Bidirectional LSTM-CRF Models for Sequence Tagging

Zhiheng Huang
Baidu research
huangzhiheng@baidu.com

Wei Xu
Baidu research
xuwei06@baidu.com

Kai Yu
Baidu research
yukai@baidu.com

Abstract

In this paper, we propose a variety of Long Short-Term Memory (LSTM) based models for sequence tagging. These models include LSTM networks, bidirectional LSTM (BI-LSTM) networks, LSTM with a Conditional Random Field (CRF) layer (LSTM-CRF) and bidirectional LSTM with a CRF layer (BI-LSTM-CRF). Our work is the first to apply a bidirectional LSTM CRF (denoted as BI-LSTM-CRF) model to NLP benchmark sequence tagging data sets. We show that the BI-LSTM-CRF model can efficiently use both past and future input features thanks to a bidirectional LSTM component. It can also use sentence level tag information thanks to a CRF layer. The BI-LSTM-CRF model can produce state of the art (or close to) accuracy on POS, chunking and NER data sets. In addition, it is robust and has less dependence on word embedding as compared to previous observations.

1 Introduction

Sequence tagging including part of speech tagging (POS), chunking, and named entity recognition (NER) has been a classic NLP task. It has drawn research attention for a few decades. The output of taggers can be used for down streaming applications. For example, a named entity recognizer trained on user search queries can be utilized to identify which spans of text are products, thus triggering certain products ads. Another example is that such tag information can be used by a search engine to find relevant webpages.

Most existing sequence tagging models are linear statistical models which include Hidden Markov Models (HMM), Maximum entropy Markov models (MEMMs) (McCallum et al., 2000), and Conditional Random Fields (CRF) (Lafferty et al., 2001). Convolutional network based models (Collobert et al., 2011) have been recently proposed to tackle sequence tagging problem. We denote such a model as *Conv-CRF* as it consists of a convolutional network and a CRF layer on the output (the term of *sentence level log-likelihood (SSL)* was used in the original paper). The Conv-CRF model has generated promising results on sequence tagging tasks. In speech language understanding community, recurrent neural network (Mesnil et al., 2013; Yao et al., 2014) and convolutional nets (Xu and Sarikaya, 2013) based models have been recently proposed. Other relevant work includes (Graves et al., 2005; Graves et al., 2013) which proposed a bidirectional recurrent neural network for speech recognition.

In this paper, we propose a variety of neural network based models to sequence tagging task. These models include LSTM networks, bidirectional LSTM networks (BI-LSTM), LSTM networks with a CRF layer (LSTM-CRF), and bidirectional LSTM networks with a CRF layer (BI-LSTM-CRF). Our contributions can be summarized as follows. 1) We systematically compare the performance of aforementioned models on NLP tagging data sets; 2) Our work is the first to apply a bidirectional LSTM CRF (denoted as BI-LSTM-CRF) model to NLP benchmark sequence tagging data sets. This model can use both past and future input features thanks to a bidirectional LSTM component. In addition, this model can use sentence level tag information thanks to a CRF layer. Our model can produce state of the art (or close to) accuracy on POS, chunking and NER data sets; 3) We show that BI-LSTM-CRF model is robust and it has less dependence on word embedding as compared to previous observations (Collobert et al., 2011). It can produce accurate tagging performance without resorting to word embedding.

The remainder of the paper is organized as follows. Section 2 describes sequence tagging mod-

els used in this paper. Section 3 shows the training procedure. Section 4 reports the experiments results. Section 5 discusses related research. Finally Section 6 draws conclusions.

2 Models

In this section, we describe the models used in this paper: LSTM, BI-LSTM, CRF, LSTM-CRF and BI-LSTM-CRF.

2.1 LSTM Networks

Recurrent neural networks (RNN) have been employed to produce promising results on a variety of tasks including language model (Mikolov et al., 2010; Mikolov et al., 2011) and speech recognition (Graves et al., 2005). A RNN maintains a memory based on history information, which enables the model to predict the current output conditioned on long distance features.

Figure 1 shows the RNN structure (Elman, 1990) which has an input layer x, hidden layer h and output layer y. In named entity tagging context, x represents input features and y represents tags. Figure 1 illustrates a named entity recognition system in which each word is tagged with *other (O)* or one of four entity types: *Person (PER)*, *Location (LOC)*, *Organization (ORG)*, and *Miscellaneous (MISC)*. The sentence of `EU rejects German call to boycott British lamb .` is tagged as `B-ORG O B-MISC O O O B-MISC O O`, where *B-*, *I-* tags indicate beginning and intermediate positions of entities.

