Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Learning to select pseudo labels: a semi-supervised method for named entity recognition*

Zhen-zhen LI, Da-wei FENG, Dong-sheng LI‡, Xi-cheng LU

College of Computer, National University of Defense Technology, Changsha 410073, China

E-mail: lizhenzhen14@nudt.edu.cn; davyfeng.c@gmail.com; dsli@nudt.edu.cn; xclu@nudt.edu.cn

Received Nov. 22, 2018; Revision accepted Apr. 17, 2019; Crosschecked Aug. 19, 2019; Published online Dec. 27, 2019

Abstract: Deep learning models have achieved state-of-the-art performance in named entity recognition (NER); the good performance, however, relies heavily on substantial amounts of labeled data. In some specific areas such as medical, financial, and military domains, labeled data is very scarce, while unlabeled data is readily available. Previous studies have used unlabeled data to enrich word representations, but a large amount of entity information in unlabeled data is neglected, which may be beneficial to the NER task. In this study, we propose a semi-supervised method for NER tasks, which learns to create high-quality labeled data by applying a pre-trained module to filter out erroneous pseudo labels. Pseudo labels are automatically generated for unlabeled data and used as if they were true labels. Our semi-supervised framework includes three steps: constructing an optimal single neural model for a specific NER task, learning a module that evaluates pseudo labels, and creating new labeled data and improving the NER model iteratively. Experimental results on two English NER tasks and one Chinese clinical NER task demonstrate that our method further improves the performance of the best single neural model. Even when we use only pre-trained static word embeddings and do not rely on any external knowledge, our method achieves comparable performance to those state-of-the-art models on the CoNLL-2003 and OntoNotes 5.0 English NER tasks.

Key words: Named entity recognition; Unlabeled data; Deep learning; Semi-supervised method
https://doi.org/10.1631/FITEE.1800743 **CLC number:** TP391.1

1 Introduction

Named entity recognition (NER) is an important foundation in many natural language processing (NLP) applications, including relation extraction and question answering. An NER task consists of detecting the entity boundaries and classifying the entities into predefined categories such as persons, organizations, and locations. Recently, NER tasks have been typically formulated as a sequence labeling problem, and deep learning methods have achieved state-of-the-art performance (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016). However, the performance of deep learning methods relies heavily on high-quality labeled data. In some specific language (such as in the Chinese clinical NER (CNER)) or domains (such as in the finance and military domains), labeled data is rather limited, while unlabeled data that contains substantial entity information is easily obtained. For example, electronic medical records contain abundant clinical entities, and financial news contains a large number of specific entities like companies or products.

To deal with the common situations where labeled data is limited but considerable unlabeled data is available, researchers try to apply semi-supervised methods for NER tasks. Liao and Veeramachaneni (2009) proposed a simple semi-supervised

‡ Corresponding author

* Project supported by the National Key Research and Development Program of China (No. 2016YFB0201305) and the National Natural Science Foundation of China (No. 61872376)

ORCID: Zhen-zhen LI, http://orcid.org/0000-0002-4116-5077; Dong-sheng LI, http://orcid.org/0000-0001-9743-2034

algorithm to improve NER tasks by automatically annotating unlabeled data as a new source of training data. They used a conditional random field (CRF) classifier to tag unlabeled data, and then corrected the low-confidence labels based on carefully designed rules. Then they improved the NER classifier by iterating between generating labeled data and retraining the classifier with the new training set. As a result, this semi-supervised method achieved significant improvements compared to supervised methods without using unlabeled data. However, this rule-based semi-supervised method can hardly be generalized to other entity types, and fails to be applied to deep learning models.

To further improve the performance of deep learning models for NER, researchers used unlabeled data to pre-train context-sensitive representations. Contextualized word embeddings derived from a bidirectional deep language model capture word semantics in the context to address the polysemous and context-dependent nature of words (Peters et al., 2017, 2018). Akbik et al. (2018) leveraged the internal states of a character language model to produce contextual string embeddings. Devlin et al. (2018) also learned contextual representations based on the deep transformer model architecture. These contextual word embeddings such as ELMo (Peters et al., 2018), flair embeddings (Akbik et al., 2018), and bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018) significantly improve the state-of-the-art results of almost all NLP tasks. They ensure the neural models take full advantage of the limited labeled data through the use of accurate and rich word representations. However, they fail to explore the abundant implicit entity information in the unlabeled data.

Recently, pseudo-labeling has been proposed to make use of unlabeled data to improve deep learning models and achieved new records in several image datasets (Lee, 2013; Wu and Prasad, 2018). It learns to assign labels to unlabeled data and uses these pseudo labels as if they were true labels. Lee (2013) created pseudo labels by picking up the class which has the maximum predicted probability, and Wu and Prasad (2018) proposed a clustering algorithm to assign cluster labels as the label for the unlabeled data. However, these pseudo labels contain quite a few obviously wrong labels, which can impair the performance of subsequent training procedures.

To address this challenge, in this study we propose a new semi-supervised framework to improve the performance of the single neural model by using additional unlabeled data for the NER task. We learn an extra module, called the Judge model, to select the most reliable pseudo labels for creating new labeled data from unlabeled data. With this pretrained Judge model, we improve the performance of the NER model by iterating between generating new labeled data and retraining the NER model with the all labeled data. Since our framework aims to improve the single neural NER model, we first summarize the general structure of neural NER models and propose some practices of building neural models with regard to different NER tasks. Second, we apply machine learning methods to train an effective Judge model for filtering out erroneous pseudo labels. At last, we propose a detailed strategy to create new labeled data with the learned Judge model and retrain the NER model with this enlarged training set.

Experimental results on two English NER tasks and a CNER task demonstrate that our semi-supervised framework can effectively improve the performance of neural NER models with unlabeled data.

2 Related work

Our work is inspired by two lines of research: deep learning for NER and semi-supervised methods to use unlabeled data.

Collobert et al. (2011) proposed a neural network architecture for the NER task without relying on any task-specific engineering or a lot of prior knowledge. Since then, researchers have paid more attention to using deep neural networks to deal with NER. Huang et al. (2015) used a bidirectional long short-term memory (BiLSTM) network to model both past and future input features, and a CRF layer to jointly decode the label sequence for the whole sentence. Chiu and Nichols (2016) used convolutional neural network (CNN) and Lample et al. (2016) used BiLSTM neural networks to extract character-level features, which further improved the performance of the BiLSTM-CRF model. Based on a similar architecture, Ma and Hovy (2016) proposed an end-to-end model without any feature engineering or data processing. This model has the

BiLSTM-CNN-CRF architecture, which employs CNNs to encode character-level information, BiLSTM to encode input sentences represented by the combination of character- and word-level representations, and CRF to decode the best label sequence. In other related work, recurrent neural networks (RNNs) have also been explored for decoding labels (Mesnil et al., 2013; Zhai et al., 2017). Shen et al. (2017) demonstrated that LSTM decoders perform better and faster than CRF decoders. In fact, we can apply our semi-supervised method to any best-performing neural NER model. In this study, we construct deep neural models based on the BiLSTM-CRF architecture, which is a popular choice for NER tasks.

