Effective Use of Bidirectional Language Modeling for Medical Named Entity Recognition

Devendra Singh Sachan[1,*], **Pengtao Xie**[1], **and Eric P Xing**[1]

[1]Petuum Inc, Pittsburgh, 15222, USA
[*]sachan.devendra@gmail.com

ABSTRACT

Biomedical named entity recognition (NER) is a fundamental task in text mining of medical documents and has a lot of applications. Existing approaches for NER require manual feature engineering in order to represent words and its corresponding contextual information. Deep learning based approaches have been gaining increasing attention in recent years as their weight parameters can be learned end-to-end without the need for hand-engineered features. These approaches rely on high-quality labeled data which is expensive to obtain. To address this issue, we investigate how to use widely available unlabeled text data to improve the performance of NER models. Specifically, we train a bidirectional language model (Bi-LM) on unlabeled data and transfer its weights to a NER model with the same architecture as the Bi-LM, which results in a better parameter initialization of the NER model. We evaluate our approach on three datasets for disease NER and show that it leads to a remarkable improvement in F1 score as compared to the model with random parameter initialization. We also show that Bi-LM weight transfer leads to faster model training. In addition, our model requires fewer training examples to achieve a particular F1 score.

Introduction

The field of biomedical text mining has received increased attention in recent years due to the rapid increase in the number of publications, scientific articles, reports, medical records etc. which are available in electronic format and can be readily accessed. These biomedical data contains a lot of mentions of biological and medical entities such as chemical ingredients, genes, proteins, medications, diseases, symptoms etc. Figure 1 shows a medical text that contains 7 diseases (highlighted in red color) and 4 anatomical entities (highlighted in yellow color). The accurate identification of such entities in text collections is a very important subtask for information extraction systems in the field of biomedical text mining as it helps in transforming the unstructured information in texts into structured data. Search engines can index, organize and link medical documents using such identified entities and this can improve medical information access as the users will be able to gather information from many pieces of text. The identification of entities can also be used for mining relations and extracting associations from medical research literature. For example, one can extract various drug-gene interactions which can be stored in a relational database so that computer programs can perform inference among them. We can also do binary relation extraction among specific text entities such as "*symptom of*" relation between diseases and symptoms, "*side effect of*" relation between drugs and diseases etc., and store this information in health knowledge bases.[1] Knowledge of text entities is also one of the first steps in more advanced natural language understanding tasks requiring complex pipelines such as Question Answering (QA) systems for biomedical dataset, entity normalization and its linking to standard knowledge databases such as MeSH. Advanced applications of entity identification include automatic text summary generation algorithms which can better summarize user's conversations in medical forums and use of chat bots in automated healthcare sector. We refer to this task of identification and tagging of entities in text into predefined categories such as diseases, chemicals, genes etc. as Named Entity Recognition (NER).

NER has been a widely studied task in the area of natural language processing (NLP) and there has been a number of works which have applied machine learning approaches for NER in the medical domain. Building NER systems with high precision and high recall for medical domain is a quite challenging task due to high linguistic variation in data. Firstly, a simple dictionary based approach which only does exact matching will fail to correctly tag ambiguous abbreviations in text. For example, the term 'STD' can refer to the phrase "*sexually transmitted disease*" or to a term which is an Internet Standard. Secondly, there exists many forms of entity names usage in clinical texts which can cause issues in entity linking and normalization. One example is that of disease "*leukemia*" which appears in different forms like "*lymphoblastic leukemia*", "*null-cell leukemia*", "*lymphoid leukemia*" etc. Thirdly, as the vocabulary of biomedical entities such as proteins is quite vast and is evolving quite rapidly, it makes the task of entity identification even more challenging and error prone as it is difficult to create labeled training examples which has a wide coverage. For example, recently discovered proteins such as "*small humanin-like peptides*" (SHLPs 1-6) may occur in new text articles in abbreviated form and this can lead to wrong predictions by NER taggers which were trained on unrelated examples. Also, in contrast to general text, entities in medical domain can have longer names as shown

in Figure 1 which can easily lead a NER tagger to incorrectly predict all the tags. Lastly, state of the art machine learning approaches for NER task rely on high-quality labeled data which is expensive to obtain and is therefore available only in limited quantity. So, there is a need for those approaches which can utilize easily accessible unlabeled data to improve the performance of their supervised variants.

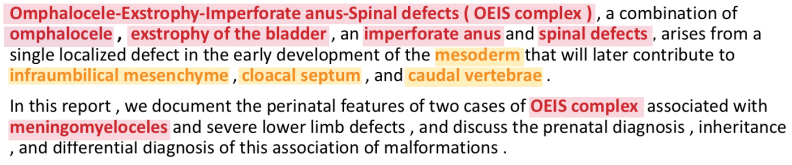


Figure 1. Example of disease and anatomical entities in medical text. Disease entities are highlighted in red color and anatomical entities are highlighted in yellow color.

In this work, we propose an approach which uses unlabeled data to pre-train the weights of a NER model using a related task. Specifically, we do language modeling in both forward and backward directions to pre-train the weights of NER model which is later finetuned using supervised training data. Our NER model applies bidirectional Long Short Term Memory (Bi-LSTM) at sequence level which has shown to effectively model the left and right context information around the center word for every time step and this context based representation of word helps in disambiguation of abbreviations. Bi-LSTM's also map terms like "*lymphoblastic leukemia*", "*null-cell leukemia*" and its varied forms into a common latent space which captures the semantic meaning in the phrases and thus can be easily normalized to a common entity type. This powerful representation of word context in a latent semantic space by Bi-LSTM can also help in correct classification in the case of unseen entities as NER categories with similar contexts are mapped closer together. For the case of longer entity names, we believe that bidirectional language modeling can help in learning the relations between adjacent words and by weights transfer, NER model should be able to learn this pattern. In addition to bidirectional language modeling, we also use unlabeled data from a large corpus of PubMed abstracts to train word vectors which are fed to Bi-LSTM. This has shown to improve the performance of NER systems over randomly initialized word vectors.