An input layer represents features at time t. They could be one-hot-encoding for word feature, dense vector features, or sparse features. An input layer has the same dimensionality as feature size. An output layer represents a probability distribution over labels at time t. It has the same dimensionality as size of labels. Compared to feedforward network, a RNN introduces the connection between the previous hidden state and current hidden state (and thus the recurrent layer weight parameters). This recurrent layer is designed to store history information. The values in the hidden and output layers are computed as follows:

$$\mathbf{h}(t) = f(\mathbf{U}\mathbf{x}(t) + \mathbf{W}\mathbf{h}(t-1)), \quad (1)$$

$$\mathbf{y}(t) = g(\mathbf{V}\mathbf{h}(t)), \quad (2)$$

where **U**, **W**, and **V** are the connection weights to be computed in training time, and $f(z)$ and $g(z)$ are sigmoid and softmax activation functions as follows.

$$f(z) = \frac{1}{1+e^{-z}}, \quad (3)$$

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}. \quad (4)$$

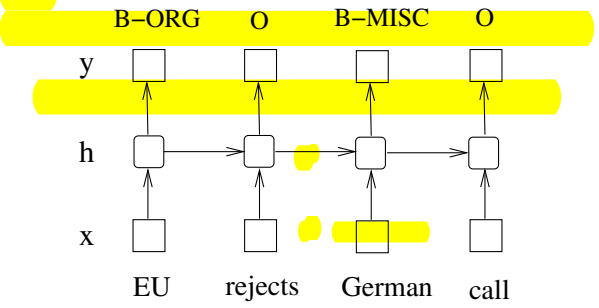

Figure 1: A simple RNN model.

In this paper, we apply Long Short-Term Memory (Hochreiter and Schmidhuber, 1997; Graves et al., 2005) to sequence tagging. Long Short-Term Memory networks are the same as RNNs, except that the hidden layer updates are replaced by purpose-built memory cells. As a result, they may be better at finding and exploiting long range dependencies in the data. Fig. 2 illustrates a single LSTM memory cell (Graves et al., 2005). The

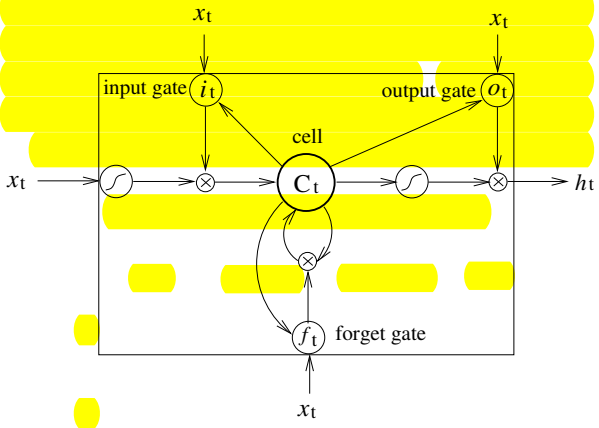

Figure 2: A Long Short-Term Memory Cell.

LSTM memory cell is implemented as the following:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}$$

where σ is the logistic sigmoid function, and i, f, o and c are the input gate, forget gate, output gate and cell vectors, all of which are the same size as the hidden vector h. The weight matrix subscripts have the meaning as the name suggests. For example, W_{hi} is the hidden-input gate matrix, W_{xo} is the input-output gate matrix etc. The weight matrices from the cell to gate vectors (e.g. W_{ci}) are diagonal, so element m in each gate vector only receives input from element m of the cell vector.

Fig. 3 shows a LSTM sequence tagging model which employs aforementioned LSTM memory cells (dashed boxes with rounded corners).

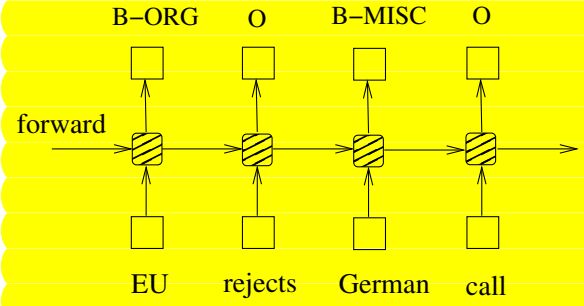

Figure 3: A LSTM network.

