Transfer Learning for Industrial Applications of Named Entity Recognition

Lingzhen Chen[1], Alessandro Moschitti[1]
Giuseppe Castellucci[2], Andrea Favalli[2], Raniero Romagnoli[2]

[1] University of Trento, POVO, Trento 38123, Italy
{lingzhen.chen, alessandro.moschitti}@unitn.it
[2] Almawave Srl., Italy
{g.castellucci, a.favalli, r.romagnoli}@almawave.it

Abstract. In this paper, we propose a Transfer Learning technique for Named Entity Recognition that is able to flexibly deal with domain changes. The proposed technique is able to manage both the case when the set of named entities does not change and the case when the set of named entities changes in the target domain. In particular, we focus on the case when the target data contains only the annotation of a target named entity, and the source data is no longer available for the target task. Our solution consists in transferring the parameters from a source model, which are then fine-tuned with the target data. The model architecture is modified when recognizing a new category by adding properly new neurons to the model. Our experiments show that it is possible to effectively transfer learned parameters in both the scenarios, resulting in strong performances over the target categories without degrading the performances on the other named entities.

Keywords: Transfer Learning · Named Entity Recognition · Sequence Labeling · Recurrent Neural Network

1 Introduction

Standard Named Entity Recognition (NER) models are supposed to be trained and applied on data coming from similar sources (i.e., in-domain data). The application of a NER model on out-of-domain data will inevitably result in poor performances [5]. For example, the contexts in which a Person name entity occur could be slightly different (from a morpho-syntatic point-of-view) in each domain. Moreover, in industrial scenarios, different domains (e.g., different customers) often will require to recognize disjoint named entity sets. For example, in finance, target entities would probably include Companies and Banks, while in politics, target entities could include Senators and Ministries. Thus, general-purpose models, for example trained starting from general-purpose datasets (e.g., wikipedia) cannot be an optimal solution. A typical solution to the out-of-domain problem is to train a NER model from scratch over different annotated datasets, i.e., one for each domain. In this way, each NER model can focus over the morpho-syntactic variations of the texts characterizing the occurrences of the different categories for each domain. However, this solution has two main disadvantages: 1) annotating a dataset from scratch for each domain can be costly; 2) there is no reuse of any acquired knowledge over the existing data (e.g., common entity categories could share some linguistic characteristics that can be useful for training a new model).

Motivated by the problems described above, in this work, we define a new paradigm of *Transfer Learning* for NER models. Without loss of generality, our setting consists of two steps: (i) an initial step, where a source model $\mathcal{M}_{\mathrm{S}}$ is trained to recognize a set of named entities on source data $\mathcal{D}_{\mathrm{S}}$; (ii) a subsequent step, where the target model $\mathcal{M}_{\mathrm{T}}$ needs to recognize the new NE categories in the target data $\mathcal{D}_{\mathrm{T}}$, which are unseen in $\mathcal{D}_{\mathrm{S}}$. Notice that $\mathcal{D}_{\mathrm{T}}$ is typically much smaller in size compared to $\mathcal{D}_{\mathrm{S}}$ and it will contain only the annotations of the target entities. Moreover, in the update step $\mathcal{D}_{\mathrm{S}}$ is no longer available. These are strong requirements often appearing in industrial scenarios, where a NER model provider wants to make available model update strategies to its customers, without distributing the original data used to train the source model.

These problems lie in the research area of Transfer Learning (TL), which aims to leverage the gained knowledge on a task in a source domain, to improve the learning of a task in a target domain [17]. To tackle the problem, we aim at applying a knowledge transfer in a deep neural network setting. Given an initial model for NER trained on source data, we modify the layers of the network to include new neurons for learning the new categories and continue to train with $\mathcal{D}_{\mathrm{T}}$. More specifically, we implement a Bidirectional LSTM (BiLSTM) with Conditional Random Fields (CRF) as $\mathcal{M}_{\mathrm{S}}$. In the update step, we modify such architecture to build $\mathcal{M}_{\mathrm{T}}$. The weights from $\mathcal{M}_{\mathrm{S}}$ are transferred to $\mathcal{M}_{\mathrm{T}}$ and then fine-tuned with $\mathcal{D}_{\mathrm{T}}$, where both seen and unseen NE categories could be observed.

