in \mathbb{R}^{d_{ne} \times C}$ for NEs, which is also a set of learnable parameters like $\mathbf{L}^{(w)}$. Each column of $\mathbf{L}^{(ne)}$ corre-

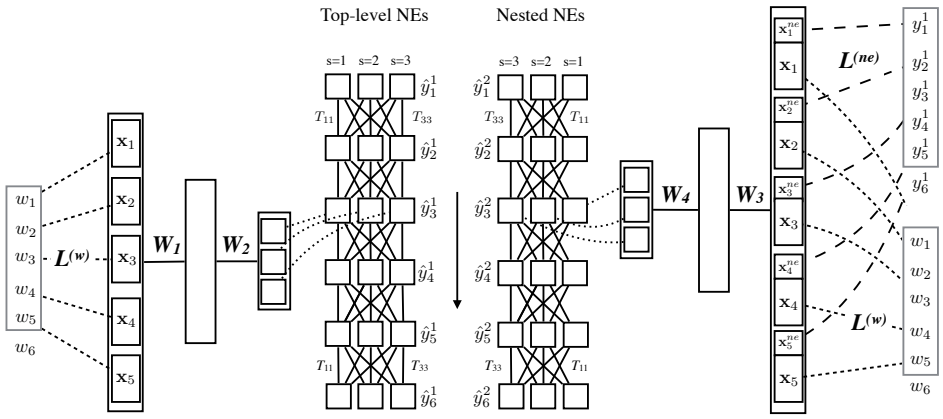

Figure 1: An illustrative example of the proposed architecture for jointly learning top-level and nested NEs. Consider a sentence $t = \{w_i, y_i^1, y_i^2\}_{i=1}^6$ of length 6, a window of size $k_w = 5$, and that we want to predict tags y_3^1, y_3^2 for a word w_3. Assuming that the number of NEs in the problem is 3, s indicates an index of a named entity. A matrix of word embeddings $\mathbf{L}^{(w)}$ and the tag transition matrix $\mathbf{T}$ are shared between two networks. Each network is trained to make predictions NEs of given a word sequence $\{w_i\}_{i=1}^6$ for each level.

sponds to a vector representation of a named entity. Given the concatenated feature vectors of a word $\mathbf{x}_i$ in the window as described in Section 2.1, we construct a vector representation for NEs in the top-level corresponding to that word, denoted by $\mathbf{x}_i^{ne} \in R^{d_{ne} \cdot k_w}$, then concatenate it to $\mathbf{x}_i$, which yields *combined* vector representations of words and NEs $\mathbf{x}_i^{comb} = \{\mathbf{x}_i, \mathbf{x}_i^{ne}\} \in R^{(d_{ne}+d_K)\times k_w}$. Similar to Eq.7 for the first level NEs, the *sentence-level* log-likelihood is also computed for the second level NEs like $\hat{\mathbf{y}}_i^2 = \mathbf{W}^{(4)} f\left(\mathbf{W}^{(3)}\mathbf{x}_i^{comb} + \mathbf{b}^{(3)}\right) + \mathbf{b}^{(4)}$. Then, the training objective considering the first- and second-level NEs simultaneously is given by

$$
\begin{aligned}
& p\left(\{y_i^1, y_i^2, w_i\}_{i=1}^{N_t}; \bar{\Theta}\right) = \\
& (1-\alpha)\; p\left(\{y_i^1\}_{i=1}^{N_t} | \{\mathbf{x}_i\}_{i=1}^{N_t}; \Theta, \mathbf{T}\right) \\
& + \alpha\, p\left(\{y_i^2\}_{i=1}^{N_t} | \{\mathbf{x}_i^{comb}\}_{i=1}^{N_t}; \theta, \mathbf{T}\right)
\end{aligned} \tag{9}
$$

where $\theta = \{\mathbf{W}^{(3)}, \mathbf{b}^{(3)}, \mathbf{W}^{(4)}, \mathbf{b}^{(4)}, \mathbf{L}^{(\cdot)}\}$, $\bar{\Theta} = \{\Theta, \theta, \mathbf{T}\}$, and $\alpha \in [0, 1]$ is a control parameter.

3.2 Learning from Pseudo Labels of Unlabelled Data

Semi-supervised learning methods are well-suited to the problems where the number of training instances is insufficient. Tag distribution of the GermEval dataset is highly skewed. In other words, the proportion of the three tag types, i.e., LOC, PER, and ORG, amounts to approximately 70% (See (Benikova et al., 2014b) for statistics).

In this work, we apply PL to only the first level in order to improve the generalization performance on such small classes. The first term of the right hand side in Eq. 9 can be re-written as

$$
\begin{aligned}
& (1-\alpha)\; p\left(\{y_i^1\}_{i=1}^{N_t} | \{\mathbf{x}_i\}_{i=1}^{N_t}; \Theta, \mathbf{T}\right) \\
& + (1-\alpha)\, \beta_t\, p\left(\{\hat{y}_{ui}^1\}_{ui=1}^{uN_t} | \{\mathbf{x}_{ui}\}_{ui=1}^{uN_t}; \Theta, \mathbf{T}\right)
\end{aligned} \tag{10}
$$

where *ui* is an index of an unlabelled sentence randomly selected from LCC, $\hat{y}_{ui}^1$ is a pseudo tag for the word w_{ui} in an un-annotated sentence, and β controls the importance of learning from unlabelled data. Scheduling the control parameter at time t takes the following form:

$$
\beta_t = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\beta_{\max} & T_1 \le t \le T_2 \\ \beta_{\max} & t > T_2 \end{cases} \tag{11}
$$

with $\beta_{\max}$ = 2, $T_1 = 100$, and $T_2 = 500$. [2] The pseudo label $\hat{y}_{ui}^1$ is determined by simply choosing the most confident one given prediction scores for an un-annotated sentence during training.