2.2 Bidirectional LSTM Networks

In sequence tagging task, we have access to both past and future input features for a given time, we can thus utilize a bidirectional LSTM network (Figure 4) as proposed in (Graves et al., 2013). In doing so, we can efficiently make use of past features (via forward states) and future features (via backward states) for a specific time frame. We train bidirectional LSTM networks using back-propagation through time (BPTT)(Boden., 2002). The forward and backward passes over the unfolded network over time are carried out in a similar way to regular network forward and backward passes, except that we need to unfold the hidden states for all time steps. We also need a special treatment at the beginning and the end of the data points. In our implementation, we do forward and backward for whole sentences and we only need to reset the hidden states to 0 at the begging of each sentence. We have batch implementation which enables multiple sentences to be processed at the same time.

2.3 CRF networks

There are two different ways to make use of neighbor tag information in predicting current tags. The first is to predict a distribution of tags for each time

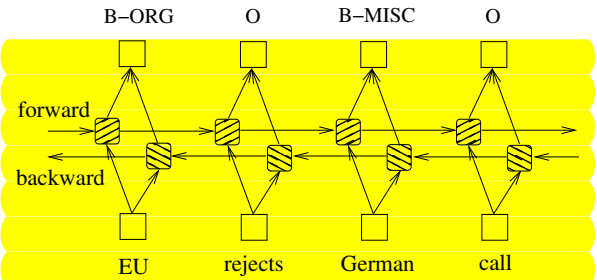

Figure 4: A bidirectional LSTM network.

step and then use beam-like decoding to find optimal tag sequences. The work of maximum entropy classifier (Ratnaparkhi, 1996) and Maximum entropy Markov models (MEMMs) (McCallum et al., 2000) fall in this category. The second one is to focus on sentence level instead of individual positions, thus leading to Conditional Random Fields (CRF) models (Lafferty et al., 2001) (Fig. 5). Note that the inputs and outputs are directly connected, as opposed to LSTM and bidirectional LSTM networks where memory cells/recurrent components are employed.

It has been shown that CRFs can produce higher tagging accuracy in general. It is interesting that the relation between these two ways of using tag information bears resemblance to two ways of using input features (see aforementioned LSTM and BI-LSTM networks), and the results in this paper confirms the superiority of BI-LSTM compared to LSTM.

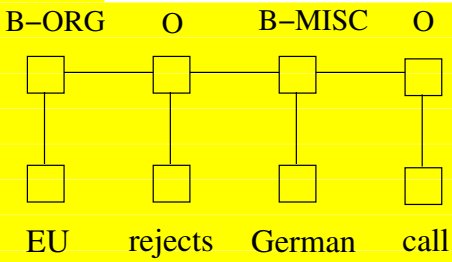

Figure 5: A CRF network.

2.4 LSTM-CRF networks

We combine a LSTM network and a CRF network to form a LSTM-CRF model, which is shown in Fig. 6. This network can efficiently use past input features via a LSTM layer and sentence level tag information via a CRF layer. A CRF layer is represented by lines which connect consecutive output layers. A CRF layer has a state transition matrix as parameters. With such a layer, we can efficiently use past and future tags to predict the current tag,

which is similar to the use of past and future input features via a bidirectional LSTM network. We consider the matrix of scores $f_\theta([x]_1^T)$ are output by the network. We drop the input $[x]_1^T$ for notation simplification. The element $[f_\theta]_{i,t}$ of the matrix is the score output by the network with parameters θ, for the sentence $[x]_1^T$ and for the i-th tag, at the t-th word. We introduce a transition score $[A]_{i,j}$ to model the transition from i-th state to j-th for a pair of consecutive time steps. Note that this transition matrix is position independent. We now denote the new parameters for our network as $\tilde{\theta} = \theta \cup \{[A]_{i,j} \forall i, j\}$. The score of a sentence $[x]_1^T$ along with a path of tags $[i]_1^T$ is then given by the sum of transition scores and network scores:

$$s([x]_1^T, [i]_1^T, \tilde{\theta}) = \sum_{t=1}^{T} ([A]_{[i]_{t-1},[i]_t} + [f_\theta]_{[i]_t,t}). \quad (5)$$

The dynamic programming (Rabiner, 1989) can be used efficiently to compute $[A]_{i,j}$ and optimal tag sequences for inference. See (Lafferty et al., 2001) for details.

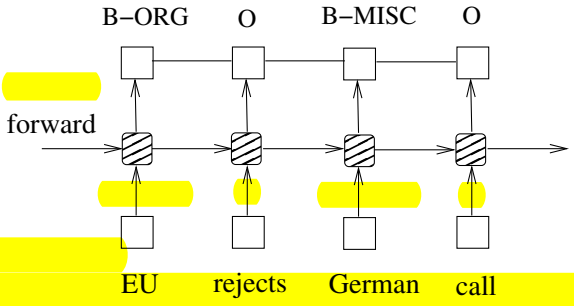

Figure 6: A LSTM-CRF model.