We conduct extensive experiments for updating NER models, analyzing the performance of both methods on a custom dataset. Our experiments show that the training and fine-tuning phases can effectively learn new knowledge about existing or new entities, while not degrading much the performances over unchanged entities.

The rest of the paper is structured as following: in sections 2 and 3 the adopted Deep Neural Network and the transfer learning methodology are described. In section 4 the experimental evaluation is provided. In section 5 the related work are discussed. Finally, in section 6 the conclusions are derived.

2 Deep Neural Network for Named Entity Recognition

The NER problem can be defined as a sequential labeling problem: given an input sequence $X = x_1, x_2, ..., x_n$ $(x_i \in \mathcal{X})$, predicts the output sequence $Y = y_1, y_2, ..., y_n$ $(y_i \in \mathcal{Y})$. $\mathcal{X}$ and $\mathcal{Y}$ represent the input and output space respectively. Typically, the model learns to maximize the conditional probability $P(Y|X)$.

The model adopted to label a token sequence with respect to the NE categories is the state-of-the-art neural model made of a Bidirectional LSTM + CRF [15]. This model consists of several layers: (i) a character-level Bidirectional LSTM (BiLSTM) to compute a vector representation of each word based on the character composing it; (ii) a token-level LSTM layer whose role is to compute the representation of a sequence; (iii) a Conditional Random Field (CRF) layer for making sequential predictions. A detailed description of the model is presented in remaining of this section.

Word & Character Representations. A word in the input sequence is represented by both its word-level and character-level embeddings. The former is used to capture semantic information of words, while the latter is adopted to model the morphological variations of words (e.g., capitalization, inflections, etc.) and they showed to provide useful information for the NER task [15]. Token-level embeddings map each word t_i (given a lookup table) to a vector $\boldsymbol{w}_i$. Character-level embeddings map, instead, each character c_{ij} of the i^{th} word (given a lookup table) to a vector $\boldsymbol{e}_{ij}$. The final representation for a word given its character embeddings is obtained by applying a BiLSTM (see next section) over the $\boldsymbol{e}_{ij}$ vectors to obtain the character-level representation $\boldsymbol{e}_i$. The final representation of the i^{th} word x_i in the input sequence is the concatenation of its word-level embedding $\boldsymbol{w}_i$ and character-level embedding $\boldsymbol{e}_i$.
Bidirectional LSTM Layer. BiLSTM is a neural architecture where two LSTM cells are applied in both the directions of a sequence. In particular, it is composed of a forward LSTM ($\overrightarrow{\text{LSTM}}$) and a backward LSTM ($\overleftarrow{\text{LSTM}}$), which read the input sequence (represented as word vectors described in the previous subsection) in both left-to-right and reverse order.

The output of the BiLSTM $\boldsymbol{h}_t$ is obtained by the concatenation of forward and backward outputs: $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{h}_t}]$, where

$$\overrightarrow{\boldsymbol{h}_t} = \overrightarrow{\text{LSTM}}(\boldsymbol{x}_t, \overrightarrow{\boldsymbol{h}}_{t-1}) \tag{1}$$

and

$$\overleftarrow{\boldsymbol{h}_t} = \overleftarrow{\text{LSTM}}(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}_{t+1}) \tag{2}$$

and $\boldsymbol{h}_t$ captures the left and right context for $\boldsymbol{x}_t$.
Prediction Layer. The representation computed by the BiLSTM is then passed through a fully-connected layer that computes the representation p_t for each word. The final prediction $\boldsymbol{y}_t$ is made based on the softmax over the output of fully-connected layer $\boldsymbol{p}_t$ by

$$\text{P}(y_t = c|p_t) = \frac{e^{W_{o,c}p_t}}{\sum_{c' \in C} e^{W_{o,c'}p_t}}, \tag{3}$$