Background and Related Work

NER can be devised as a supervised machine learning task in which the training data consists of labels for each token in text. A typical approach for NER task is to extract word level features followed by training a linear model for word level classification. Researchers have worked on carefully designing hand-engineered features to represent a word such as the use of part-of-speech tags, capitalization information, use of rules such as regular expressions to identify numbers, use of gazetteers etc. A combination of supervised classifiers using such features was used to achieve best performance on CoNLL-2003 benchmark dataset related to NER task[2]. Lafferty et al.[3] popularized the use of graphical models such as linear chain Conditional Random Fields (CRF) for NER tasks. Among the early approaches of NER systems in biomedical domain include ABNER[4], BANNER[5] and GIMLI[6] which utilized a variety of lexical, contextual and orthographic features as input to a linear chain CRF. Leaman et al.[7] proposed semi-Markov models to perform joint NER and normalization using external dictionaries.

Recently, there has been a growing interest in using neural network based methods for natural language understanding tasks as they can be trained end-to-end using only the available labeled data. As compared to hand engineered features which are task dependent and sometimes labor intensive, in neural networks features are learned from data during training step. In their seminal work, Collobert et al.[8] trained models such as window based and sentence based convolutional feed forward deep neural networks for tasks such as POS tagging, chunking, semantic role labeling and demonstrated that the performance of neural network based methods can be competitive with the state of the art systems for these tasks. Huang et al.[9] uses spelling features, context features, word embeddings along with bidirectional Long Short Term Memory (Bi-LSTM) followed by a CRF layer for NER tasks on newswire texts. In order to better use morphological and syntactic features from characters, Lample et al.[10] makes use of last time step hidden features of Bi-LSTM applied to character embeddings in addition to word embeddings as features for sequence level Bi-LSTM. Similarly, in order to learn character level information, Chiu et al.[11] and Ma et al.[12] uses Convolutional Neural Networks (CNNs) on character embeddings followed by max pooling.

There have been several recent works which apply neural network based approaches on biomedical text. Wei et al.[13] applies Bi-LSTM and hand engineered features based CRF model followed by Support Vector Machine (SVM) classifier in final stage which combines the output of the two systems. Zeng et al.[14] experiments with Bi-LSTM based character features along with

Bi-LSTM based sequence layer for the task of drug named entity recognition. Habibi et al.[15] does a careful study of the effect of pre-trained word embeddings on several medical NER datasets based on a deep learning model proposed by Lample et al.[10]

In this work, we investigate if the parameters learned during language modeling (LM) task be effectively used to increase the performance of a neural network based NER system. In this task, a recurrent neural network (LSTM) is used to model the context from the previous words to predict the next word in the sequence. In a related work, Dai et al.[16] perform pre-training of word level LSTM using LM and sequence autoencoder and apply succesfully on text classification tasks. Kim et al.[17] shows that a LM with fewer parameters can be trained using only character level information as CNN's applied over character embeddings can be effectively used to extract character level features which are given as input to word level LSTM. As mentioned above, current NER models use a sequence level Bi-LSTM whose word level features are a concatenation of unsupervised word embeddings and features extracted from characters using either character level CNN or character level LSTM. In our NER system, we pre-train the weights of the model from a language model with the same architecture. We believe that such pre-training can prevent overfitting, improve model training and its convergence speed. In order to pre-train the weights of Bi-LSTM in NER task, we do forward and backward language modeling in which the weights of character CNN and word embeddings are shared. This will be discussed in detail in the methods section. We show that such pre-training of weights helps as it increases the F1 score on three tasks consisting of disease NER in medical texts.

Methods

In the first part, we discuss the approach to extract character level features from a word. Next we discuss, sequence modeling using word level features. Then, we discuss the training methodology of NER model followed by a discussion of language modeling. Lastly, we discuss datasets and model training strategies.

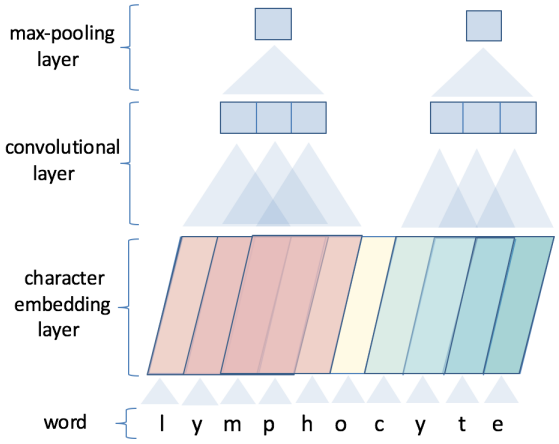


Figure 2. Character CNN block diagram

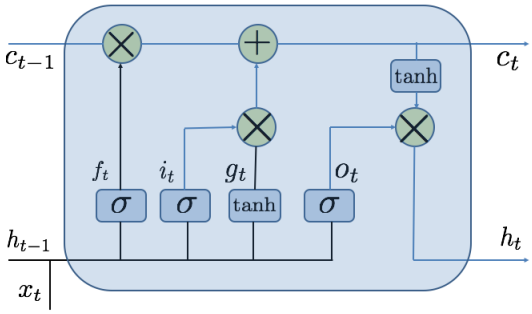


Figure 3. LSTM block diagram

Character Level Convolutional Neural Network. Convolutional Neural Networks[18] (CNNs) are widely used in computer vision tasks requiring feature extraction from images such as object recognition[19] and visual question answering[20]. In NLP, where the data is mostly sequential, successful applications of CNNs include text classification[21] and sequence labeling tasks[8]. In our work, we use CNNs to extract word level features from characters as they can encode morphological and syntactic patterns observed in languages.