2.5 BI-LSTM-CRF networks

Similar to a LSTM-CRF network, we combine a bidirectional LSTM network and a CRF network to form a BI-LSTM-CRF network (Fig. 7). In addition to the past input features and sentence level tag information used in a LSTM-CRF model, a BI-LSTM-CRF model can use the future input features. The extra features can boost tagging accuracy as we will show in experiments.

3 Training procedure

All models used in this paper share a generic SGD forward and backward training procedure. We choose the most complicated model, BI-LSTM-CRF, to illustrate the training algorithm as shown in Algorithm 1. In each epoch, we divide the whole training data to batches and process one batch at a time. Each batch contains a list of sentences which is determined by the parameter of *batch size*. In our experiments, we use batch size of 100 which means to include sentences whose total length is no greater than 100. For each batch, we first run bidirectional LSTM-CRF model forward pass which includes the forward pass for both forward state and backward state of LSTM. As a result, we get the the output score $f_\theta([x]_1^T)$ for all tags at all positions. We then run CRF layer forward and backward pass to compute gradients for network output and state transition edges. After that, we can back propagate the errors from the output to the input, which includes the backward pass for both forward and backward states of LSTM. Finally we update the network parameters which include the state transition matrix $[A]_{i,j} \forall i, j$, and the original bidirectional LSTM parameters θ.

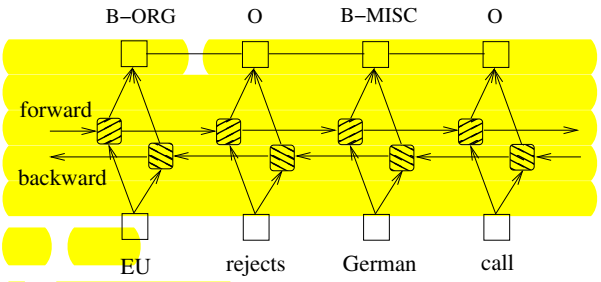

Figure 7: A BI-LSTM-CRF model.

Algorithm 1 Bidirectional LSTM CRF model training procedure

```
for each epoch do
    for each batch do
        1) bidirectional LSTM-CRF model forward pass:
            forward pass for forward state LSTM
            forward pass for backward state LSTM
        2) CRF layer forward and backward pass
        3) bidirectional LSTM-CRF model backward pass:
            backward pass for forward state LSTM
            backward pass for backward state LSTM
        4) update parameters
    end for
end for
```

4 Experiments

4.1 Data

We test LSTM, BI-LSTM, CRF, LSTM-CRF, and BI-LSTM-CRF models on three NLP tagging tasks: Penn TreeBank (PTB) POS tagging, CoNLL 2000 chunking, and CoNLL 2003 named entity tagging. Table 1 shows the size of sen-

tences, tokens, and labels for training, validation and test sets respectively.

POS assigns each word with a unique tag that indicates its syntactic role. In chunking, each word is tagged with its phrase type. For example, tag *B-NP* indicates a word starting a noun phrase. In NER task, each word is tagged with *other* or one of four entity types: *Person*, *Location*, *Organization*, or *Miscellaneous*. We use the `BIO2` annotation standard for chunking and NER tasks.

4.2 Features

We extract the same types of features for three data sets. The features can be grouped as spelling features and context features. As a result, we have 401K, 76K, and 341K features extracted for POS, chunking and NER data sets respectively. These features are similar to the features extracted from Stanford NER tool (Finkel et al., 2005; Wang and Manning, 2013). Note that we did not use extra data for POS and chunking tasks, with the exception of using Senna embedding (see Section 4.2.3). For NER task, we report performance with spelling and context features, and also incrementally with Senna embedding and Gazetteer features[1].

4.2.1 Spelling features

We extract the following features for a given word in addition to the lower case word features.

- whether start with a capital letter
- whether has all capital letters
- whether has all lower case letters
- whether has non initial capital letters
- whether mix with letters and digits
- whether has punctuation
- letter prefixes and suffixes (with window size of 2 to 5)
- whether has apostrophe end ('s)
- letters only, for example, I. B. M. to IBM
- non-letters only, for example, A. T. &T. to ..&
- word pattern feature, with capital letters, lower case letters, and digits mapped to 'A', 'a' and '0' respectively, for example, D56y-3 to A00a-0
- word pattern summarization feature, similar to word pattern feature but with consecutive identical characters removed. For example, D56y-3 to A0a-0

[1]Downloaded from http://ronan.collobert.com/senna/

4.2.2 Context features

For word features in three data sets, we use unigram features and bi-grams features. For POS features in CoNLL2000 data set and POS & CHUNK features in CoNLL2003 data set, we use unigram, bi-gram and tri-gram features.