where W_o are parameters to be learned on the output layer and C represents the set of all the possible output labels.
CRF Layer. CRF [14] is a powerful model for sequence labeling tasks such as NER as it considers neighboring information of the categories when making predictions. We implement a Linear Chain CRF over the output of the BiLSTM to improve the prediction ability of the model, i.e., by taking the neighboring prediction into account while classifying each time step. Recall that we aim at maximizing the conditional probability P$(Y|X)$ of the output label sequence Y on the input word sequence X. In the CRF setting, P$(Y|X)$ is computed by

$$\text{P}(Y|X) = \frac{e^{score(X,Y)}}{\sum_{Y'} e^{score(X,Y')}} \tag{4}$$

where Y' is all possible label sequences and $score(\cdot)$ is calculated by adding up the transition and emission scores for a label sequence. More specifically, the emission score is the probability of predicting a label y_t for t word in the sequence, and is given by the prediction made from the BiLSTM. The transition score is the probability of transiting from previously predicted label y_{t-1} to current label y_t. The outputs of fully-connected layer $\boldsymbol{p}_t$ at time step t provides the emission scores for all possible value of y. A square matrix **P** of size $C + 2$ is used to store transitional probabilities among C output labels, as well as a *start* label and an *end* label. Hence,

$$score(X, Y) = \sum_t \boldsymbol{p}_t[y_t] + \mathbf{P}_{y_t, y_{t-1}} \tag{5}$$

3 Updating Algorithm for NER Neural Models

The NER architecture described in the previous section is usually adopted to train a model in a domain that is able to recognize named entities within texts of the same domain. Our goal is to make this architecture adaptable to different domains without the need of a full re-training of the model. Recall, in fact, that we are assuming that the dataset used to train a base model, could be not available in the update step.

Let us consider data coming from a source domain D_{S} containing named entities annotations for the entities $s \in S$ and data coming from a target domain D_{T}. The target data contains annotations of entities $t \in T$, where $T = S \cup O$ and O is a possibly empty set of new entity categories.

In the initial step, the model is trained on D_{S} until the optimal parameters $\hat{\theta}^{\mathrm{S}} = \hat{\theta}^{\mathrm{S}}_{-o} \cup \hat{\theta}^{\mathrm{S}}_{o}$ are obtained and saved. $\hat{\theta}^{\mathrm{S}}_{-o}$ contains the parameters of the network except those of the output layer, while $\hat{\theta}^{\mathrm{S}}_{o}$ contains the parameters only of the output layer. There are two different TL cases, depending on the nature of the T set. First is the case where $T = S \cup O$ and O is empty, i.e., the set of entities does not change between source and target domains. In this setting, what changes is only the domain of the data where the entities of the set T need to be recognized. In this case, the adaptation of the neural model can be achieved by updating the weights of the network by performing a training stage over D_{T}. In this way, the weights of the network will shift towards the new data D_{T}, and the network training will accommodate within them the relevant morpho-syntactic information of the new domain.

The second scenario occurs when $T = S \cup O$ and O is a non-empty set. In this case, the model structure needs to be changed in order to accommodate the new categories. In order to make the model able to recognize the new NE categories contained in O, the major modification is made on the output layer after the BiLSTM. In the standard BiLSTM architecture, the fully-connected layer maps the outputs $\boldsymbol{h}$ of BiLSTM to a vector $\boldsymbol{p}$ of size $nC + 1$, where n is a factor depending on the tagging format of the dataset. For example, $n = 2$ if the data is annotated with an IOB format: in fact, a word can be **O**utside a NE or can be at the **B**eginning or **I**nside an annotation. For recognizing target NEs, we build the same model architecture for all layers before the output layer and initialize all the parameters by $\hat{\theta}^{\mathrm{S}}_{-o}$ obtained in the initial step. We extend the output layer by size nE, where E is the number of new NE categories. The extended part is initialized with weights drawn from the random distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, where μ and σ are the mean and standard deviation of the pre-trained weights in the same layer.