Similar to the concept of word embeddings, each character in the text is represented by its embedding. Let V_c be the vocabulary of characters and let D_c be the dimensions of character embedding. These character embeddings are stored in a lookup table $W_c \in R^{V_c \times D_c}$. In order to have the same length for every word in a mini-batch, each word is right padded with a special padding character so that the length of the word is same as that of the longest word in every mini-batch. The embedding of the padding character is always a zero vector. In order to compute character level features, we perform valid 1-D convolution along the temporal dimension. Mathematically, this can be written as:

$$y^k[i] = f(W^k * X[:, i+s-1] + b) \tag{1}$$

In the above equation, $*$ is the dot product operator, b is the bias, $X \in R^{D_c \times w_l}$ is the character based embedding of word, w_l is the maximum length of a word in a mini-batch, W^k are filter weights, s is convolution stride, f can be any nonlinear function

such as tanh or rectifier. Filters of different strides are used to compute character level features. Finally, max-pooling over time is done to get a single feature for every weight matrix used in CNNs. All the features are concatenated in order to obtain character based word representation v^w_{char}. A block diagram of character level CNN is shown in Figure 2.

Word Level Recurrent Neural Network. Recurrent Neural Networks[22] such as Long Short Term Memory[23] (LSTM) are widely used in various NLP tasks because they can easily model the long-term dependence structure common in languages with their memory cells and explicit gating mechanism. The dynamics of LSTM is controlled by input vector (x_t), forget gate (f_t), input gate (i_t), output gate (o_t), cell state (c_t), hidden state (h_t) which are computed as below:

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2a}$$
$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2b}$$
$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{2c}$$
$$g_t = \tanh(W_g * [h_{t-1}, x_t] + b_g) \tag{2d}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2e}$$
$$h_t = o_t \odot \tanh(c_t) \tag{2f}$$

Here, c_{t-1} and h_{t-1} are the cell state and hidden state from previous time step, σ is the sigmoid function ($\frac{1}{1+e^{-x}}$), tanh is the hyperbolic tangent function ($\frac{e^x - e^{-x}}{e^x + e^{-x}}$), $\odot$ denotes element-wise multiplication. Figure 3 shows a block diagram of the LSTM cell. The parameters of LSTM are shared across all the time steps. For an input x_t to LSTM, we concatenate the word embeddings (v^w_{emb}) and character CNN (v^w_{char}) features for the word.

$$x_t = [v^w_{emb}\ v^w_{char}] \tag{3}$$

In our work, the LSTM module consists of two LSTM cells, a forward LSTM cell and a backward LSTM cell which is referred to as bidirectional LSTM (Bi-LSTM). The input to the backward LSTM is the reversed order of words in the sequence.

NER Model Training. For every word, hidden state representations computed using a Bi-LSTM are concatenated ($h_t = [h^f_t\ h^b_t]$) and is given as input to the decoder layer. The decoder computes an affine transformation of the hidden states for every word. Let H be the dimensionality of the Bi-LSTM hidden state, T be the total number of tags, then the output of decoder is the following

$$d_t = W_d h_t + b \tag{4}$$

Here, $W_d \in R^{T \times H}$ is the decoder weight and b is the bias. Decoder outputs are referred to as *logits* in the following discussion. In order to compute the probability of a tag for a word, softmax function is applied to its respective logits.

$$\hat{y}_{t,j} = \frac{\exp(d_{t,j})}{\sum_{j=1}^{T} \exp(d_{t,j})} \tag{5}$$

In this expression, $\hat{y}_{t,j}$ refers to the probability of tag j at time t. If $T_1, T_2, .., T_N$ is the sequence of tags in the training corpus, then the cross entropy (CE) loss is calculated as:

$$\text{CE} = -\sum_{i=1}^{N} \sum_{j=1}^{T} \mathbb{1}(T_i = \hat{y}_{ij}) \log y_{ij} \tag{6}$$

Figure 4 shows the block diagram of our NER model architecture. In order to learn the parameters of our model, we minimize the average CE loss between the actual tag and its predicted likelihood score by backpropagation through time[24] (BPTT) approach.

Language Modeling. In this section, we provide a short description of language modeling, as its parameters are used to initialize the NER model. In this, the task is to train a model which maximizes the likelihood of a given sequence of words. At every step, a language model computes the probability of the next word in a sequence given all the previous words. If the sequence of words are w_1, w_2, .., w_n, the overall probability can be written as:

$$\text{P}(w_1, w_2, .., w_n) = \prod_{i=2}^{n+1} \text{P}(w_i | w_1, .., w_{i-1}) \tag{7}$$

