Bi-Directional Decoder Model with Efficient Fine-Tuning of Embedding for Named Entity Recognition

Panuwat Assawinjaipetch
Japan Advanced Institute of Science and Technology
a.panuwat@jaist.ac.jp

Kiyoaki Shirai
Japan Advanced Institute of Science and Technology
kshirai@jaist.ac.jp

Virach Sornlertlamvanich
Sirindhorn International Institute of Technology
virach@siit.tu.ac.th

Sanparith Marukatat
Thailand's National Electronics and Computer Technology Center
sanparith.marukatat@nectec.or.th

Abstract

Named Entity Recognition (NER) is one of the important tasks in natural language processing. In this paper, we propose a novel method for NER in Japanese. It consists of three deep learning modules: a Character Encoder, Word Encoder and Tag Decoder, which are implemented by Long Short-Term Memory (LSTM) or Bi-Directional LSTM (BiLSTM). Pre-trained character and word embeddings are used as the input of our model. Our new idea is to combine a forward and backward LSTM at Tag Decoder. This enables us to consider named entity (NE) tags of both the previous and succeeding words in the classification, while only a previous NE tag was taken into account in most of the past studies. We also introduce a separate fine-tuning of the embedding that enables us to efficiently fine-tune the parameters of the character and word embeddings. In this method, the parameters of embeddings and other model parameters are trained separately. In an experiment using a large Japanese named entity tagged corpus, the F1-score of our proposed method was 0.944, which was better than the baseline by 0.06 points.

1 Introduction

Named Entities Recognition (NER) is the task of identifying named entities, such as person, location, organization, and so on, in a text. An NER system is often used as a core system in various types of Natural Language Processing (NLP) including question answering, information retrieval, dialogue system, topic modeling, etc. In modern research, NER systems are implemented as classifiers using abstract representation of a sentence as features, where an abstract representation is obtained by deep learning architectures.

NER is usually defined as a sequential labeling problem. Regarding a word sequence of a given sentence as a time sequence, named entity (NE) tags for words are determined one by one from the first to last words. In order to classify an NE tag of a word in a certain time step, it is common to use an NE tag determined in the previous time step as a feature for classification. In this research, we propose a novel method to use the NE tags of not only the previous but also the succeeding words in a deep learning model.

On the other hand, fine-tuning of the word embedding is widely applied in deep learning models for NLP. That is, the word embedding is pre-trained using a huge amount of texts in a general domain, then the parameters of the word embedding are updated using a relatively small amount of data that is specific to the target domain. Another contribution of this paper is to propose a novel method for fine-tuning of the word and character embeddings. It performs parameter estimation with a deep neural network and fine-tuning of embeddings separately.

2 Related work

2.1 Natural language processing with deep learning

Recently, deep learning has been actively studied in the NLP research field. Collobert et al. (2011) intro-

duced a neural network model for NLP. Their system used minimal feature engineering but achieved promising results. However, the proposed feed forward network could not capture relations between the words in a sentence, although they can be a useful feature for various NLP tasks. Recurrent Neural Network (RNN) has been proposed, which is well-known for its ability to detect hidden relationship between words. Later, RNN and its variants have been applied for many NLP applications. RNN can be generally classified into three types. The first one is the traditional RNN called the Hopfield network, proposed by Hopfield (1982). The second one is the most popular network, called Long Short-Term Memory (LSTM), proposed by Hochreiter and Schmidhuber (1997). It was also the first network that tried to mitigate the vanishing gradient problem. The third one is Gate Recurrent Unit (GRU) proposed by Cho et al. (2014). Since the second and third networks were less sensitive to the problem of vanishing gradient, most modern research tends to use them rather than the traditional RNN. However, it is still uncertain which is better, LSTM or GRU. Finally, models that combined neural network in forward and backward directions, such as bi-directional LSTM (BiLSTM) (Graves and Schmidhuber, 2005), achieved further improvement due to their ability to capture left and right contexts.

2.2 Deep neural network for named entity recognition

The modern neural architectures for NER can be broadly classified into categories according to their representation of an input sentence. The representation can be based on words, characters, and other features such as affix *n*-gram, as well as combinations of these.