In this way, the associated weight matrix of the fully-connected layer $\mathbf{W}_o^s$ also updates from the original shape $nC \times p$ to a new matrix $\mathbf{W}_o^{s'}$ of shape $(nC + nE + 1) \times p$. Note that the parameters $\hat{\theta}^{\mathrm{S}}_{output}$ and $\hat{\theta}^{\mathrm{S}}_{-output}$ are essentially the weights in the matrix $\mathbf{W}_o^s$ and $\mathbf{W}_{-o}^s$. In the BiLSTM + CRF architecture, the modification must be made also on the **P** transition matrix in order to include the new rows and columns that are relative to the new entity categories. The new model parameters are updated in the same way as in the initial step by performing training steps over D_{T}.

4 Experiments

We divide the experiments on dynamically updating the NER model into three sub-tasks (i), (ii) and (iii). For task (i), we experiment on updating $\mathcal{M}_{\mathrm{T}}$ with regard to a target NE category with more training data coming from a new domain (i.e. $\mathcal{D}_{\mathrm{T}}$). For task (ii), we experiment on forgetting and retraining an entity class (seen in $\mathcal{D}_{\mathrm{S}}$) with new training data (i.e. $\mathcal{D}_{\mathrm{T}}$) from a different domain. Note that the difference between task (i) and task (ii) is the morphology of the target NE in $\mathcal{D}_{\mathrm{T}}$. For task (i), the morphology of the target NE is expected to be the same in $\mathcal{D}_{\mathrm{S}}$ and $\mathcal{D}_{\mathrm{T}}$ while they are expected to be different for task (ii) (hence the *forgetting* and *retraining*). Lastly, for task (iii), we experiment on updating $\mathcal{M}_{\mathrm{T}}$ on a target NE that is unseen in $\mathcal{D}_{\mathrm{S}}$, with new training data $\mathcal{D}_{\mathrm{T}}$.

4.1 Data

The experiments are carried out with a dataset provided by Almawave. The dataset consists of sentences coming from chat conversations in Italian between users and customer service support representatives. The dataset contains conversations coming from two different customers/domains. Each sentence is annotated with respect to 8 named entity categories. The specification of each of the named entity category can be found in Table 1. The annotation process was provided by Almawave in the context of the Italian educational project *Alternanza Scuola-Lavoro*. In this project, two groups of students coming from a scientific and a language high school, annotated the dataset by Webanno annotation tool [22] in two weeks (one week for each group): each group was split into 9 teams, the annotations of each team were checked against the other. Finally, an annotation review process was performed by Almawave's employees.

The dataset is split in three parts, reflecting the three sub-tasks. In Table 2, the details about the dataset are provided. The target NE category for three subtasks are `PaymentInformation`, `CustomerIdentification` and `Address`, respectively. Note that in $\mathcal{D}_{\mathrm{S}}$, all NEs are annotated but in $\mathcal{D}_{\mathrm{T}}$, only the annotations of the target NE with regard to the subtask is provided. That is to say, there are linguistic representations of other NEs in $\mathcal{D}_{\mathrm{T}}$ but they are annotated as "O". We first use pretrained $\mathcal{M}_{\mathrm{S}}$ to annotate these NEs before we train the $\mathcal{M}_{\mathrm{T}}$ on $\mathcal{D}_{\mathrm{T}}$. There is no specific preprocessing of the dataset, except for replacing all digits with 0 to reduce the size of the vocabulary.

Entity	**Description**	**Example**
Address	The occurrence of a postal address in a sentence.	*L'indirizzo sul contratto **Via di Casal Boccone 188**.* *(The address on the contract is **Via di Casal Boccone 188**.)*
CustomerIdentification	The occurrence of information of the customer, e.g., client id, username.	*codice cliente **P10AA210645*** *(client id **P10AA210645**)*
Location	The occurrence of a location.	*La mia residenza a **Roma**, ma il contratto registrato a **Latina**.* *(My residency is in **Rome**, ma the contract is registered in **Latina**.)*
Organization	The occurrence of the name of an organization.	*Ho ricevuto un offerta migliore da **Fastweb**.* *(I got a better offer from **Fastweb**)*
PaymentInformation	The occurrence of information regarding a payment, e.g., a credit card number or an IBAN.	*Si, certo pu fare l'accredito sul conto **IT17X 000011111000001234 000**.* *(Sure, you can credit on **IT17X 000011111000001234 000**.)*
Person	The occurrence of information regarding the name of a person.	*La linea registrata a nome di **Mario Rossi**.* *(The line is registered to **Mario Rossi**)*
PersonalInformation	The occurrence of information regarding personal information, e.g. date of birth.	*Nato a Veroli provincia di Roma il **23/05/1956**.* *(I am born in Veroli province of Rome on **23/05/1956**)*
Phone	The occurrence of information regarding phone numbers.	*Mi pu contattare al **333447755** o al **00667788*** *(You can reach me at **333447755** or at **00667788**)*