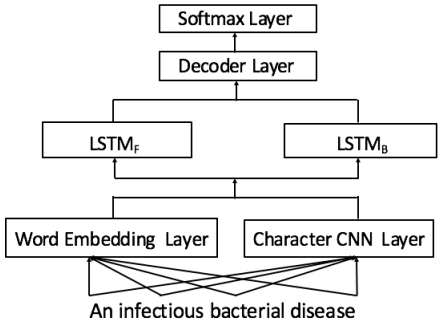


Figure 4. NER model architecture

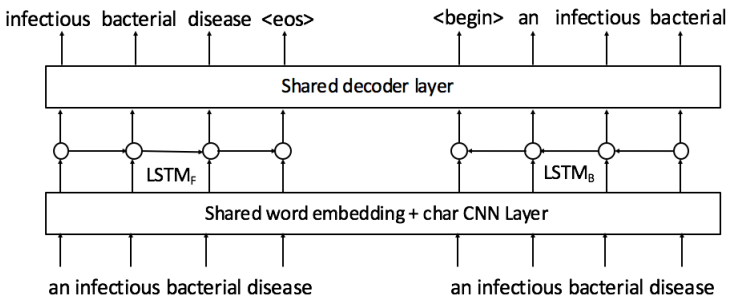


Figure 5. Bidirectional language modeling architecture

Properties	**NCBI Disease**	**BC5CDR**	**Clinical Notes**
Document type	Pubmed abstracts	Pubmed abstracts	Clinical notes
# Disease mentions	6,892	5,818	1,982
# Sentences	7,295	13,907	8,024
# Words	184,552	360,315	132,819
# Training documents	593	500	496
# Development documents	100	500	165
# Test documents	100	500	167

Table 1. General statistics of the datasets used in this work. # symbol stands for the term 'number of'

Here, w_{n+1} is a special symbol for end of sequence. LSTM can be used to predict the probability of next word given the current word and previous sequence of words[25]. This is done by taking an affine transform of the hidden state of LSTM at every time step so that the logit score for every word in vocabulary can be computed. Similar to NER model, BPTT algorithm can be applied on CE loss to learn the parameters of the network. We refer to this approach as forward language model (LM$_F$).

We can also model the reverse sequence of words in exactly the same manner. In this we compute the probability of the reverse sequence as follows:

$$P(w_n, w_{n-1}, .., w_1) = \prod_{i=n-1}^{i=0} P(w_i | w_{i+1}, .., w_n) \tag{8}$$

Here, w_0 is a special symbol for start of sequence. We refer to this approach as backward language model (LM$_B$). The network architecture of both LM$_F$ and LM$_B$ is similar to NER model and is shown in Fig 5. Both LM$_F$ and LM$_B$ share the weights of word embedding layer, character embedding layer, character CNN filters and decoder layer. In order to do bidirectional language modeling, we perform joint training of our model by minimizing the sum of CE loss of both LM$_F$ and LM$_B$.

Dataset Preparation and Evaluation. We evaluate our model on three datasets: NCBI Disease[26] dataset, BioCreative V Chemical Disease Relation Extraction (BC5CDR) task[27] and Clinical Notes dataset [I]. Overall summary of these datasets such as number of sentences and words, training/development/test splits are mentioned in Table 1. We use training and development splits from all three datasets as unlabeled data for language modeling task.

For the Clinical Notes dataset, we do sentence segmentation using Spacy toolkit [II] followed by sentence tokenization using Penn Treebank style tokenizer from NLTK[III]. We convert all text to lowercase and use a special token for numbers. In all our experiments, we use IOB tagging format for the output tags. For evaluation, we report the precision, recall and F1 scores over test set. We do exact matching of entity chunks in order to compute these metrics.

Experiments. For all datasets, we tune the hyperparameters of our model on development set. Final training is done on both the training and development sets. We use Pytorch[IV] framework for all our experiments. We first describe the model architecture followed by details of language model training and NER model training.

[I] These clinical notes were obtained from hospital.
[II] https://spacy.io/
[III] http://www.nltk.org/_modules/nltk/tokenize/treebank.html
[IV] https://github.com/pytorch/pytorch

Model Architecture Details. As we use language model weights to initialize the parameters of NER model, both the models have identical configurations except the last decoder layer. Dimensions of character embeddings and word embeddings are set to 100 and 300 respectively. CNN filters have widths (w) in the range from 1 to 7. Number of filter weights is computed as a function of filter width as $min(200, 50 * w)$. The hidden state of LSTM has 300 dimensions. As the decoder layer is not shared between NER model and language model, the dimensions of decoder layer are different for each of them. For NER model, as it concatenates the hidden states of forward and backward LSTM to give input to decoder layer, the dimensions of decoder matrix are $W_d^{NER} \in R^{600 \times T}$. In the case of language model, dimensions of decoder matrix are $W_d^{LM} \in R^{300 \times V}$. Here, V is the number of words in vocabulary.

Language Model Training. To initialize word embedding parameters for language modeling task, we learn the word vectors using skip-gram[28] approach[V] on a large collection of PubMed abstracts available from BioASQ Task 4a challenge[29]. The embeddings of those words in NER datasets which are not present in these vectors are initialized to zero. Parameters of LSTM are initialized uniformly in the range $(-0.005, 0.005)$. We use Xavier initialization[30] for all other parameters as it has shown to improve convergence of the model during training. We tie the weights of the word embedding layer and the decoder layer while training as it has shown to improve perplexity[31]. We use mini-batch stochastic gradient descent (SGD) with a batch size of 32. At the start of every mini-batch step, the LSTM starts from zero initial state. We do sentence level language modeling and the network is trained using BPTT. Optimization is done using Adam Optimizer[32] with default parameters settings. We use an initial learning rate of 0.001 and decay the learning rate by multiplying it by 0.9 after every epoch. Model is trained for 20 epochs and early stopping is done if the perplexity does not improve for 3 consecutive epochs. In order to regularize the model, we perform dropout[33] with probability 0.25 on the input to hidden layer of LSTM and with probability 0.6 on the output of LSTM hidden layer. In order to prevent gradient explosion problem, we perform gradient clipping by constraining the norm of the gradient to be less than 5[34].