2.2.1 NER based on word embedding

In the usual deep neural networks for NER, a sequence of words in a sentence is given as an input. Usually, each word is represented by a word embedding and given to the neural networks. Collobert et al. (2011) first introduced a convolutional neural network that accepted a word sequence as an input. Then, several methods of RNN that handled a sequence of words were proposed (Mesnil et al., 2013; Nguyen et al., 2016). Huang et al. (2015) proposed LSTM with Conditional Random Field (CRF) (Lafferty et al., 2001) that achieved an 84.26% F1 score on the CoNLL-2003 English data set (Tjong Kim Sang and De Meulder, 2003). By slightly modifying this model, Shao et al. (2016) proposed a window-based bi-directional LSTM for NER. Neural network models for NER on specific domains (e.g. the medical domain) have also been investigated (Chalapathy et al., 2016; Xu et al., 2018).

2.2.2 NER based on character embeddings

Each sentence is taken to be a sequence of characters in several previous methods. Each character is converted into a vector representation by character embedding. The potential of the character NER neural model was first highlighted by Kim et al. (2016). The character based architecture has the ability to tackle an out-of-vocabulary problem and can improve the performance of NER in morphologically rich languages. The architecture was applied to various languages such as Vietnamese (Pham and Phuong, 2017) and Chinese (Dong et al., 2016). Kuru et al. (2016) applied a character based model to 7 different languages.

2.2.3 NER based on word and character embeddings

Several studies have proven that the incorporation of both word and character sequences in a neural network model can contribute to develop a strong NER system. Ma and Hovy (2016) and Chiu and Nichols (2016) proposed such models and achieved 91.21% and 91.62% F1-score on the CoNLL-2003 English data set, respectively. Misawa et al. (2017) proposed a model using word and character emdeddings for Japanese NER. Lample et al. (2016) and Yang et al. (2016) applied BiLSTM and GRU for feature extraction at both the word and character levels.

2.3 Output layer in neural based NER models

Most neural network models for NER consist of two parts. One is constituted by the layers to obtain an abstract representation of an input from word and/or character embeddings. The other is constituted by the layers that determine an NER tag for each word based on the abstract representation. Hereafter, we call the latter the *hidden2tag* layer. CRF has been commonly used in the *hidden2tag* layer. However,

other networks are also used. For example, Shen et al. (2018) and Mesnil et al. (2013) proposed methods to use RNN and feed-forward network in the *hidden2tag* layer.

However, to the best of our knowledge, no bi-directional RNN or LSTM has been used in the *hidden2tag* layer. As discussed in Section 1, NER is usually regarded as a sequential labeling problem, where the NER tags of words are determined one by one. Furthermore, the NE tag of the previous word is commonly used in the classification of the NE tag for the current word. However, the NE tag of the succeeding word cannot be used as a feature, since it is not determined yet in a sequential labeling. If the NE tags are determined in a backward direction (from the last to first word), the succeeding NE tag can be used, but the previous NE tag cannot. Therefore, the bi-directional RNN or LSTM cannot be applicable to the *hidden2tag* layer. Intuitively, both the NER tags of the previous and succeeding words are effective for NER. This paper proposes a way to use both of them.

Mesnil et al. (2013) proposed a model called bi-directional Jordan-type RNN for the slot filling task in spoken language. Their model also considered both the previous and succeeding output tags, but at time t only the output tags in the near words from $t-T$ to $t+T$ were used. Our model combines the ordinary forward and backward LSTM that can take long dependencies into account.

3 Proposed method

3.1 Task

We define classes of named entities following Sekine's extended named entity hierarchy (version 7.1.0)(Sekine and Nobata, 2004)[1], which consists of 200 fine grained named entity classes. Since the number of NE classes is large, coarse grained NE types in the hierarchy are used. Table 1 shows a list of the 26 NE classes in our NER task. An extended named entity annotated corpus in Japanese (Hasimoto et al., 2008) was used to develop our method.[2] In the corpus, named entities are annotated with IOB encoding, where the NE classes are those in Sekine's extended named entity hierarchy. The corpus consists of 8,228 articles, 53,224 distinct words, and 2,226,147 tokens. It is about 7.4 times larger than the CoNLL-2003 English corpus, which consists of 1,393 articles and 301,418 tokens. As preprocessing, we used the tool CaboCha (Taku Kudo, 2002) for word segmentation, POS tagging, and chunking.