Liao and Veeramachaneni (2009) proposed a simple semi-supervised method which exploits rule-based independent features to help assign high-precision labels to unlabeled data. These features are based on entity context and types, so it is difficult to extend the rules to other domains. Qi et al. (2009) proposed a word-class distribution learning method to incorporate additional features from self-labeled examples. Sun et al. (2017) proposed a modified RNN for NER, and used co-training to improve the RNN model and probability statistic models with unlabeled data when the training data is limited. The major difference between our approach and previous semi-supervised studies is that we learned a machine-learning based model to select self-labeled data. Instead of relying on heuristic rules or statistic features as complements, we aim to learn a general model based on the neural NER model for selecting reliable labels for unlabeled data.

The language modeling objective has proven to be effective in obtaining context-sensitive representation (Peters et al., 2017; Rei, 2017). Training bidirectional language models over large unlabeled corpora can help produce contextual embeddings for polysemous words depending on their context (Akbik et al., 2018; Peters et al., 2018), which is helpful for the NER task. Devlin et al. (2018) also proposed a deep transformer model architecture to learn contextual representations, which can capture word semantics in different contexts. In contrast to classic word embedding algorithms that give a word a fixed representation, such as Word2Vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014), these algorithms produce contextual representations, which set new records for almost all NLP tasks (Akbik et al., 2018; Devlin et al., 2018; Peters et al., 2018). These methods can make full use of limited labeled data by including accurate and rich word representations, but fail to explore abundant implicit entity information in the unlabeled data.

Recently, pseudo labeling has been used to improve the performance of deep learning models. Lee (2013) proposed a semi-supervised method to train labeled data and unlabeled data with pseudo labels simultaneously for deep neural networks. Wu and Prasad (2018) proposed a semi-supervised clustering algorithm, which includes pairwise must- and cannot-link constraints to cluster labeled data and unlabeled data, and the algorithm produced high-quality pseudo labels. The authors used all the training data together with the pseudo labels to pre-train deep neural networks and then fine-tuned the model with the limited labeled data. These methods put forward a new way to use unlabeled data in deep neural models and obtain promising performance for image classification tasks.

3 Approaches

3.1 Problem definition

In this subsection, we introduce our approach to create more labeled data to improve the performance of deep learning models for NER tasks.

Given a sentence, which is naturally regarded as a sequence of tokens, we adopt the BIOES tag scheme which stands for begin, inside, outside, end, and single, to annotate the entity type and relative position of each entity token (Chiu and Nichols, 2016). For example, the sentence "Bill Gates lives in Washington" is formatted (Table 1), and thus we can identify a person entity "Bill Gates" and a location entity "Washington." With this kind of tag scheme, NER can be addressed as a sequence labeling problem by assigning a predefined tag to each token of the sentence. The neural network based sequence labeling model achieves state-of-the-art performance on

Table 1 An example of a formatted sentence

Sentence	"Bill"	"Gates"	"lives"	"in"	"Washington"
Tag	B-PER	E-PER	O	O	S-LOC

B-PER: begin of person entity; E-PER: end of person entity; O: outside; S-LOC: a single location entity

public NER datasets (Lample et al., 2016; Ma and Hovy, 2016; Rei, 2017; Akbik et al., 2018), although its performance is severely limited due to the insufficient clean labeled data in practical applications.

In this study, we investigate a way to use unlabeled data in the same domain and try to automatically create high-quality labeled data to improve the performance of deep learning models for NER. Liao and Veeramachaneni (2009) designed textual rules to generate specific types of new training data, which may be accurate and non-redundant, but these rules are difficult to be extended to other domains. Recently, Lee (2013) and Wu and Prasad (2018) applied the pseudo-labeling technique to deep learning models. Lee (2013) proved that training with pseudo labels is in effect equivalent to entropy regularization (Grandvalet and Bengio, 2006), and can achieve better generalization performance. The pseudo labels they obtained are used as if they were true labels without any screening process. However, in lieu of that approach, we propose an extra Judge model to filter out noisy labels, and use the selected pseudo labels to train the neural NER model.

3.2 Overview

The main components in our semi-supervised learning method are illustrated in Fig. 1. The pre-trained component (in grey) is implemented in a supervised mode. With the limited labeled data, we train a baseline NER model and a Judge model to select high-quality pseudo labels. The semi-supervised component (in orange) shows how to use unlabeled data to improve the performance of the NER model with a pre-trained Judge model.

3.3 Basic named entity recognition models

Our baseline NER model is based on deep neural networks (DNNs), closely following a number of recent studies (Chiu and Nichols, 2016; Jagannatha and Yu, 2016; Ma and Hovy, 2016). To achieve the best performance with DNNs for different NER tasks, we break the NER model into four parts so as to flexibly modify some parts according to specific languages and domains. We identify this structure from many DNN architectures as input representation, contextual encoding layer, fully connected layer, and decoding layer.

1. Input representation

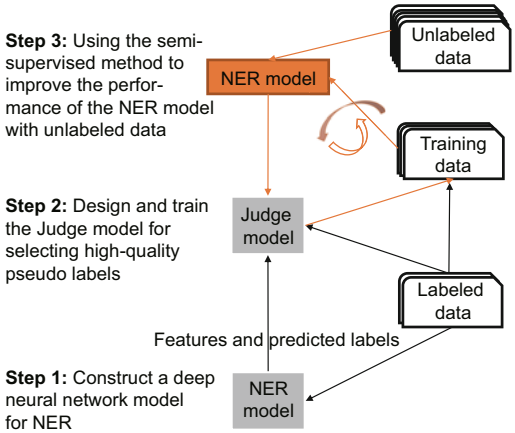

Fig. 1 Framework of our semi-supervised learning method (NER: named entity recognition)
References to color refer to the online version of this figure

Input representation varies according to the language involved. For example, in English, each word is usually regarded as a token and it is popular to concatenate character-level features and word embeddings as token representations. However, in Chinese, the phrase or word after word segmentation is usually regarded as a token.