Table 1: Named Entity Categories and Examples.

	Task (i)	Task (ii)	Task (iii)
$\mathcal{D}_{\mathrm{S}}$	12377 (213)	12377 (947)	12377 (0)
$\mathcal{D}_{\mathrm{T}}$	7045 (143)	6979 (758)	4389 (3828)
$\mathcal{D}_{\mathrm{test}}$	1496 (16)	1496 (16)	3488 (484)

Table 2: Number of total NEs contained in each dataset. The numbers in parentheses are those of the target NE categories. The target NE categories for task (i), (ii), (iii) are `PaymentInformation`, `CustomerIdentification` and `Address` respectively

4.2 Experimental Setup

We use 300 dimension GLOVE pretrained embedding[3] for Italian to initialize the weights of the embedding layer. Since we do not lowercase the tokens in the input sequence, we map the words having no direct mapping to the pretrained word embeddings to their lowercased counterpart, if one is found in the pretrained word embeddings.

We map the infrequent words (words that appear in the dataset for less than twice) to $<$*UNK*$>$ as well as the unknown words appearing in the test set. The word embedding for $<$*UNK*$>$ is drawn from a uniform distribution between $[-0.25, 0.25]$. The character embedding lookup table is randomly initialized with embedding size of 25. The hidden size of the character-level BLSTM is 25 while the word level one is 128.

We apply a dropout regularization on the word embeddings with a rate of 0.5. All models are implemented in TensorFlow [1], as an extension of NeuroNER [6]. In the fist step, we use the Adam [13] optimizer with a learning rate of 0.05, gradient clipping of 5.0 to minimize the categorical cross entropy, and a maximum epoch number of 100 at each step. The neural hyperparameters of the update step are the same as the ones of the first step except for the learning rate. In fact, we observed that it is a better choice to lower the value of the learning rate: it is needed to avoid catastrophic forgetting phenomena. In our experiments for the update step, we set the value of learning rate to

[3] http://hlt.isti.cnr.it/wordembeddings/

NE type	# of training examples	# of delta training examples	# of test examples	test F1 (Init. step)	test F1 (Sub. step)
Address	0	0	172	00.00	00.00
CustomerIdentification	947	515	131	80.70	79.64 (-1.06)
Location	1932	931	16	12.31	12.24 (-0.07)
Organization	2662	1450	383	69.76	69.75 (-0.01)
PaymentInformation	213	143	16	85.71	**86.67 (+0.96)**
Person	4042	3599	736	91.34	93.28 (+1.94)
PersonalInfornation	613	78	4	26.67	36.36 (+9.69)
Phone	1968	329	38	59.84	56.49 (-3.35)
All	12377	7045	1496	72.50	73.31

Table 3: Test F1 on each NE categories in the initial and the subsequent step for task (i).

0.005. Finally, the models are evaluated with the F1 score [4] as in the official CONLL 2003 shared task [21].

4.3 Results

In this section, we present results on the three experiments.

Fine-tuning seen NE with Additional Data from a different domain First of all, we show the result in Table 3 of task (i), i.e. training $\mathcal{M}_{\mathrm{T}}$ with new training data on target NE.