4 Experiments

Our experiments were performed on the GermEval 2014 dataset, where the tags constitutes four major types, i.e., LOC, PER, ORG and OTH, and their sub-types which end with "-deriv" or "-part" using a BIO tagging scheme. The results in

[2] The hyperparameters for scheduling PL were chosen via cross validation.

Table 1: Effect of word embeddings

Initialization	P	R	F_1
Random	69.67	54.19	60.96
Pretrain	68.39	69.27	68.82

Table 2: Effect of Pseudo Label as a regularizer

Learning scheme	P	R	F_1
Sup. learning ($\beta_t = 0$)	68.39	69.27	68.82
Semi-sup. ($\beta_{\max} = 2$)	77.08	68.40	72.48

Table 1 and 2 are reported in terms of the official metric, namely M1 (See (Benikova et al., 2014a)), in the GermEval 2014 Shared Task.

4.1 Details of Training

We evaluated the proposed method with the following hyperparameter settings over the number of hidden units $F = 300$, the dimension of capitalization features $d_{cap} = 3$, the dimension of named entity features $d_{ne} = 10$, window size $k_w = 5$, $\alpha = 0.5$, a fixed learning rate 0.01 for SGD with AdaGrad (Duchi et al., 2011). In addition, we used length normalization over all embeddings such that $\|x\| = 10$ to prevent overfitting. For the transfer function in Eq.1, *ReLU*, $f(x) = \max(0, x)$, is used. The feature matrices $\mathbf{L}^{(caps)}$ and $\mathbf{L}^{(ne)}$ were initialized randomly.

4.2 Importance of Word Embeddings

We used *word2vec* (Mikolov et al., 2013) for learning word embeddings because of its efficiency.[3] We set the dimension of word embeddings d_w to 128 and the size of vocabulary $|V|$ is about 4M which yields the feature matrix $\mathbf{L}^{(w)} \in \mathbb{R}^{128\times 4M}$. We run the *word2vec* for 10 epochs with a fixed learning rate 0.01 on approximately 87M sentences from a German Wikipedia dump, LCC, and SDeWac (Faaß and Eckart, 2013).

The results of using pretrained word embeddings on unlabelled data in comparison to random initialization are shown in Table 1. We observed that NNs using pretrained word embeddings perform much better in terms of *recall*.

4.3 Effect of Semi-Supervised Learning

We evaluated our proposed approach for nested NER. The results of this experiment are shown in Table 2. The semi-supervised approach outperforms the purely supervised one. We observe that learning with pseudo labels reduces the number of false positives which results in higher precision. In particular, the number of predictions in the top-level resulting from the supervised approach is 2738 while the semi-supervised approach yields 2378 predictions. Interestingly, we also observe performance improvement on LOC and ORG as well as the smaller classes including OTH and "deriv"- and "part"-classes, but not all of them.

Table 3: The System Performance on Unseen Data

Metrics	P	R	F_1
M1	76.76	66.16	71.06
M2	78.09	67.31	72.30
M3 (1^{st} level)	77.93	68.52	72.92
M3 (2^{nd} level)	57.86	37.86	45.77

4.4 Results of GermEval 2014 Shared Task

The proposed method was submitted to the GermEval 2014 Named Entity Recognition Shared Task. Our system called **PLsNER** was ranked at 5^{th} and the scores are shown in Table 3. More results and comparisons with other systems can be found in (Benikova et al., 2014a).

5 Conclusions

We proposed a neural network architecture, which is capable of learning from top-level NEs and nested NEs jointly in nested NER. By making use of unlabelled data in a semi-supervised fashion, we also demonstrated its effectiveness when a small number of training examples are provided.

Our experiments show that the use of word embeddings improves *recall* compared to random initialization. Pseudo labels make it possible to get more *precise* predictions. Additionally, our system performs pretty well on unseen data without use of language-dependent feature engineering steps.

Acknowledgments

This work has been supported by the German Institute for Educational Research (DIPF) under the Knowledge Discovery in Scientific Literature (KDSL) program.

[3] https://code.google.com/p/word2vec/

References

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Pado. 2014a. GermEval 2014 Named Entity Recognition: Companion Paper. In *Proceedings of the KONVENS GermEval Shared Task on Named Entity Recognition*, Hildesheim, Germany.

Darina Benikova, Chris Biemann, and Marc Reznicek. 2014b. NoSta-D Named Entity Annotation for German: Guidelines and Dataset. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Gertrud Faaß and Kerstin Eckart. 2013. SdeWaC–a Corpus of Parsable Sentences from the Web. In *Language processing and knowledge in the Web*, pages 61–68. Springer.

Yves Grandvalet and Yoshua Bengio. 2005. Semi-supervised Learning by Entropy Minimization. In *Advances in Neural Information Processing Systems 17*, pages 529–536.

Dong-Hyun Lee. 2013. Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In *Workshop on Challenges in Representation Learning, ICML*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Uwe Quasthoff, Matthias Richter, and Chris Biemann. 2006. Corpus Portal for Search in Monolingual Corpora. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 1799–1802, Genoa.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-Supervised Self-Training of Object Detection Models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision*, WACV-MOTION '05, pages 29–36.