For each token, we want to build a vector $\boldsymbol{x} \in \mathbb{R}^k$ that will capture the meaning and relevant features for the NER task. It has been proven effective to build this vector by concatenating a pre-trained word embedding $\boldsymbol{x}_{\text{word}} \in \mathbb{R}^{d_1}$ and a vector containing features extracted from the character level. One option is to use hand-crafted features such as stemming and spelling features, which can be implemented with predefined rules, e.g., using a component with "0" or "1" to represent whether the word starts with a capital letter. Another option is to use some kind of neural network to automatically extract character-level features from character embeddings, which can learn morphological information (like the prefix or suffix of a word).

Take the BiLSTM network as an example. We apply this character-level encoder to each token and obtain a fixed-size vector $\boldsymbol{x}_{\text{chars}} \in \mathbb{R}^{d_2}$. Then the input representation of each token is the concatenation of two parts $\boldsymbol{x} = [\boldsymbol{x}_{\text{word}}, \boldsymbol{x}_{\text{chars}}] \in \mathbb{R}^k$ with $k = d_1 + d_2$.

Alternatively, CNNs are used to extract character-level features (Chiu and Nichols, 2016; Zhai et al., 2017), whose computational cost is much lower than that of the LSTM module, but the

performance is slightly degraded (Shen et al., 2017).

2. Contextual encoding layer

The input tokens are mapped to a sequence of vectors after the input representation part, $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$. Then we use the BiLSTM layer to learn context-dependent representation for every token. The LSTM network (Hochreiter and Schmidhuber, 1997) is a popular variant of RNNs, capable of capturing long-distance dependencies and alleviating gradient vanishing problems. BiLSTM uses two independent LSTMs to process the sequence on opposite directions, which is helpful for learning past (left) and future (right) contexts. For each time step, the hidden state from the previous time step and the current input token vector are fed to the LSTM unit to learn a new hidden state. Then the two hidden states of each time step are concatenated to form the final output:

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

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

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t]. \tag{3}$$

After the BiLSTM layer, we have a sequence of vectors $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n)$, and $\boldsymbol{h}_i \in \mathbb{R}^{2m}$ $(i = 1, 2, \ldots, n)$ with m being the number of LSTM units. Note that $\boldsymbol{x}_t$ captures only information at the word level (syntax and semantics), while $\boldsymbol{h}_t$ takes context into account as well.

3. Fully connected layer

The output of the BiLSTM layer is passed through a fully connected neural network, projecting the vector representation of each token to the final output label space, where each entry corresponds to a score for each tag. Given that the final labels have s classes, the output of this layer $\boldsymbol{z} \in \mathbb{R}^{n \times s}$ is computed as: $\boldsymbol{z}_i = \boldsymbol{h}_i \cdot \boldsymbol{U} + \boldsymbol{b}$, where $\boldsymbol{U} \in \mathbb{R}^{2m \times s}$ and $\boldsymbol{b} \in \mathbb{R}^s$ are trainable parameters. We can reasonably interpret the j^{th} value of $\boldsymbol{z}_i \in \mathbb{R}^s$ as the score of class j for token i.

This layer is described as an independent part because we save its output matrix $\boldsymbol{z}$ as the input features for the Judge model.

4. Decoding layer

Based on the learned vector representations $\boldsymbol{z}$, we predict the final class label for each token. To learn the dependencies between successive labels, linear-chain CRF (Lafferty et al., 2001) is used to jointly decode the best chain of labels for a given sentence. The chain CRF computes the probability distribution over all of the sequences of labels and finds the sequence of labels with the highest probability. Take the label sequence $(y_1, y_2, \ldots, y_n)$ as an example:

$$P[y_1, y_2, \ldots, y_n | \boldsymbol{z}] \propto \exp\left(\sum_{i=1}^{n-1} A[y_i, y_{i+1}] + \sum_{i=1}^{n} \boldsymbol{z}_i\right), \tag{4}$$

where $\boldsymbol{A} \in \mathbb{R}^{s \times s}$ is a trainable parameter matrix and learns the dependencies between two successive labels. Dynamic programming is employed to compute Eq. (4), that is, using the forward-backward algorithm at training time and the Viterbi algorithm at test time (Collobert et al., 2011).

Chain CRF has been adopted to decode output label sequences by most DNN models for NER (Huang et al., 2015; Chiu and Nichols, 2016; Lample et al., 2016; Peters et al., 2017). Another option is RNNs (Zhai et al., 2017) and Shen et al. (2017) have demonstrated that RNNs can achieve a comparable performance to the chain CRF layer.

3.4 Judge model

In this subsection, we describe how to learn and use the Judge model. We use the Judge model to filter out the wrongly predicted labels for unlabeled data. Thus, we try to use supervised machine learning methods to train a binary classifier as the Judge model. Fig. 2 illustrates the process to train a Judge model. We first construct a transformed dataset for the Judge model with the pre-trained NER model and limited labeled data. Then we train and select the best Judge model with the transformed dataset.

1. Transformed data for the Judge model

To identify the wrongly predicted labels for a sequence of tokens, input features for each token should include as much relevant information as possible. Thus, we use the output of the fully connected layer $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n)$ together with the predicted labels $\boldsymbol{y}' = (y'_1, y'_2, \ldots, y'_n)$ to form input features. The Judge model needs to judge the predicted label for each token of the sentence, so we regard $\{(\boldsymbol{z}_i, y'_i), i = 1, 2, \ldots, n\}$ as an input sample and assign the corresponding label t_i as

$$t_i = \begin{cases} 0, & y'_i = y_i, \\ 1, & \text{otherwise}, \end{cases} \tag{5}$$

where y_i is the gold label, and if the predicted label

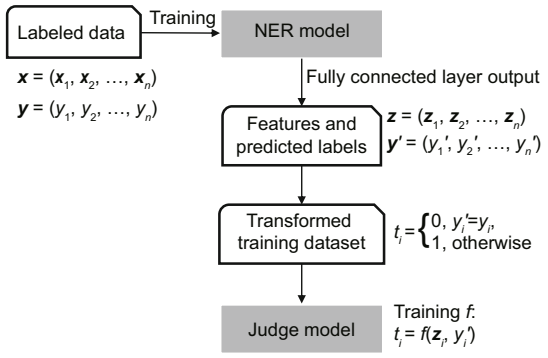

Fig. 2 Process for learning the Judge model
After the named entity recognition (NER) model is pre-trained with labeled data, we obtain the output vectors of its fully connected layer and predicted labels. Then we build a transformed dataset for the Judge model by assigning each token a label, so as to learn the supervised Judge model

is not equal to the gold label, its value is 1. In this way, we deal with all the labeled sentences and build a dataset, where each token from a sentence forms an independent sample. We split this dataset into two parts, the training set and test set, for training the supervised Judge model.

Since the precision of the NER model is pretty high (i.e., the majority of predicted labels are correct), we take the minority, the incorrectly predicted labels, as positive samples. In fact, the transformed data is fairly class-imbalanced, so hybrid sampling strategies are used to balance the training set when training a supervised Judge model.