In terms of the test F1 score in the initial step, we see that model performs well on categories that have a substantial number of instances in $\mathcal{D}_{\mathrm{S}}$, such as `Organization`, `Person` and `Phone`. One unusual case being `Location`, which has 1932 training instances but only got 12.31 in test F1 in the initial step. This is possibly due to the fact that there are only 16 instances in the test set, and most of them are not commonly seen in the training set. For example, surface form *Manfredonia*, *Aprilia Latina*, *Acilia Roma* or *Veroli* are never presented in the training set, which makes it quite difficult for the model to correctly distinguish them from non-NE words, especially in a different context. Surface form *Trento* is presented many time in the training set but annotated as `O`, which also causes confusion for the model.

`Phone` is a NE category with plenty of training instances but the F1 score is not as high as expected. After a comparison between training and test data, we found out that the low performance could be due to the different format of the phone number. In the training set, phone number are often presented as a token of several digits in the format *000000000* where each 0 means a digit, whilst in the test set, they are presented as several separate tokens with shorter numbers of digits, instead of one token.

With regard to fine-tuning the dataset on $\mathcal{D}_{\mathrm{T}}$, the F1 of the target NE category `PaymentInformation` increased by 1 point with the new training data in $\mathcal{D}_{\mathrm{T}}$. For

[4] https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt

NE type	# of training examples	# of delta training examples	# of test examples	test F1 (step1)	test F1 (step2)
Address	0	0	172	0.00	0.00
CustomerIdentification	947	758	131	78.76	**89.73 (+10.97)**
Location	1932	867	16	10.47	10.81 (+0.34)
Organization	2662	1222	383	63.06	64.24 (+1.18)
PaymentInformation	213	23	16	60.87	60.87 (+0.00)
Person	4042	3813	736	93.10	94.72 (+1.62)
PersonalInformation	613	15	4	33.33	33.33 (+0.00)
Phone	1968	281	38	55.07	69.72 (+14.65)
All	12377	6979	1496	71.10	73.53 (+4.43)

Table 4: Test F1 on each NE categories in the initial and the subsequent step for task (ii).

some of the NE categories, the test F1 sees a very minor amount of decrease, while for some others, there are improvement in the test F1. This result suggests the effectiveness of the TL techniques on transferring the learned knowledge in $\mathcal{D}_{\text{S}}$ to $\mathcal{M}_{\text{T}}$. It is also able to further improve on both the target NE category and several other categories without catastrophic forgetting.

Retraining seen NE with additional data from a different domain In Table 4, we present the results of task (ii), where the target model $\mathcal{M}_{\text{T}}$ is trained on $\mathcal{D}_{\text{T}}$ with the same set of NE as $\mathcal{D}_{\text{S}}$ but in a different domain. In terms of the test F1 in the initial step, the result is quite similar to that of task (i). It is reasonable because the training procedure for the source model in these two tasks are the same. As in the subsequent step for task (ii), we see a consistent improvement on the test F1 for all NE categories. It indicates that $\mathcal{M}_{\text{T}}$ is able to adapt fairly well to the new domain for NEs, with the parameters transferred from $\mathcal{M}_{\text{S}}$.

More importantly, since the $\mathcal{D}_{\text{test}}$ in task (ii) is from the same domain as $\mathcal{D}_{\text{T}}$ for task (ii), task (ii) simulates the industrial scenario where a company have at hand a pretrained model and some in-domain training for their target task. The good performance of the proposed TL techniques suggests that the model performance on the target task can be improved with a small amount of target data, compared to using only the pretrained model.

Recognizing new NE with new training data Last but not the least, we present in Table 5 the results of task (iii) on recognizing NE that is unseen in $\mathcal{D}_{\text{S}}$. As seen from the result, in the subsequent step, we got a fairly good performance on recognizing the target new NE along with improvements in test F1 scores on other seen NE categories. It is worth noting that, in $\mathcal{D}_{\text{T}}$, there are not many training examples of those seen NE categories. (for example, only 36 for `PaymentInformation`. The test F1 score only sees a minor decrease in two categories `CustomerIdentification` and `Phone`.