Table 1: List of named entity classes

God, Percent, Location, Latitude Longitude, Product, Ordinal Number, Name Other, Numex Other, Multiplication, School Age, Timex, Age, Natural Object, Disease, Person, Organization, Facility, Colour, Money, Point, Rank, Countx, Frequency, Measurement, Event, Period

The graphical notation of the task definition is illustrated in Figure 1. An input of our model is a sentence represented as a sequence of words $\{w_1 \cdots w_n\}$. The number of the words for each sentence is fixed at n: padding is used when the length of a sentence is less than n. A sentence is also represented as a sequence of characters $\{c_1 \cdots c_m\}$. Similarly, the number of the characters in a sentence, denoted by m, is also fixed. Following the IOB encoding of named entities, the model predicts an output tag t_i for each word w_i. t_i is represented by B_x, I_x or O, where x stands for a type of a name entity (e.g. "Organization", "Person").

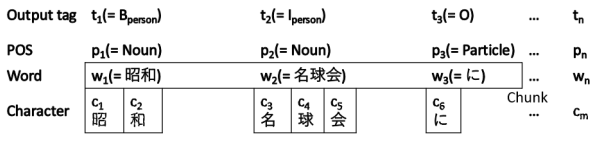

Figure 1: Task definition

3.2 Model

Our proposed model is based on Shen's architecture (Shen et al., 2018) , which achieved a 90.89% F1-score on the CoNLL-2013 dataset. An overview of our model is shown in Figure 2. It consists of three modules: Character Encoder, Word Encoder and Tag Decoder. Tag Decoder corresponds to the *hidden2tag* layer.

[1] https://nlp.cs.nyu.edu/ene/

[2] Although the corpus includes newspaper articles and white papers, only news texts were used in this study.

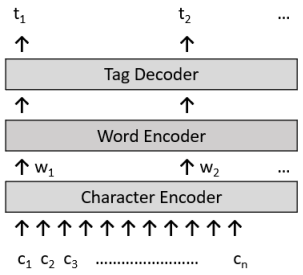

Figure 2: Overview of NER model

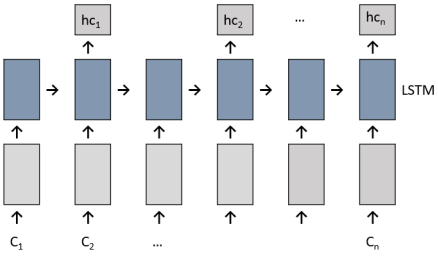

Figure 3: Character Encoder

3.2.1 Character Encoder

Character Encoder produces an abstract representation of a word from a sequence of characters. The motivation of introducing this module is that some Japanese characters indicate types of named entities. For example, the character c_5 "会(*kai*)" in Figure 1 literally means "association". A proper noun including it tends to be classified as an organization. Similarly, a proper noun including the character "岳(*gaku*)", which means "mountain", tends to be a location.

The architecture of Character Encoder is shown in Figure 3. Character embedding is entered as an input of Character Encoder. Character embedding is pre-trained from the training corpus by the skip-gram model implemented by the *word2vec* tool (Mikolov et al., 2013). In our preliminary experiment, it was found that a single direction LSTM was slightly better than BiLSTM. Thus, we apply a single direction LSTM, while Shen et al. (2018) used BiLSTM in Character Encoder. The hidden state of the last character of each word is passed to the next module. The output of Character Encoder applied to the *i*th word is denoted by hc_i.