2. Methods for training the Judge model

The aim of the Judge model is to identify as many incorrectly predicted labels as possible; that is, learn a binary classifier with function f: $t_i = f(\boldsymbol{z}_i, y_i')$ and a high recall rate is preferred.

The simplest and most efficient method is to calculate the confidence score of the predicted label by the softmax formula. Since vector $\boldsymbol{z}_i$ implies the score distribution among all label classes, the softmax-based method can effectively filter erroneously predicted labels by selecting an appropriate threshold. For any token $\boldsymbol{z}_i \in \mathbb{R}^s$ (s is the number of label classes), provided that the predicted label is $y_i' = j$ $(1 \le j \le s)$, its confidence value is computed as

$$v_i = \mathrm{e}^{z_{i,j}} \Big/ \sum_{k=1}^{s} \mathrm{e}^{z_{i,k}}. \tag{6}$$

In addition, we choose two representative machine learning algorithms to learn the class-imbalanced binary classifier: support vector machines (SVMs) (Cortes and Vapnik, 1995) and feed-forward neural network (FNN) (Schmidhuber, 2015). SVMs can find the boundary that separates classes by as wide a margin as possible, which is important in the situation where two classes cannot be clearly separated. As for the FNN, it can handle token representations learned from the BiLSTM encoder better, because they are both neural networks and the features are represented in a similar way.

Furthermore, the ensemble models that combine several classification results by voting are tested as candidate Judge models.

3. Pre-trained Judge model selection

By applying different training methods, we can obtain Judge models with different advantages. We can evaluate the performance of Judge models in two ways. One is to evaluate the Judge models directly using cross validation on the transformed data, and the other is to evaluate their performance in selecting pseudo labels by testing how much improvement the NER model achieved when applying our framework.

The preference for selecting a Judge model is also related to the size of the unlabeled data. For most cases, we assume that unlabeled data is easy to obtain and unlimited, so we prefer a Judge model that has a high recall in the transformed test set. In this way, we can filter out most of the incorrectly predicted pseudo labels. When only a certain amount of unlabeled data is provided, however, we may prefer a Judge model with a higher F_1-score; thus, there is a trade-off between recall and precision.

3.5 Semi-supervised learning component

In this subsection, we present the technical details of using unlabeled data and the pre-trained Judge model to improve the performance of the neural NER model.

1. Creating silver-standard data

After we have learned an NER model and a Judge model, we can use them to process unlabeled data to automatically create new trusted labeled data, which we call silver-standard data. The NER model is applied to tag unlabeled sentences with predicted labels, and the Judge model is used to provide estimated labels for the predicted labels for each token in the sentences. We assume that the positively estimated labels cover most of the wrongly predicted labels, and the remaining new labeled named entities are trusted.

To filter out named entities (NEs) with wrongly predicted labels and to preserve the valid context of the trusted NEs as completely as possible, we combine the contiguous clauses with trusted NEs as a complete sentence. We propose a strategy (Algorithm 1) for automatically creating silver-standard data. For each sentence, we first check if there exist noisy labels (lines 4–9). If there are noisy labels, we combine continuous trusted clauses (lines 10–15) and filter out meaningless new sentences at last (line 17).

2. Iteratively improving the performance of the NER model

If the unlabeled data is too large, we split it into several parts and deal with each part iteratively. Supposing that we have learned the NER model M_NER and Judge model M_Judge based on a small set of labeled training data L, the semi-supervised component (in orange) in Fig. 1 is further detailed in the following four steps. At first, the training data T contains only L.

Step 1: Tag $\boldsymbol{U}_i$ with model M_NER.

Step 2: Select pseudo labels with the model M_Judge and automatically create new labeled data D_i with the proposed strategy; update T by adding D_i to it.

Step 3: Retrain the NER model with T to obtain a temporary model Tmp_NER.

Step 4: Evaluate Tmp_NER on the test dataset. If there is no improvement or no more unlabeled data, the algorithm comes to an end and returns M_NER; if there is an improvement, M_NER is updated with Tmp_NER and returns to step 1.

In this way, we create new trustable labeled data D_i from unlabeled data and improve the performance of the NER model iteratively.

Algorithm 1 Strategy for automatically creating silver-standard sentences

Require: an unlabeled sentence $\boldsymbol{u} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n)$, predicted labels $\boldsymbol{y}' = (y'_1, y'_2, \ldots, y'_n)$, and evaluated labels $\boldsymbol{t}' = (t'_1, t'_2, \ldots, t'_n)$

Ensure: trusted sentences and their corresponding predicted labels S

1: **for** each clause C_i in sentence $\boldsymbol{u}$ **do**
2: dirt = 0 // Flag if a noisy label exists in the clause
3: List tmpClauses is set empty
4: **for** each token $\boldsymbol{w}_i$, y'_i, t'_i in clause C_i **do**
5: **if** $t'_i = 1$ and $y'_i \in$ NamedEntityLabels **then**
6: dirt = 1
7: break
8: **end if**
9: **end for**
10: **if** dirt = 0 **then**
11: Add C_i and its predicted labels to tmpClauses
12: **else if** tmpClauses $\neq$ null **then**
13: Add tmpClauses to S and set tmpClauses empty
14: dirt=0
15: **end if**
16: **end for**
17: $S \leftarrow$ RemoveSent(S) // Remove sentences without // named entities

4 Experiments

4.1 Experimental setup

4.1.1 Dataset

We evaluated our semi-supervised method of using unlabeled data for the NER task on three datasets: two from the most common English NER tasks and one from a CNER task given unlabeled data. The sizes of the two English NER datasets in terms of sentences, tokens, and entities are shown in Table 2. We used the English corpus from Old Newspapers (https://www.kaggle.com/alvations/old-newspapers/data) as additional unlabeled data, because it contains a large number of entities of the same entity types as these two tasks.

1. CoNLL-2003 English NER

This task consists of newswire from the Reuters RCV1 corpus tagged with four different entity types: location, organization, person, and miscellaneous (Tjong Kim Sang and de Meulder, 2003). We used the standard training, development, and test sets for evaluation.

2. OntoNotes 5.0 English NER

Pradhan et al. (2013) compiled a core portion

Table 2 Statistics for the English NER datasets

Dataset	Training	Size of datasets		
		Train	Dev	Test
CoNLL-2003	Tok	204 567	51 578	46 666
	Sent	14 041	3250	3453
	Ent	23 499	5942	5648
OntoNotes 5.0	Tok	1 088 503	147 724	152 728
	Sent	59 924	8528	8262
	Ent	81 828	11 066	11 257

NER: named entity recognition; Tok: tokens; Sent: sentences; Ent: entities; Train: standard training set; Dev: development set; Test: test set

of the OntoNotes 5.0 dataset for the CoNLL-2012 shared task and described a standard train/dev/test split, which we used for our evaluation. This dataset is annotated with 18 entity types, and is much larger than CoNLL-2003. It consists of texts from a wide variety of sources such as broadcast conversations, broadcast news, newswires, magazines, telephone conversations, and web texts.