NE type	# of training examples	# of delta training examples	# of test examples	test F1 (step1)	test F1 (step2)
Address	0	3828	172	00.00	**79.55 (+79.55)**
CustomerIdentification	947	23	243	88.05	87.75 (-0.30)
Location	1932	174	288	62.86	67.56 (+4.70)
Organization	2662	258	828	79.22	82.81 (+3.59)
PaymentInformation	213	36	51	84.44	85.56 (+1.12)
Person	4042	289	1223	93.61	94.73 (+1.12)
PersonalInformation	613	58	28	51.02	52.08 (+1.06)
Phone	1968	137	343	88.42	88.10 (-0.32)
All	12377	4389	3488	85.10	85.50 (+0.40)

Table 5: Test F1 on each NE categories in the initial and the subsequent step for task (iii).

Hence, the TL techniques are able to transfer well the learned knowledge in the source model $\mathcal{M}_{\mathrm{S}}$ to the target model $\mathcal{M}_{\mathrm{T}}$.

5 Related Work

Our work is related to research in Named Entity Recognition and Transfer Learning. We report on both in the following sections.

5.1 Named Entity Recognition

In the earlier years of the study on Named Entity Recognition, most work approached the task by engineering linguistic features [3, 2], such as lexical features, case information, orthographically patterns and so on. Models such as Maximum Entropy Model, Perceptrons and CRF are often used for the task [9, 3, 7, 10].

Recent approaches also include neural models. Current state-of-the-art methods on NER are mostly Recurrent Neural Network models that incorporate word and character level embeddings and/or additional morphological features. [11] uses BiLSTM combined with CRF to established the state-of-the-art performance on NER (90.10 in terms of test F1 on CONLL 2003 NER dataset). In their system, they used a variety of handcraft features together with word embeddings as the input to the model. Later, [15] implemented the same CRF over BiLSTM model without using any handcraft features. They reported 90.94 of test F1 on the same dataset. [4] also implemented a similar BiLSTM model with Convolutional filters as character feature extractor, achieving 91.62 in the F1 score using also lexical features.

In this work, we opted to adopt a BiLSTM + CRF in order to test whether our proposed methods can be applied on the state-of-the-art models.

5.2 Transfer Learning

Neural networks based TL has proven to be very effective for image recognition [8, 19]. In the NLP area, [16] showed that TL can also be successfully applied on semantically equivalent NLP tasks. Researches were carried out on NER related TL too. [18] explored TL for NER with different NE categories (different output spaces). They pretrain a linear-chain CRF on large amount annotated data in the source domain. A two linear layer neural network is used to learn the discrepancy between the source and target label distributions. Finally, they initialize another CRF with learned weight parameters in linear layers for the target domain. [12] experimented with transferring features and model parameters between similar domains, where the label types are different but may have semantic similarity. Their main approach is to construct label embeddings to automatically map the source and target label types to help improve the transfer. In [20], they have also used an adapter to help transfer. They propose progressive networks to solve a sequence of reinforcement learning task while being immune to the forgetting, by leveraging learned knowledge with the adapter, which is an additional connection between new model and learned models. This connection is realized by a feed-forward neural layer with non-linear activation.

6 Conclusion

In this paper, we study the Named Entity Recognition problem in a domain adaptation setting involving a two-step process. The first step regards the acquisition of a source model, while the second step regards the update of this model.

The main characteristics of our setting are: (i) the set of named entities between source and target domain can possibly change; (ii) the dataset used to acquire the source model is no longer available in the update step; (iii) the dataset used to train the target model is only annotated with target NE. These are strong requirements often appearing in industrial scenarios, where a NER model provider wants to make available model update strategies to its customers, without distributing the original data used to train the source model.

We verified that our method can be applied to current state-of-the-art neural models for Named Entity Recognition, i.e., a Bidirectional LSTM + CRF applied over word and character embeddings inputs. We carried out extensive experiments to analyze the effect on the performance of the transfer approach. The empirical results show the effectiveness of the proposed methods and techniques. In particular, the update step we studied demonstrated to be effective in recognizing a specific target entity, while not degrading the performances on the other categories.