NER Model Training. Once the language model is trained, we remove the decoder layer from top and transfer the remaining weights to NER model with same configuration. The parameters of the decoder layer of NER model are initialized using Xavier initialization. We use mini-batch SGD with a batch size of 20 to train the model. The hidden state of the LSTM is initialized with zero at the start of every mini-batch. Network is trained using BPTT using Adam Optimizer. We use an initial learning rate of 0.001 and we decay it by multiplying by a factor of 0.1 if the F1 score does not improve for one epoch. We train the model for 30 epochs and do early stopping if the F1 score does not improve for five consecutive epochs. We select the value of dropout for each corpus using grid search according to best performance on the corresponding development set. Gradient clipping is done using a maximum norm of 5.

Models Based on Weights Pre-training. We compare the performance of the following methods which are based on the manner in which the NER model weights are initialized and finetuned.

- **No pre-training:** We randomly initialize the parameters of NER model except word embeddings followed by supervised training.
- **LM_F pre-training:** We initialize the parameters of the NER model using the forward language model weights. The parameters of backward LSTM and decoder are randomly initialized.
- **LM_B pre-training:** We initialize the parameters of the NER model using the backward language model weights. The parameters of forward LSTM and decoder are randomly initialized.
- **Bi-LM pre-training:** In this, the parameters of NER model are initialized using the bidirectional language model weights. The parameters of decoder are randomly initialized.

Results

For the above models, precision, recall and F1 scores for NCBI Disease, BC5CDR and Clinical Notes dataset are shown in Table 2, Table 3 and Table 4 respectively.

From the results, we see that the approach of Bi-LM pre-training performs the best across all the datasets. For NCBI Disease dataset, the F1 score is 84.58, which is an absolute improvement of 1.46% in F1 score over just randomly initializing the weights of the model. Similarly, for BC5CDR dataset, our model's F1 score is 83.44% and for Clinical Notes dataset it is 84.44%. Our model gives an absolute improvement of 1.31% and 1.04% in F1 score respectively over the model with no pre-training. We note that for all the datasets, LM_F pre-training and LM_B pre-training gives an improvement over the case with no pre-training. From the results, we also observe that the Bi-LM pre-training achieves higher F1 score and precision in

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

	Precision	Recall	F1 Score
No pre-training	81.01	85.34	83.12
LM_F pre-training	81.94	86.49	84.16
LM_B pre-training	82.15	85.76	83.91
Bi-LM pre-training	**82.28**	**87.02**	**84.58**

Table 2. Precision, recall and F1 score for all the models on NCBI Disease dataset

	Precision	Recall	F1 Score
No pre-training	81.18	83.10	82.13
LM_F pre-training	82.27	83.28	82.78
LM_B pre-training	81.62	83.81	82.70
Bi-LM pre-training	**82.30**	**84.60**	**83.44**

Table 3. Precision, recall and F1 score for all the models on BC5CDR dataset

	Precision	Recall	F1 Score
No pre-training	81.59	85.27	83.39
LM_F pre-training	82.00	85.51	83.72
LM_B pre-training	81.94	**86.22**	84.03
Bi-LM pre-training	**83.18**	85.75	**84.44**

Table 4. Precision, recall and F1 score for all the models on Clinical Notes dataset

	Precision	Recall	F1 Score
NCBI Disease	83.48	87.33	85.36
BC5CDR	85.41	83.90	84.65
Clinical Notes	84.63	87.65	86.11

Table 5. Scores for Bi-LM pre-trained model after the addition of CRF layer on top of decoder layer

comparison to LM_F pre-training and LM_B pre-training, thus highlighting the importance of performing language modeling in both directions.

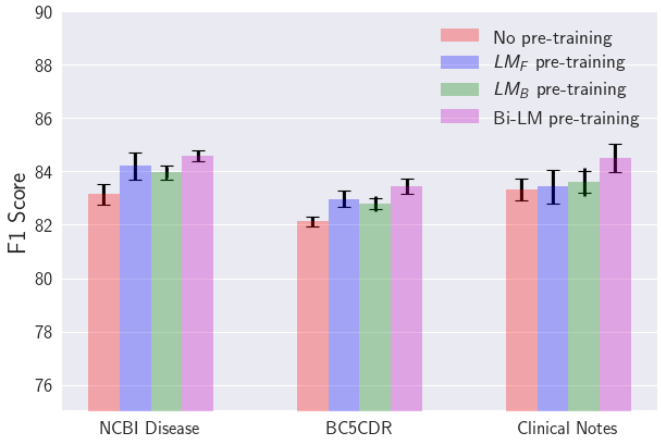


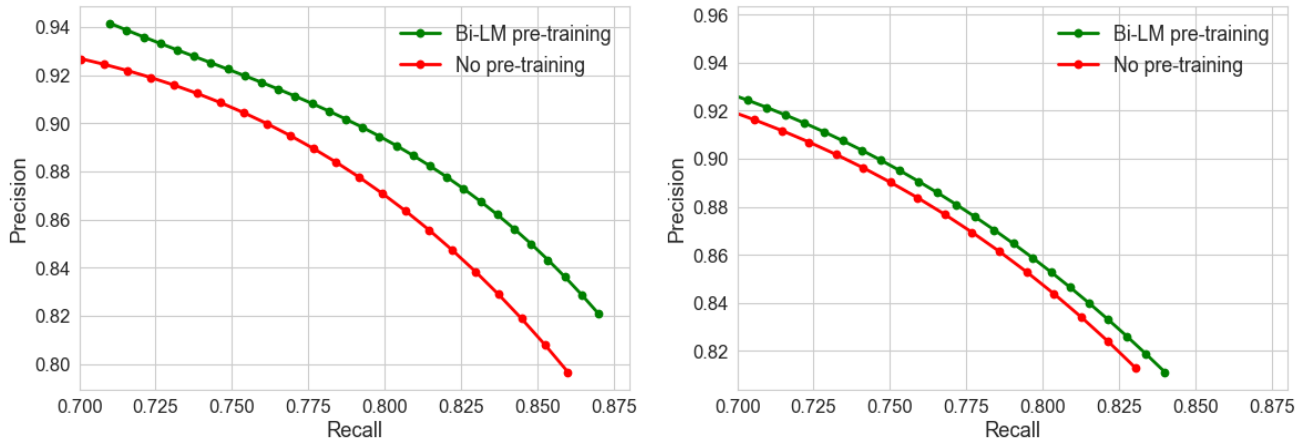
Figure 6. F1 score and standard error for all datasets and models. The error bars indicate the standard deviation of the F1 scores.