3. CNER-2017

In the 2017 CCKS CNER task, organizers provided 1198 electronic medical records (EMRs) as labeled training data, 398 EMRs as test data, and 10 420 unlabeled EMRs (Xiao and Wang, 2017). The goal of this task is to identify medical clinically related entity mentions, and classify them into five predefined categories. These categories are symptoms and signs, examination and inspection, disease and diagnosis, treatment, and body parts, which are abbreviated as SYM, EXA, DIS, TRE, and BOD, respectively. Table 3 shows the statistics for the entity in different categories. We randomly sampled 10% of the sentences in the training dataset as the validation set.

Table 3 Statistics for the entities in different categories for CNER-2017

Set	Size of entities in different categories					Total
	SYM	EXA	DIS	TRE	BOD	
Training	7831	9546	722	1048	10 719	29 866
Test	2311	3143	553	465	3021	9493

SYM: symptoms and signs; EXA: examination and inspection; DIS: disease and diagnosis; TRE: treatment; BOD: body parts

4.1.2 Neural NER model training

Our semi-supervised framework can be applied to any neural NER model to improve its performance further by selecting high-quality pseudo labels and creating new labeled data. Here we introduced our baseline neural model settings based on the state-of-the-art single neural NER model structure.

In this study, we used BiLSTM-BiLSTM-CRF architecture for CoNLL-2003 NER, which means using BiLSTM neural networks to encode character-level features as well as contextual features, and using the CRF layer for decoding labels. For the OntoNotes 5.0 dataset, our baseline model was based on the BiLSTM-CNN-CRF structure (Yang and Zhang, 2018), and the only difference was to use CNNs to encode character-level features.

Recently, new word embedding techniques have been proposed to learn the representation of a word based on language models, taking into consideration the context of a word. For example, ELMo (Peters et al., 2018), Flair (Akbik et al., 2018), and BERT (Devlin et al., 2018) all learn the word embeddings dynamically according to the context of a word, improving the performance of multiple NLP tasks to a new level. However, computing the word embeddings as a sentence or a sequence of tokens being processed requires a great amount of computing resources. Our semi-supervised method will produce more labeled data iteratively, which makes the neural model training process much slower when using the dynamic word embeddings. To evaluate our semi-supervised framework efficiently, we adopted the commonly used pre-trained Glove (Pennington et al., 2014) word embeddings (https://nlp.stanford.edu/projects/glove/) for the English NER tasks.

As for the Chinese CNER-2017 task, we tested two ways of forming an input representation: words as tokens and characters as tokens. We pre-trained the word embeddings and character embeddings using the Glove and Word2Vec algorithms (Mikolov et al., 2013). Similarly, the NER model for the CNER-2017 task was based on the BiLSTM-CRF architecture.

The optimizer and hyperparameters we used for training the NER neural models are similar to those of Lample et al. (2016) and Ma and Hovy (2016). We selected the optimal hyperparameters to achieve the best single NER model, so as to obtain the best pseudo labels for further improvements.

4.1.3 Judge model selection

As Section 3.4 described, we constructed the Judge models based on three different algorithms. With the transformed data for the Judge model, we selected a proper threshold for the softmax-based method and trained it with two supervised methods. The other two supervised algorithms were implemented with existing tools such as Keras for FNN and LIBSVM (Chang and Lin, 2011) for SVMs.

We used the development and test datasets for each task to create the transformed data for the Judge model. To compare and select the Judge

model, we divided the transformed data into a training set and test set with a 7:3 ratio.

The softmax-based model did not need training, so we computed the confidence values of the training set and determined the threshold by choosing "one point" on the receiver operating characteristic curve (ROC) space. In our study, we selected the threshold that brings the true positive rate (TPR) (equaling the recall value) closest to 90%. This is a trade-off because a high recall value means that we can identify and filter out most of the wrongly predicted labels, while a too high recall value will also filter out some valuable new entities.

When training the supervised Judge model, we need to balance the training set of the transformed data at first. Since the NER model can achieve a high accuracy and most labels are "O" (meaning not an entity token), the transformed data is heavily out of balance; i.e., the positive samples usually account for less than 5% and the "O"-tag negative samples account for more than 60%. Too unbalanced training data becomes detrimental to learning classifiers. We kept the test set unchanged for testing our Judge model, and applied hybrid sampling methods to balance the training set.

Our hybrid sampling strategies include undersampling on "O"-tag negative samples and oversampling on positive samples. We obtained two processed training sets, the resampled-training set and SMOTE-training set, which use the same undersampling rate and different oversampling methods. The former duplicates positive samples for oversampling, while the latter uses the SMOTE (Chawla et al., 2002) method to create synthetic positive samples. The downsampling rate for "O"-tag negative samples is 1/2 or 1/3 for different NER datasets. After the hybrid sampling strategies, the positive samples in the resampled-training set account for nearly 20%, and reach the same amount as the negative samples in the SMOTE-training set.

We also implemented the ensemble Judge models by voting. To understand which metric can be used to evaluate Judge model performance on trading off between filtering out erroneous pseudo labels and retaining valuable new labeled data, we built three ensemble Judge models with high F_1-score (Ensemble_hF1), high precision (Ensemble_hpr), and high recall value (Ensemble_hrc). The voting strategy that produces the ensemble results was based on the number of times that single models classify a sample as positive. Ensemble_hrc considers a sample positive once it is classified as positive, Ensemble_hF1 regards it as positive only when it is classified as positive at least twice, and Ensemble_hpr obtains a positive sample when it has at least three positive predictions.

In our study, considering the size of the unlabeled data and computing efficiency, we selected the softmax-based Judge model for the CoNLL-2003 NER task, FNN_SMOTE for the OntoNotes 5.0 NER task, and the ensemble Judge model with the highest F_1-score for the CNER-2017 task.

4.1.4 Evaluation metrics

In this study, we used official evaluation metrics (micro-averaged F_1) to evaluate our framework on the two English NER tasks. All the evaluations on the 2017 CCKS CNER task were performed on the official test set using the official evaluation tool. Here we chose the micro-averaged F_1-score under "strict" criteria to evaluate the model performance. The "strict" criteria mean that an entity is correctly identified only when the boundary and category of the entity exactly match a gold one.