In future work, we should investigate how to update a model with respect to more than one named entity. Moreover, we believe that this approach can be also applied to different and more complex tasks that we aim to investigate, e.g., coreference resolution.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia,

Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Carreras, X., Màrquez, L., Padró, L.: Learning a perceptron-based named entity chunker via online recognition feedback. In: Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003. pp. 156–159 (2003), http://aclweb.org/anthology/W/W03/W03-0422.pdf
3. Chieu, H.L., Ng, H.T.: Named entity recognition with a maximum entropy approach. In: Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003. pp. 160–163 (2003), http://aclweb.org/anthology/W/W03/W03-0423.pdf
4. Chiu, J., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics **4**, 357–370 (2016), https://www.aclweb.org/anthology/Q16-1026
5. Ciaramita, M., Altun, Y.: Named-entity recognition in novel domains with external lexical knowledge. In: Proceedings of the NIPS Workshop on Advances in Structured Learning for Text and Speech Processing. 2005. (2005)
6. Dernoncourt, F., Lee, J.Y., Szolovits, P.: Neuroner: an easy-to-use program for named-entity recognition based on neural networks. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017 - System Demonstrations. pp. 97–102 (2017), https://aclanthology.info/papers/D17-2017/d17-2017
7. Diesner, J., Carley, K.M.: Conditional random fields for entity extraction and ontological text coding. Computational & Mathematical Organization Theory **14**(3), 248–262 (2008). https://doi.org/10.1007/s10588-008-9029-z, https://doi.org/10.1007/s10588-008-9029-z
8. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. pp. 647–655 (2014), http://jmlr.org/proceedings/papers/v32/donahue14.html
9. Florian, R., Ittycheriah, A., Jing, H., Zhang, T.: Named entity recognition through classifier combination. In: Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003. pp. 168–171 (2003), http://aclweb.org/anthology/W/W03/W03-0425.pdf
10. He, Y., Kayaalp, M.: Biological entity recognition with conditional random fields. In: AMIA 2008, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 8-12, 2008 (2008), http://knowledge.amia.org/amia-55142-a2008a-1.625176/t-001-1.626020/f-001-1.626021/a-060-1.626384/a-061-1.626381
11. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR **abs/1508.01991** (2015), http://arxiv.org/abs/1508.01991
12. Kim, Y., Stratos, K., Sarikaya, R., Jeong, M.: New transfer learning techniques for disparate label sets. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015,

Beijing, China, Volume 1: Long Papers. pp. 473–482 (2015), http://aclweb.org/anthology/P/P15/P15-1046.pdf

13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014), http://arxiv.org/abs/1412.6980
14. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001. pp. 282–289 (2001)
15. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016. pp. 260–270 (2016), http://aclweb.org/anthology/N/N16/N16-1030.pdf
16. Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., Jin, Z.: How transferable are neural networks in NLP applications? In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016. pp. 479–489 (2016), http://aclweb.org/anthology/D/D16/D16-1046.pdf
17. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010). https://doi.org/10.1109/TKDE.2009.191, https://doi.org/10.1109/TKDE.2009.191
18. Qu, L., Ferraro, G., Zhou, L., Hou, W., Baldwin, T.: Named entity recognition for novel types by transfer learning. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016. pp. 899–905 (2016), http://aclweb.org/anthology/D/D16/D16-1087.pdf
19. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: An astounding baseline for recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014. pp. 512–519 (2014). https://doi.org/10.1109/CVPRW.2014.131, https://doi.org/10.1109/CVPRW.2014.131
20. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks. CoRR **abs/1606.04671** (2016), http://arxiv.org/abs/1606.04671
21. Sang, E.F.T.K., Meulder, F.D.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003. pp. 142–147 (2003), http://aclweb.org/anthology/W/W03/W03-0419.pdf
22. Yimam, S.M., Gurevych, I., de Castilho, R.E., Biemann, C.: Webanno: A flexible,web-based and visually supported system for distributed annotations. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (System Demonstrations) (ACL 2013). pp. 1–6. Association for Computational Linguistics (August 2013), http://tubiblio.ulb.tu-darmstadt.de/98019/