As the datasets have small test sets, we also compute the standard deviation for 10 runs of the best hyperparameter settings for the above models. In Figure 6, we show the mean value of F1 score for all models and datasets along with their standard deviation. We see that the standard deviation of F1 Score of Bi-LM pre-trained model is less than 0.3 for NCBI Disease and BC5CDR datasets and is less than 0.5 for Clinical Notes dataset.

In Figure 7a and 7b, we plot the precision-recall curve for NCBI Disease and BC5CDR datasets. From the plots, we see that the area under curve of Bi-LM pre-trained NER model is always more than that of the model with randomly initialized weights.

Addition of CRF Layer. We also perform experiments by adding a CRF layer on top of decoder layer. As compared to word level prediction, CRF has transition parameters which models the probability of transition from one tag to another tag and does optimization of parameters at the sequence level. It also performs inference at sequence level using Viterbi algorithm[35]. For mathematical details of CRF, we refer the reader to Collobert et al.[8] In Table 5, we show the best scores of Bi-LM pre-trained model with CRF layer for all the datasets. We note that the addition of CRF layer gives an additional improvement of 1-1.5% in F1 score across all the datasets. [VI]

[VI] As noted in Chiu et al.[11], we also observed that training process by adding a CRF layer becomes unstable. The results in Table 5 show our best scores obtained in 3 runs for a dataset.



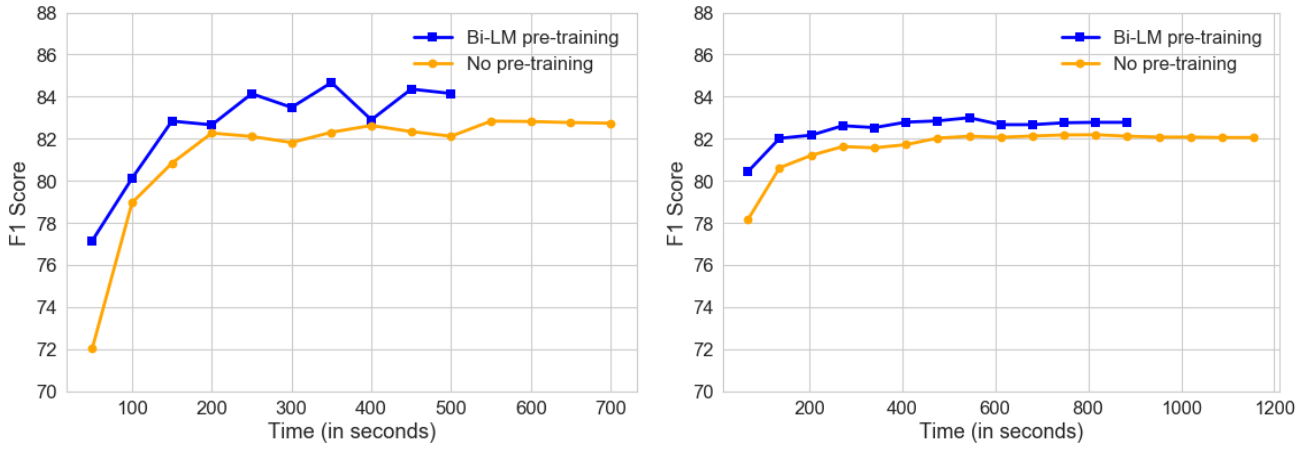
(a) NCBI Disease dataset

(b) BC5CDR dataset

Figure 7. Precision-recall curves for the models with Bi-LM pre-training and no pre-training

We also study additional properties of the NER model with Bi-LM pre-training and compare it with the NER model with random initialization of weights.

Rate of Convergence. We monitor the overall clock time and the time taken per epoch required for the two models to converge. We follow the same training process as outlined above. A typical run for both the models on NCBI Disease and BC5CDR dataset is shown in Figure 8a and Figure 8b respectively. For NCBI Disease dataset, the model with Bi-LM pre-training converges in 10 epochs ($\approx$ 500s) as compared to the model with no pre-training which typically converges in 14 epochs ($\approx$ 700s). We observe a similar trend in BC5CDR dataset where Bi-LM pre-training results in convergence in 11 epochs ($\approx$ 900s) whereas no pre-training takes around 17 epochs ($\approx$ 1150s). Thus, in terms of total time taken, we observe that pre-training using Bi-LM weights results in faster convergence by about 28-35% as compared with random parameter initialization setting. We also see that Bi-LM pre-training results in better F1 score from first epoch onwards for both the datasets.