4.2 Overall system results

Recent studies that use unlabeled data to pretrain language models for contextual word embeddings have achieved significant improvements on NER tasks, such as ELMo (Peters et al., 2018), flair embeddings (Akbik et al., 2018), and BERT (Devlin et al., 2018). Since our method will produce a large amount of new labeled data, it is unrealistic to learn the dynamic word embeddings with our new training set. Hence, we evaluated our framework based on the neural NER models that are trained with static word embeddings. Consequently, the models trained with dynamic word embeddings are not directly comparable, so we do not report their results. In our work, we focus on exploiting the potential abundant entity information in unlabeled data for NER tasks by automatically creating new trusted labeled data, which is a different perspective for using unlabeled data.

4.2.1 CoNLL-2003 English NER

Table 4 compares the results of our semi-supervised model with other neural network

models on the test dataset from CoNLL-2003 NER. Our semi-supervised model achieved 91.73% F_1 without any additional labeled data or task specific gazetteers, which is a significant increase over the previous best result of 91.21% from Ma and Hovy (2016). When trained on both the training and development sets, our semi-supervised model scored 92.04% F_1, surpassing the state-of-the-art result from Peters et al. (2017) with the same setting.

Table 4 Test set F_1 comparison with previous neural network models on the CoNLL-2003 NER task

Model	F_1-score (%)
Collobert et al. (2011)♣	89.59
Huang et al. (2015)	90.10
Lample et al. (2016)	90.94
Ma and Hovy (2016)	91.21
Chiu and Nichols (2016)♣‡	91.62
Peters et al. (2017)‡	91.93
Rei (2017)	86.26
Ghaddar and Langlais (2018)♣	91.73
Our semi-supervised model	91.73
Our semi-supervised model‡	92.04

NER: named entity recognition. ♣ indicates models that use additional supervised information (gazetteers or Wikipedia); ‡ indicates models trained on both the training and development sets

Table 5 compares the improvements of existing semi-supervised methods on the CoNLL-2003 English dataset. Our semi-supervised method significantly improved the performance of the baseline NER model by 0.94% F_1, and was much better than the multitask learning method that optimizes the NER model as a language model at the same time (Rei, 2017). The performance of our method was close to that of Peters et al. (2017), and our method can be easily combined with other methods for further improvements by introducing more labeled data.

Table 5 Improvements in test set F_1 in CoNLL-2003 NER with semi-supervised methods

Model	F_1-score (%)		△ (%)
	Baseline NER model	Semi-supervised method	
Rei (2017)	86.00	86.26	+0.26
Peters et al. (2017)‡	90.87	91.93	+1.06
Ours	90.79	91.73	+0.94
Ours‡	91.12	92.04	+0.92

NER: named entity recognition. ‡ indicates models trained on both the training and development sets; △ indicates the improvement

4.2.2 OntoNotes 5.0 English NER

Table 6 presents the results of previous neural network models on the test set of the OntoNotes 5.0 English NER task. To quickly evaluate the performance of our semi-supervised framework, we used the public classic Glove.6B.100d word embeddings to train our neural NER model.

Table 6 Test set F_1 comparison with previous neural models on the OntoNotes 5.0 NER task

Model	F_1-score (%)
Chiu and Nichols (2016)♣	86.28
Strubell et al. (2017)	86.99
Li et al. (2017)	87.21
Ghaddar and Langlais (2018)♣	87.95
Our baseline NER model	87.18
Our semi-supervised model	87.80

NER: named entity recognition. ♣ indicates models that use additional supervised information (gazetteers or Wikipedia)

Ghaddar and Langlais (2018) learned lexical features from annotated data, which was produced using supervised information from Wikipedia. They also incorporated the external knowledge resources, gazetteers, as Chiu and Nichols (2016). In this study, our semi-supervised framework learned from only unlabeled data, and obtained a 0.62% F_1-score increase, reaching a comparable performance of 87.80% F_1.

4.2.3 CNER-2017

In the 2017 CCKS CNER task, the way of processing Chinese electronic medical records may have a significant impact on the results. As Table 7 shows, our experimental results proved three effective ways to deal with the CNER task:

Table 7 Performance of our NER models with different settings on the CNER-2017 test set

Model	F_1-score (%)
words-as-tokens	86.17
characters-as-tokens	87.18
characters-as-tokens + data_aug	87.91
+PL w/o Judge model	87.90
Our semi-supervised model	88.56

NER: named entity recognition; data_aug means data augmentation; +PL w/o Judge model means using pseudo labels without selection by the Judge model

1. The model with a characters-as-tokens setting performs better than that with a words-as-tokens

setting by nearly 1% F_1. This is because the former setting can reduce entity boundary recognition errors caused by inappropriate word segmentation.

2. Data augmentation by oversampling sentences that contain entity mentions with the least two categories (TRE and DIS) achieves an obvious gain in F_1-score by 0.73%.

3. Our semi-supervised method with the Judge model to select pseudo labels effectively increases the baseline model by 0.65% F_1, while the model shows a slight decrease in performance when using pseudo labels in the same way without Judge model selection.

Table 8 compares the performance improvement of different models with different settings when using unlabeled data in the 2017 CCKS CNER task. These models all used unlabeled data to create new labeled data with different approaches. Our method achieved an increase of 0.65% F_1, which is more than three times that of the self-taught learning method (Hu et al., 2017). Our method also outperformed the method that combines self-taught learning and active learning, which introduces manually relabeled data (Hu et al., 2017). In contrast, our method has no human intervention. Xia and Wang (2017) trained three machine learning based NER models to tag unlabeled data and decided the final predicted labels by voting. Their vote-based approach for tagging unlabeled data has the same principles as ensemble learning, so their BiLSTM model can achieve a 0.97% F_1 increase to their best F_1-score of 91.14%, while their overall voted model (ensemble) increased by only 0.06% F_1. Thus, their vote-based self-training approach (ensemble) for using unlabeled data is not comparable to ours, which is based on a single model.

Table 8 Improvements in test set F_1 in CNER-2017 when using unlabeled data

Model	F_1-score (%)		△ (%)
	w/o	w/	
Xia and Wang (2017)[1]	90.17	91.14	+0.97
Xia and Wang (2017)[2]	91.02	91.08	+0.06
Hu et al. (2017)[1]	88.47	88.67	+0.20
Hu et al. (2017)[2]	88.47	88.83	+0.36
Ours	87.91	88.56	+0.65

Xia and Wang (2017)[1,2] represent single model and voted model, respectively; Hu et al. (2017)[1] represents adding self-taught learning only; Hu et al. (2017)[2] represents adding both self-taught learning and active learning. △ indicates the improvement

4.3 Analysis

In this part, we analyze the performance of different Judge models and the silver-standard data created with our method based on the limited CoNLL-2003 NER dataset.