4.2.3 Word embedding

It has been shown in (Collobert et al., 2011) that word embedding plays a vital role to improve sequence tagging performance. We downloaded[2] the embedding which has 130K vocabulary size and each word corresponds to a 50-dimensional embedding vector. To use this embedding, we simply replace the one hot encoding word representation with its corresponding 50-dimensional vector.

4.2.4 Features connection tricks

We can treat spelling and context features the same as word features. That is, the inputs of networks include both word, spelling and context features. However, we find that direct connections from spelling and context features to outputs accelerate training and they result in very similar tagging accuracy. Fig. 8 illustrates this network in which features have direct connections to outputs of networks. We will report all tagging accuracy using this connection. We note that this usage of features has the same flavor of Maximum Entropy features as used in (Mikolov et al., 2011). The difference is that features collision may occur in (Mikolov et al., 2011) as feature hashing technique has been adopted. Since the output labels in sequence tagging data sets are less than that of language model (usually hundreds of thousands), we can afford to have full connections between features and outputs to avoid potential feature collisions.

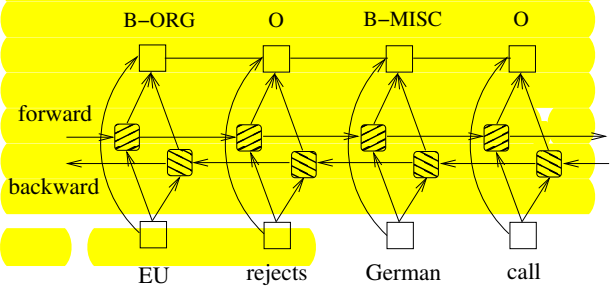

Figure 8: A BI-LSTM-CRF model with MaxEnt features.

[2]http://ronan.collobert.com/senna/

Table 1: Size of sentences, tokens, and labels for training, validation and test sets.

		POS	CoNLL2000	CoNLL2003
training	sentence #	39831	8936	14987
	token #	950011	211727	204567
validation	sentence #	1699	N/A	3466
	token #	40068	N/A	51578
test	sentences #	2415	2012	3684
	token #	56671	47377	46666
	label #	45	22	9

4.3 Results

We train LSTM, BI-LSTM, CRF, LSTM-CRF and BI-LSTM-CRF models for each data set. We have two ways to initialize word embedding: *Random* and *Senna*. We randomly initialize the word embedding vectors in the first category, and use Senna word embedding in the second category. For each category, we use identical feature sets, thus different results are solely due to different networks. We train models using training data and monitor performance on validation data. As chunking data do not have a validation data set, we use part of training data for validation purpose.

We use a learning rate of 0.1 to train models. We set hidden layer size to 300 and found that model performance is not sensitive to hidden layer sizes. The training for three tasks require less than 10 epochs to converge and it in general takes less than a few hours. We report models' performance on test datasets in Table 2, which also lists the best results in (Collobert et al., 2011), denoted as Conv-CRF. The POS task is evaluated by computing per-word accuracy, while the chunk and NER tasks are evaluated by computing F1 scores over chunks.

4.3.1 Comparison with Cov-CRF networks

We have three baselines: LSTM, BI-LSTM and CRF. LSTM is the weakest baseline for all three data sets. The BI-LSTM performs close to CRF on POS and chunking datasets, but is worse than CRF on NER data set. The CRF forms strong baselines in our experiments. For random category, CRF models outperform Conv-CRF models for all three data sets. For Senna category, CRFs outperform Conv-CRF for POS task, while underperform for chunking and NER task. LSTM-CRF models outperform CRF models for all data sets in both random and Senna categories. This shows the effectiveness of the forward state LSTM component in modeling sequence data. The BI-LSTM-CRF models further improve LSTM-CRF models and they lead to the best tagging performance for all cases except for POS data at random category, in which LSTM-CRF model is the winner. The numbers in parentheses for CoNLL 2003 under Senna categories are generated with Gazetteer features.