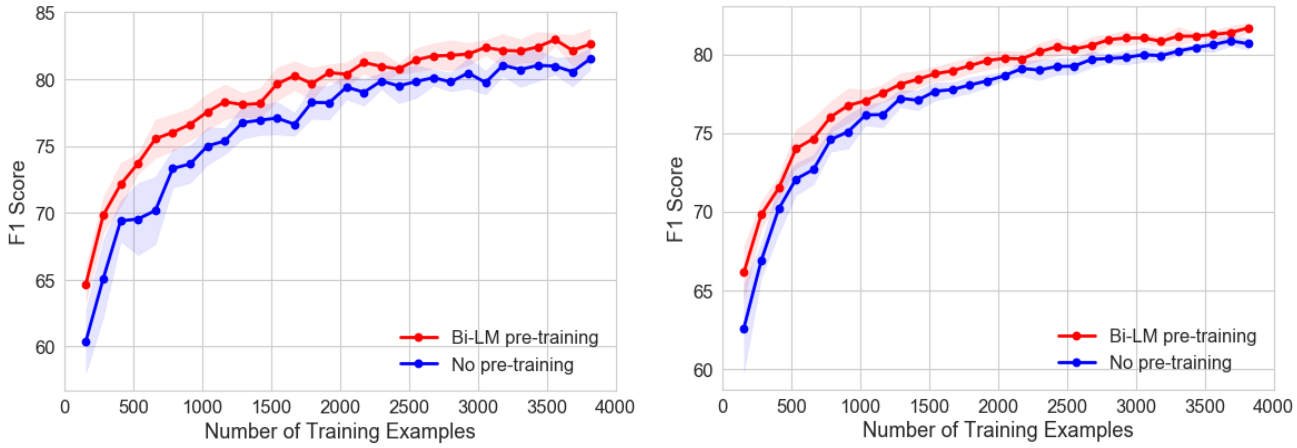
(a) NCBI Disease dataset

(b) BC5CDR dataset

Figure 8. F1 score vs overall time taken for training to converge for models with Bi-LM pre-training and no pre-training.

Effect of Number of Training Examples. In this setup, we analyze the performance of models by feeding them with an increasing number of examples during training process (learning curve). We compare the learning curve of the Bi-LM pre-trained model with the model with no pre-training. The learning curve for NCBI Disease and BC5CDR datasets both the models is shown in Figure 9a and 9b respectively. We can see that pre-trained model is always optimal (higher F1 score) for

any setting of the number of training examples.



(a) NCBI Disease dataset

(b) BC5CDR dataset

Figure 9. F1 score vs number of training examples for models with Bi-LM pre-training and no pre-training.

Discussion

In this section, we will discuss qualitative results of Bi-LM pre-trained NER model on NCBI Disease dataset and Clinical Notes dataset as each of them belongs to a different domain in biomedicine and thus have unique characteristics. NCBI Disease dataset consists of abstracts from medical research papers which are written in a technical language and have a lot of complex entity names while Clinical Notes dataset contains the mention of relatively simple disease entity names which can also be understood by non-medical professionals and it also has fewer disease mentions as compared to NCBI Disease dataset.

In the NCBI Disease dataset, the combined training and development set contains 1902 unique mentions of disease entities. In its test set, there are 423 unique occurrences of disease names and the Bi-LM pre-trained NER model is able to correctly predict 339 such diseases. Some examples of the longer disease names which are hard to recognize but our approach is able to correctly predict them are "*sporadic breast , brain , prostate and kidney cancer*", "*deficiency of the ninth component of human complement*", "*von hippel - lindau (vhl) tumor*" and "*deficiency of the lysosomal enzyme aspartylglucosaminidase*". Among the 423 unique mentions of diseases in test set, 232 of them are unseen in the combined training and development set. Our model was able to correctly predict around 100 unseen disease entities in the test set. Some examples of unseen disease entities which are correctly predicted are "*deficiency of the lysosomal enzyme aspartylglucosaminidase*", "*campomelic - metaphyseal skeletal dysplasia*", "*atrophic benign epidermolysis bullosa*" and "*ectopic intracranial retinoblastoma*". This can be attributed to the improved modeling of the relationship among context words during bidirectional language modeling pre-training step. Some examples of the disease entities where our model fails are "*bannayan - zonana (bzs) or ruvalcaba - riley - smith syndrome*", "*very - long - chain acyl - coenzyme a dehydrogenase deficiency*", "*vwf - deficient*", and "*diffuse mesangial sclerosis*". From the examples, we see that the model fails to correctly predict when the disease entities have longer names which can also contain abbreviations.

On Clinical Notes dataset, our model correctly predicts the tags of 164 unique disease names out of 214 unique names in test set. Some of the examples of correctly predicted disease entities are "*high-grade squamous intraepithelial lesion*", "*diphtheria tetanus pertussis*" and "*chronic obstructive pulmonary disease*". In order to evaluate the generalization performance of our model, we also measure performance in the case of those disease names in the test set which were unseen in training set. Our model was able to correctly predict the tags of 52 out of these 93 new disease names in the test set. Some of the examples of these disease names are "*systolic left ventricular dysfunction*", "*progressive hypoxic respiratory failure*" and "*congenital adrenal hyperplasia*". Some of the cases where our model fails is "*dtap*", "*dvt*", "*protrusion of the posterior vaginal wall*" and "*fractured right clavicle*". We think that some of these disease entities may be incorrectly predicted as they contain short abbreviations without case specific information which are difficult to model. Errors in predictions can also happen due to the lack of similar context information in the training data.

Limitations

Our approach replies on pre-training of weights in order to improve the performance of our model in NER task. During training step, we observe that the performance of our model is sensitive to the value of dropout parameter both on the input to hidden layer and on the output of LSTM hidden layer. In order to achieve maximum performance, one needs to carefully tune the value of both the dropout parameters.