1. Judge model selection

To analyze the ability of different Judge models to select wrong pseudo labels, we conducted experiments with one part of the unlabeled data on the CoNLL-2003 NER task; that is, we compared the results on the test set of CoNLL 2003 by applying our semi-supervised method in one iteration. We also evaluated the classification performance of different Judge models with metrics of precision (P), recall (R), and F_1-score (F_1) on the test set of the transformed data.

As introduced in Section 4.1.3, we compared five single Judge models and three ensemble Judge models. Besides the softmax-based model, the other four single Judge models were obtained by training two algorithms on two training sets of transformed data, which were processed with different hybrid sampling strategies (resampled and SMOTE).

Table 9 shows that ensemble Judge models generally have a better performance than the single models. When selecting a proper threshold, the softmax-based Judge model can achieve the best final F_1-score of 91.10% on the test set of the CoNLL-2003 NER task, which was improved by 0.31% compared with that of the baseline NER model with a 90.79% F_1-score. Following the softmax-based Judge model, the Ensemble_hF1 model obtained a comparable final F_1-score in this case.

We can learn from Table 9 that the final F_1-score is not directly proportional to the performance of the Judge model on the test set of transformed data. However, Judge models that have high recall values are more likely to achieve better pseudo-label-selecting performance. Among the single models, the softmax-based model and FNN_Smo model have relatively high recall values. We find that combining the prediction results of these two models can achieve an F_1-score comparable to that of the Ensemble_hF1 model on the test set of transformed data. This means they can recognize different erroneous pseudo labels.

2. Using pseudo labels

Table 10 demonstrates that using unlabeled

Table 9 Results of different Judge models on the test set of transformed data for the Judge model and results on the test set (final F_1-score) of the CoNLL-2003 NER task when applying one iteration of our method

Model	P (%)	R (%)	F_1-score (%)	Final F_1 (%)
Softmax	25.00	**87.01**	38.84	**91.10**
FNN_Res	**64.79**	7.20	12.96	90.72
FNN_Smo	25.80	81.06	**39.14**	90.83
SVM_Res	60.99	13.46	22.05	90.12
SVM_Smo	49.00	15.34	23.37	90.38
Ensemble_hF1	31.21	79.12	**44.76**	**91.07**
Ensemble_hpr	**52.19**	20.57	29.50	91.00
Ensemble_hrc	21.88	**91.21**	35.30	90.92

Ensemble_hF1: ensemble Judge models with high F_1-score; Ensemble_hpr: ensemble Judge models with high precision; Ensemble_hrc: ensemble Judge models with high recall value. P: metric of precision; R: metric of recall. A result in bold represents the maximum value for a single Judge model or an ensemble Judge model

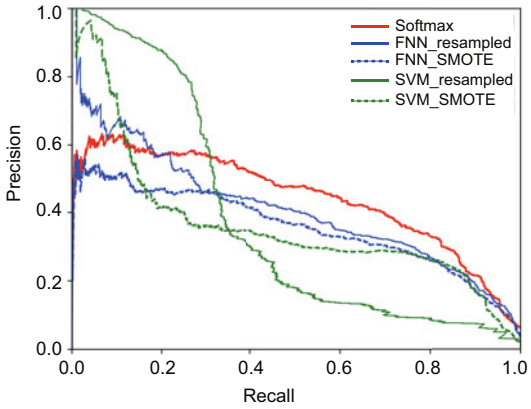

Fig. 3 Results of different single Judge models on the test set of transformed data for the Judge model on the CoNLL-2003 NER task (NER: named entity recognition; FNN: feed-forward neural network; SVM: support vector machine)

References to color refer to the online version of this figure

Table 10 Test set F_1 comparison on the CoNLL-2003 NER task when using one part of the unlabeled data

Model	F_1-score (%)
Our baseline model	90.79
+PL w/o Judge model	90.23
+Softmax Judge model	91.10
+Ensemble_hF1 Judge model	91.07

+PL w/o Judge model means using pseudo labels without selection by the Judge model; Ensemble_hF1: ensemble Judge models with high F_1-score

data with pseudo labels may not always improve the performance of the neural NER model, but using our learned Judge models can consistently improve NER tasks. Compared with the baseline model that does not use unlabeled data, the model using pseudo labels without any selection performs worse on the test set of the CoNLL-2003 NER dataset. When using our learned Judge models to filter out wrong pseudo labels and create new labeled data from part of the unlabeled data to increase the training set, the neural NER model improves its performance. This means that wrong labels in the pseudo labels will impair the training process, while our semi-supervised method can effectively filter them out.

To compare the performance of single Judge models without threshold selection interference, we draw a precision-recall curve based on the test set of transformed data. Fig. 3 shows that the softmax-based model has relatively stable performance and performs better than the other four models most of the time when the recall rate is higher than 0.4.

The untrained softmax-based Judge model can achieve stable and good selection performance for two reasons: First, the size of the transformed dataset for training the Judge model is too small and the positive samples are even fewer, so other trainable models may tend to overfit features from the original data. Second, the softmax-based model can flexibly balance precision and recall by choosing a proper threshold, that is, finding a trade-off between filtering out erroneous pseudo labels and retaining valuable new labeled data.

Since unlabeled data is usually abundant, our Judge model with a high recall value, which filters out most wrong pseudo labels at the cost of omitting some correct ones, can still pick out enough valuable pseudo labels to improve the performance of neural models. In practice, we choose mainly the simple and efficient softmax-based Judge model to continue our semi-supervised method.

5 Conclusions

In this study, we have presented a semi-supervised deep learning method to use unlabeled data for the named entity recognition task. Our method consists of three steps: (1) constructing a single NER model; (2) learning a Judge model; (3) creating new labeled data to improve the performance of the NER model. In the first step, the optimal NER model can be efficiently constructed

by selecting and combining four components of the deep neural NER model structure according to the specific language and task. In the second step, we proposed a method to train and assess the Judge model so as to select a suitable Judge model for different tasks. The softmax-based Judge model and FNN_SMOTE Judge model can recognize most erroneous pseudo labels, which can be used to deal with a great amount of unlabeled data efficiently. In the last step, we proposed a strategy to automatically create new high-quality labeled data and improve the performance of the NER model iteratively. Our semi-supervised method consistently improves the performance of the single neural model on three different NER tasks.

In conclusion, our semi-supervised method can be easily applied to any NER task and further improve the performance with extra unlabeled data. In the future, we would like to explore more methods to use the new labeled data.

Compliance with ethics guidelines

Zhen-zhen LI, Da-wei FENG, Dong-sheng LI, and Xicheng LU declare that they have no conflict of interest.