It is interesting that our best model BI-LSTM-CRF has less dependence on Senna word embedding compared to Conv-CRF model. For example, the tagging difference between BI-LSTM-CRF model for random and Senna categories are 0.12%, 0.33%, and 4.57% for POS, chunking and NER data sets respectively. In contrast, the Conv-CRF model heavily relies on Senna embedding to get good tagging accuracy. It has the tagging difference of 0.92%, 3.99% and 7.20% between random and Senna category for POS, chunking and NER data sets respectively.

4.3.2 Model robustness

To estimate the robustness of models with respect to engineered features (spelling and context features), we train LSTM, BI-LSTM, CRF, LSTM-CRF, and BI-LSTM-CRF models with word features only (spelling and context features removed). Table 3 shows tagging performance of proposed models for POS, chunking, and NER data sets using Senna word embedding. The numbers in parentheses indicate the performance degradation compared to the same models but using spelling and context features. CRF models' performance is significantly degraded with the removal of spelling and context features. This reveals the fact that CRF models heavily rely on engineered features to obtain good performance. On the other hand, LSTM based models, especially BI-LSTM and BI-LSTM-CRF models are more robust and they are less affected by the removal of engineering features. For all three tasks, BI-LSTM-CRF models result in the highest tagging accuracy. For example, It achieves the F1 score of 94.40 for CoNLL2000 chunking, with slight degradation

Table 2: Comparison of tagging performance on POS, chunking and NER tasks for various models.

		POS	CoNLL2000	CoNLL2003
Random	Conv-CRF (Collobert et al., 2011)	96.37	90.33	81.47
	LSTM	97.10	92.88	79.82
	BI-LSTM	97.30	93.64	81.11
	CRF	97.30	93.69	83.02
	LSTM-CRF	**97.45**	93.80	84.10
	BI-LSTM-CRF	97.43	**94.13**	**84.26**
Senna	Conv-CRF (Collobert et al., 2011)	97.29	94.32	88.67 (89.59)
	LSTM	97.29	92.99	83.74
	BI-LSTM	97.40	93.92	85.17
	CRF	97.45	93.83	86.13
	LSTM-CRF	97.54	94.27	88.36
	BI-LSTM-CRF	**97.55**	**94.46**	**88.83** (**90.10**)

(0.06) compared to the same model but using spelling and context features.

4.3.3 Comparison with existing systems

For POS data set, we achieved state of the art tagging accuracy with or without the use of extra data resource. POS data set has been extensively tested and the past improvement can be realized in Table 4. Our test accuracy is 97.55% which is significantly better than others in the confidence level of 95%. In addition, our BI-LSTM-CRF model already reaches a good accuracy without the use of the Senna embedding.

All chunking systems performance is shown in table 5. Kudo et al. won the CoNLL 2000 challenge with a F1 score of 93.48%. Their approach was a SVM based classifier. They later improved the results up to 93.91%. Recent work include the CRF based models (Sha and Pereira, 2003; Mcdonald et al., 2005; Sun et al., 2008). More recent is (Shen and Sarkar, 2005) which obtained 95.23% accuracy with a voting classifier scheme, where each classifier is trained on different tag representations (`IOB`, `IOE`, etc.). Our model outperforms all reported systems except (Shen and Sarkar, 2005).

The performance of all systems for NER is shown in table 6. (Florian et al., 2003) presented the best system at the NER CoNLL 2003 challenge, with 88.76% F1 score. They used a combination of various machine-learning classifiers. The second best performer of CoNLL 2003 (Chieu., 2003) was 88.31% F1, also with the help of an external gazetteer. Later, (Ando and Zhang., 2005) reached 89.31% F1 with a semi-supervised approach. The best F1 score of 90.90% was reported in (Passos et al., 2014) which employed a new form of learning word embeddings that can leverage information from relevant lexicons to improve the representations. Our model can achieve the best F1 score of 90.10 with both Senna embedding and gazetteer features. It has a lower F1 score than (Passos et al., 2014) , which may be due to the fact that different word embeddings were employed. With the same Senna embedding, BI-LSTM-CRF slightly outperforms Conv-CRF (90.10% vs. 89.59%). However, BI-LSTM-CRF significantly outperforms Conv-CRF (84.26% vs. 81.47%) if random embedding is used.

5 Discussions

Our work is close to the work of (Collobert et al., 2011) as both of them utilized deep neural networks for sequence tagging. While their work used convolutional neural networks, ours used bidirectional LSTM networks.

Our work is also close to the work of (Hammerton, 2003; Yao et al., 2014) as all of them employed LSTM network for tagging. The performance in (Hammerton, 2003) was not impressive. The work in (Yao et al., 2014) did not make use of bidirectional LSTM and CRF layers and thus the tagging accuracy may be suffered.