Another limitation of our model is that it uses more than 1000 CNN filters of various sizes in order to compute the character based representation of a word. Use of a large number of CNN weights of different strides drastically increases the number of parameters of the model and hence the overall training and inference time as compared to using CNN filters of same strides.

Although our model performs automatic features extraction and is trained end-to-end, it ignores orthographic features such as the use of capitalization information, punctuations etc. The use of such features along with word embeddings has shown to improve the performance of NER systems[8] in newswire texts. In biomedical domain, we believe that the use of such information can lead to an improvement in overall performance as the medical literature has a lot of abbreviations and complex entity names.

Lastly, we find that the recall of our model in predicting the unseen entities is around 50% which is quite low as compared to the overall recall on various datasets. One possible way to improve the performance on unseen entities is to train deeper and larger neural network models so that they can learn complex information. We also believe that proper linkage of external knowledge sources such as MeSH Database can be used to increase the model performance on unseen test set entities.

Conclusions

In this work, we present an approach for NER system in which we perform pre-training step by transferring the weights of a bidirectional language model to a NER model. The architectures of both the models are similar. Bi-LM is trained in an unsupervised manner using only unlabeled data. We use CNN with different filter strides in order to extract morphological information from characters. Bidirectional LSTM is used for word level sequence modeling and it takes both word embeddings and character CNN features as inputs.

We show that such pre-training step leads to an improvement in F1 score across 3 datasets related to disease NER task. Such Bi-LM pre-training requires less supervised data in order to achieve a particular F1 score as compared to a model whose weights are randomly initialized. Pre-training also leads to a faster convergence during NER model training. We also evaluate the performance of our model on both seen and unseen disease entities during training step. For future work, we aim to incorporate additional features from orthographic information and use external structured knowledge bases in order to further improve model performance.

References

1. Rotmensch, M., Halpern, Y., Tlimat, A., Horng, S. & Sontag, D. Learning a health knowledge graph from electronic medical records. *Sci. Reports* (2017).
2. Florian, R., Ittycheriah, A., Jing, H. & Zhang, T. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (2003).
3. 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 '01 (2001).
4. Settles, B. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications* (Association for Computational Linguistics, 2004).
5. Leaman, R., Gonzalez, G. *et al.* Banner: an executable survey of advances in biomedical named entity recognition. In *Pacific symposium on biocomputing* (2008).
6. Campos, D., Matos, S. & Oliveira, J. L. Gimli: open source and high-performance biomedical name recognition. *BMC Bioinforma.* **14**, 54 (2013).
7. Leaman, R. & Lu, Z. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinforma.* **32** (2016).
8. Collobert, R. *et al.* Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **12**, 2493–2537 (2011).
9. Huang, Z., Xu, W. & Yu, K. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

10. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. & Dyer, C. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT* (2016).

11. Chiu, J. P. & Nichols, E. Named entity recognition with bidirectional lstm-cnns. *Transactions Assoc. for Comput. Linguist.* (2016).

12. Ma, X. & Hovy, E. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Association for Computational Linguistics, 2016).

13. Wei, Q., Chen, T., Xu, R., He, Y. & Gui, L. Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks. *Database* **2016** (2016).

14. Zeng, D., Sun, C., Lin, L. & Liu, B. Lstm-crf for drug-named entity recognition. *Entropy* **19** (2017).

15. Habibi, M., Weber, L., Neves, M., Wiegandt, D. L. & Leser, U. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinforma.* **33**, i37–i48 (2017).

16. Dai, A. M. & Le, Q. V. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, 3079–3087 (2015).

17. Kim, Y., Jernite, Y., Sontag, D. & Rush, A. M. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (AAAI Press, 2016).

18. LeCun, Y. *et al.* Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems 2, NIPS 1989* (1990).

19. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105 (2012).

20. Antol, S. *et al.* Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2425–2433 (2015).

21. Kim, Y. Convolutional neural networks for sentence classification. In *In EMNLP* (Citeseer, 2014).

22. Werbos, P. J. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks* **1**, 339–356 (1988).

23. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9** (1997).

24. Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).

25. Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

26. Doğan, R. I. & Lu, Z. An improved corpus of disease mentions in pubmed citations. In *Proceedings of the 2012 workshop on biomedical natural language processing*, 91–99 (Association for Computational Linguistics, 2012).

27. Li, J. *et al.* Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database* **2016** (2016).

28. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119 (2013).

29. Tsatsaronis, G. *et al.* An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinforma.* **16**, 138 (2015). DOI 10.1186/s12859-015-0564-6.

30. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010).

31. Press, O. & Wolf, L. Using the output embedding to improve language models. *EACL 2017* 157 (2017).

32. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

33. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. machine learning research* **15**, 1929–1958 (2014).

34. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 1310–1318 (2013).

35. Forney, G. D. The viterbi algorithm. *Proc. IEEE* **61**, 268–278 (1973).

Acknowledgements

We would like to thank Mrinmaya Sachan for helpful discussions and Christy Li for providing the Clinical Notes dataset for experiments. This work was supported by a generous funding from MCDS Students Grant.

Author contributions statement

D.S.S., P.X. and E.P.X. conceived and designed the workflow of the study. D.S.S. conducted the experiments. D.S.S. and P.X. performed the analysis. D.S.S. and P.X. wrote the paper. All authors reviewed the manuscript. D.S.S. takes the responsibility of the paper as a whole.

Additional information

Accession codes (where applicable);
Competing financial interests: The authors declare that they have no competing interests.

The corresponding author is responsible for submitting a competing financial interests statement on behalf of all authors of the paper. This statement must be included in the submitted article file.