References

Akbik A, Blythe D, Vollgraf R, 2018. Contextual string embeddings for sequence labeling. Proc 27th Int Conf on Computational Linguistics, p.1638-1649.

Chang CC, Lin CJ, 2011. LIBSVM—a library for support vector machines. *ACM Trans Intell Syst Technol*, 2, Article 27. https://doi.org/10.1145/1961189.1961199

Chawla NV, Bowyer KW, Hall LO, et al., 2002. SMOTE: synthetic minority over-sampling technique. *J Artif Intell Res*, 16:321-357. https://doi.org/10.1613/jair.953

Chiu JPC, Nichols E, 2016. Named entity recognition with bidirectional LSTM-CNNs. *Trans Assoc Comput Ling*, 4:357-370. https://doi.org/10.1162/tacl_a_00104

Collobert R, Weston J, Bottou L, et al., 2011. Natural language processing (almost) from scratch. *J Mach Learn Res*, 12:2493-2537.

Cortes C, Vapnik V, 1995. Support-vector networks. *Mach Learn*, 20(3):273-297. https://doi.org/10.1007/BF00994018

Devlin J, Chang MW, Lee K, et al., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805

Ghaddar A, Langlais P, 2018. Robust lexical features for improved neural network named-entity recognition. Proc 27th Int Conf on Computational Linguistics, p.1896-1907.

Grandvalet Y, Bengio Y, 2006. Entropy regularization. In: Chapelle O, Schölkopf B, Zien A (Eds.), Semi-supervised Learning. MIT Press, Cambridge, Mass, p.151-168. https://doi.org/10.7551/mitpress/9780262033589.001.0001

Hochreiter S, Schmidhuber J, 1997. Long short-term memory. *Neur Comput*, 9(8):1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu J, Shi X, Liu Z, et al., 2017. HITSZ_CNER: a hybrid system for entity recognition from Chinese clinical text. China Conf on Knowledge Graph and Semantic Computing, p.1-6.

Huang Z, Xu W, Yu K, 2015. Bidirectional LSTM-CRF models for sequence tagging. https://arxiv.org/abs/1508.01991

Jagannatha AN, Yu H, 2016. Structured prediction models for RNN based sequence labeling in clinical text. Proc Conf on Empirical Methods in Natural Language Processing, p.856. https://doi.org/10.18653/v1/D16-1082

Lafferty JD, McCallum A, Pereira FCN, 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. Proc 18th Int Conf on Machine Learning, p.282-289.

Lample G, Ballesteros M, Subramanian S, et al., 2016. Neural architectures for named entity recognition. North American Chapter of the Association for Computational Linguistics, p.260-270. https://doi.org/10.18653/v1/N16-1030

Lee DH, 2013. Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks. Work Shop on Challenges in Representation Learning, p.1-6.

Li PH, Dong RP, Wang YS, et al., 2017. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. Proc Conf on Empirical Methods in Natural Language Processing, p.2664-2669. https://doi.org/10.18653/v1/D17-1282

Liao WH, Veeramachaneni S, 2009. A simple semi-supervised algorithm for named entity recognition. Proc NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing, p.58-65.

Ma XZ, Hovy E, 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. Proc 54th Annual Meeting of the Association for Computational Linguistics, p.1064-1074. https://doi.org/10.13140/RG.2.1.2182.5685

Mesnil G, He X, Deng L, et al., 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. Interspeech, p.1-5.

Mikolov T, Sutskever I, Chen K, et al., 2013. Distributed representations of words and phrases and their compositionality. Proc 26th Int Conf on Neural Information Processing Systems, p.3111-3119.

Pennington J, Socher R, Manning CD, 2014. Glove: global vectors for word representation. Proc Empirical Methods in Natural Language Processing, p.1532-1543.

Peters ME, Ammar W, Bhagavatula C, et al., 2017. Semi-supervised sequence tagging with bidirectional language models. Proc 55th Annual Meeting of the Association for Computational Linguistics, p.1756-1765. https://doi.org/10.18653/v1/P17-1161

Peters ME, Neumann M, Iyyer M, et al., 2018. Deep contextualized word representations. https://arxiv.org/abs/1802.05365

Pradhan S, Moschitti A, Xue N, et al., 2013. Towards robust linguistic analysis using ontonotes. Proc 7th Conf on Computational Natural Language Learning, p.143-152.

Qi YJ, Collobert R, Kuksa P, et al., 2009. Combining labeled and unlabeled data with word-class distribution learning. Proc 18th ACM Conf on Information and Knowledge Management, p.1737-1740. https://doi.org/10.1145/1645953.1646218

Rei M, 2017. Semi-supervised multitask learning for sequence labeling. 55th Annual Meeting of the Association for Computational Linguistics, p.2121-2130. https://doi.org/10.18653/v1/P17-1194

Schmidhuber J, 2015. Deep learning in neural networks: an overview. *Neur Netw*, 61:85-117. https://doi.org/10.1016/j.neunet.2014.09.003

Shen YY, Yun H, Lipton ZC, et al., 2017. Deep active learning for named entity recognition. https://arxiv.org/abs/1707.05928

Strubell E, Verga P, Belanger D, et al., 2017. Fast and accurate entity recognition with iterated dilated convolutions. Proc Conf on Empirical Methods in Natural Language Processing, p.2670-2680.

Sun YQ, Li L, Xie ZW, et al., 2017. Co-training an improved recurrent neural network with probability statistic models for named entity recognition. Int Conf on Database Systems for Advanced Applications, p.545-555. https://doi.org/10.1007/978-3-319-55699-4_33

Tjong Kim Sang EF, de Meulder F, 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. Proc 7th Conf on Natural Language Learning at HLT-NAACL, p.142-147. https://doi.org/10.3115/1119176.1119195

Wu H, Prasad S, 2018. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE Trans Image Process*, 27(3):1259-1270. https://doi.org/10.1109/TIP.2017.2772836

Xia Y, Wang Q, 2017. Clinical named entity recognition: ECUST in the CCKS-2017 shared task 2. CEUR Workshop Proc, p.43-48.

Xiao Y, Wang Z, 2017. Clinical Named Entity Recognition Evaluation Tasks at CCKS 2017. http://ceur-ws.org/Vol-1976/

Yang J, Zhang Y, 2018. NCRF++: an open-source neural sequence labeling toolkit. Proc 56th Annual Meeting of the Association for Computational Linguistics, p.74-79. http://aclweb.org/anthology/P18-4013

Zhai F, Potdar S, Xiang B, et al., 2017. Neural models for sequence chunking. Proc 31st AAAI Conf on Artificial Intelligence, p.3365-3371.