Finally, our work is related to the work of (Wang and Manning, 2013) which concluded that non-linear architecture offers no benefits in a high-dimensional discrete feature space. We showed that with the bi-directional LSTM CRF model, we consistently obtained better tagging accuracy than a single CRF model with identical feature sets.

6 Conclusions

In this paper, we systematically compared the performance of LSTM networks based models for se-

Table 3: Tagging performance on POS, chunking and NER tasks with only word features.

		POS	CoNLL2000	CoNLL2003
Senna	LSTM	94.63 (-2.66)	90.11 (-2.88)	75.31 (-8.43)
	BI-LSTM	96.04 (-1.36)	93.80 (-0.12)	83.52 (-1.65)
	CRF	94.23 (-3.22)	85.34 (-8.49)	77.41 (-8.72)
	LSTM-CRF	95.62 (-1.92)	93.13 (-1.14)	81.45 (-6.91)
	BI-LSTM-CRF	**96.11** (-1.44)	**94.40** (-0.06)	**84.74** (-4.09)

Table 4: Comparison of tagging accuracy of different models for POS.

System	accuracy	extra data
Maximum entropy cyclic dependency network (Toutanova et al., 2003)	97.24	No
SVM-based tagger (Gimenez and Marquez, 2004)	97.16	No
Bidirectional perceptron learning (Shen et al., 2007)	97.33	No
Semi-supervised condensed nearest neighbor (Soegaard, 2011)	97.50	Yes
CRFs with structure regularization (Sun, 2014)	97.36	No
Conv network tagger (Collobert et al., 2011)	96.37	No
Conv network tagger (senna) (Collobert et al., 2011)	97.29	Yes
BI-LSTM-CRF (ours)	**97.43**	No
BI-LSTM-CRF (Senna) (ours)	**97.55**	Yes

Table 5: Comparison of F1 scores of different models for chunking.

System	accuracy
SVM classifier (Kudo and Matsumoto, 2000)	93.48
SVM classifier (Kudo and Matsumoto, 2001)	93.91
Second order CRF (Sha and Pereira, 2003)	94.30
Specialized HMM + voting scheme (Shen and Sarkar, 2005)	**95.23**
Second order CRF (Mcdonald et al., 2005)	94.29
Second order CRF (Sun et al., 2008)	94.34
Conv-CRF (Collobert et al., 2011)	90.33
Conv network tagger (senna) (Collobert et al., 2011)	94.32
BI-LSTM-CRF (ours)	94.13
BI-LSTM-CRF (Senna) (ours)	94.46

Table 6: Comparison of F1 scores of different models for NER.

System	accuracy
Combination of HMM, Maxent etc. (Florian et al., 2003)	88.76
MaxEnt classifier (Chieu., 2003)	88.31
Semi-supervised model combination (Ando and Zhang., 2005)	89.31
Conv-CRF (Collobert et al., 2011)	81.47
Conv-CRF (Senna + Gazetteer) (Collobert et al., 2011)	89.59
CRF with Lexicon Infused Embeddings (Passos et al., 2014)	**90.90**
BI-LSTM-CRF (ours)	84.26
BI-LSTM-CRF (Senna + Gazetteer) (ours)	90.10

quence tagging. We presented the first work of applying a BI-LSTM-CRF model to NLP benchmark sequence tagging data. Our model can produce state of the art (or close to) accuracy on POS, chunking and NER data sets. In addition, our model is robust and it has less dependence on word embedding as compared to the observation in (Collobert et al., 2011). It can achieve accurate tagging accuracy without resorting to word embedding.

References

[Ando and Zhang.2005] R. K. Ando and T. Zhang. 2005. *A framework for learning predictive structures from multiple tasks and unlabeled data*. Journal of Machine Learning Research (JMLR).

[Boden.2002] M. Boden. 2002. *A Guide to Recurrent Neural Networks and Back-propagation*. In the Dallas project.

[Chieu.2003] H. L. Chieu. 2003. *Named entity recognition with a maximum entropy approach*. Proceedings of CoNLL.

[Collobert et al.2011] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. 2011. *Natural Language Processing (Almost) from Scratch*. Journal of Machine Learning Research (JMLR).

[Elman1990] J. L. Elman. 1990. *Finding structure in time*. Cognitive Science.

[Finkel et al.2005] J. R. Finkel, T. Grenager, and C. Manning. 2005. *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling*. Proceedings of ACL.

[Florian et al.2003] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. *Named entity recognition through classifier combination*. Proceedings of NAACL-HLT.

[Gimenez and Marquez2004] J. Gimenez and L. Marquez. 2004. *SVMTool: A general POS tagger generator based on support vector machines*. Proceedings of LREC.

[Graves et al.2005] A. Graves and J. Schmidhuber. 2005. *Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures*. Neural Networks.

[Graves et al.2013] A. Graves, A. Mohamed, and G. Hinton. 2013. *Speech Recognition with Deep Recurrent Neural Networks*. arxiv.

[Hammerton2003] J. Hammerton. 2003. *Named Entity Recognition with Long Short-Term Memory*. Proceedings of HLT-NAACL.

[Hochreiter and Schmidhuber1997] S. Hochreiter and J. Schmidhuber. 1997. *Long short-term memory*. Neural Computation, 9(8):1735-1780.

[Kudo and Matsumoto2000] T. Kudo and Y. Matsumoto. 2000. *Use of support vector learning for chunk identification*. Proceedings of CoNLL.

[Kudo and Matsumoto2001] T. Kudo and Y. Matsumoto. 2001. *Chunking with support vector machines*. Proceedings of NAACL-HLT.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. Proceedings of ICML.

[McCallum et al.2000] A. McCallum, D. Freitag, and F. Pereira. 2000. *Maximum entropy Markov models for information extraction and segmentation*. Proceedings of ICML.

[Mcdonald et al.2005] R. Mcdonald , K. Crammer , and F. Pereira. 2005. *Flexible text segmentation with structured multilabel classification*. Proceedings of HLT-EMNLP.

[Mesnil et al.2013] G. Mesnil, X. He, L. Deng, and Y. Bengio. 2013. *Investigation of recurrent-neural-network architectures and learning methods for language understanding*. Proceedings of INTERSPEECH.

[Mikolov et al.2010] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur. 2010. *Recurrent neural network based language model*. INTERSPEECH.

[Mikolov et al.2011] T. Mikolov, A. Deoras, D. Povey, L. Burget, J. Eernocky. 2011. *Strategies for Training Large Scale Neural Network Language Models*. Proceedings of ASRU.

[Mikolov et al.2013] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. *Distributed Representations of Words and Phrases and their Compositionality*. Proceedings of NIPS.

[Passos et al.2014] A. Passos, V. Kumar, and A. McCallum. 2014. *Lexicon Infused Phrase Embeddings for Named Entity Resolution*. Proceedings of CoNLL.

[Rabiner1989] L. R. Rabiner. 1989. *A tutorial on hidden Markov models and selected applications in speech recognition*. Proceedings of the IEEE.

[Ratnaparkhi1996] A. Ratnaparkhi. 1996. *A maximum entropy model for part-of-speech tagging*. Proceedings of EMNLP.

[Sha and Pereira2003] F. Sha and F. Pereira. 2003. *Shallow parsing with conditional random fields*. Proceedings of NAACL.

[Shen et al.2007] L. Shen, G. Sara, and A. K. Joshi. 2007. *Guided learning for bidirectional sequence classification*. Proceedings of ACL.

[Shen and Sarkar2005] H. Shen and A. Sarkar. 2005. *Voting between multiple data representations for text chunking*. Canadian AI.

[Soegaard2011] A. Soegaard. 2011. *Semi-supervised condensed nearest neighbor for part-of-speech tagging*. Proceedings of ACL-HLT.

[Sun2014] X. Sun. 2014. *Structure Regularization for Structured Prediction*. Proceedings of NIPS.

[Sun et al.2008] X. Sun, L.P. Morency, D. OKanohara and J. Tsujii. 2008. *Modeling Latent-Dynamic in Shallow Parsing: A Latent Conditional Model with Improved Inference*. Proceedings of COLING.

[Toutanova et al.2003] K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*. Proceedings of HLT-NAACL.

[Wang and Manning2013] M. Wang and C. D. Manning. 2013. *Effect of Non-linear Deep Architecture in Sequence Labeling*. Proceedings of IJCNLP.

[Xu and Sarikaya2013] P. Xu and R. Sarikaya. 2013. *Convolutional neural network based triangular CRF for joint intent detection and slot filling*. Proceedings of ASRU.

[Yao et al.2014] K. S. Yao, B. L. Peng, G. Zweig, D. Yu, X. L. Li, and F. Gao. 2014. *Recurrent conditional random fields for language understanding*. ICASSP.

[Yao et al.2014] K. S. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi. 2014. *Spoken Language Understanding using Long Short-Term Memory Neural Networks*. IEEE SLT.