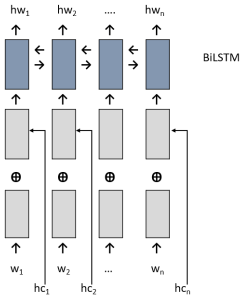

Figure 4: Word Encoder

3.2.2 Word Encoder

Word Encoder produces contextual information of the words in a sentence. The architecture of Word Encoder is shown in Figure 4. The word embedding of w_i and the output of Character Encoder hc_i are concatenated, then this is passed to the BiLSTM model. Finally, the hidden state of each word hw_i is obtained as the output of this module. It is almost the same as Shen's original model (Shen et al., 2018) except that CNN was used in their model. In addition, the word embedding is pre-trained from the training data by the skip-gram model using the *word2vec* tool.

3.2.3 Tag Decoder

For each word w_i, Tag Decoder predicts the output vector (denoted by o_i) that represents the distribution of the scores of the output tags. The architecture of Tag Decoder is shown in Figure 5. The output of Word Encoder hw_i and the previous output vector o_{i-1} are concatenated and passed to LSTM. The output of LSTM (hd_i) is augmented by two additional features. One is the POS embedding pe_i that represents the information of the POS (p_i) of each word. The other is the Japanese particle embedding jp that represents the syntactic information of the chunk.[3] Since we believe that both POS and the Japanese particle are effective for NER, the hw_i are concatenated with pe_i and jp. Then, they are entered to a

[3]"Particle" is one of the POSs and represents a case maker in Japanese. It plays an important grammatical role, especially in determining the type of a chunk. For example, a chunk "noun + *ga*" represents a nominative case of a predicate, while "noun + *ni*" represents a dative case ("*ga*" and "*ni*" are Japanese particles). For each chunk, the particle that appears in the rightmost position (denoted by JP in Figure 5) is identified, which determines the grammatical role of the chunk. Then the embedding of JP, denoted by jp, is added to hd_i of all the words in the chunk.

feed forward network (FFN) to determine the output vector o_i. As for the final result, a single NE tag t_i for each word is determined by the index of the highest value in the output vector o_i.

3.2.4 Two directional Tag Decoder

We propose a new tag decoder that uses the information of the previous NE tag (the output vector o_{i-1}) and the succeeding NE tag (o_{i+1}) for the decoding of t_i, since we can assume that both of them are effective features. Note that BiLSTM cannot be applied as Tag Decoder. o_{i+1} is not predicted by the model at time i when the NE tags are determined in the forward direction, and using the backward direction leaves o_{i-1} undetermined. In our method, two unidirectional LSTM models are trained in the training phase and combined in the test phase as follows.

- M_f, a model using a forward LSTM in Tag Decoder, is trained. In this model, o_{t-1} is added as an input of LSTM at time i, as shown in Figure 5.
- M_b, a model using a backward LSTM in Tag Decoder, is trained. In this model, o_{t+1} is added as an input of LSTM at time i.
- In order to recognize named entities from an unknown sentence, M_f and M_b are applied and the output vectors o_i^f and o_i^b are obtained. Then, the average of $o_i = (o_i^f + o_i^b)/2$ is calculated. The NE tag at time i is determined by the index of the highest score of o_i.

3.3 Separate embedding fine-tuning

In our method, the parameters of the word and character embeddings are fine-tuned, that is, they are updated through training of the NER model. However, since the number of parameters is increased by the fine-tuning, not only does this involve a high computational cost, but also the fine-tuned embedding parameters might not fit the NER task well. Our method, called *separate embedding fine-tuning*, tackles these problems. The basic idea is to train the *embedding parameters* and *model parameters* separately. The model parameter are the parameters except for the word and character embeddings, including LSTM in Character Encoder, BiLSTM in

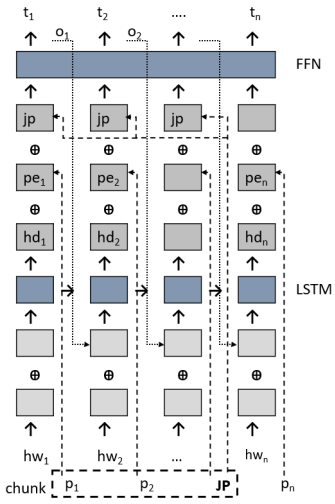

Figure 5: Tag Decoder

Word Encoder, part-of-speech embedding, Japanese particle embedding, LSTM in Tag Decoder, and FNN in Tag Decoder. Our separate embedding fine-tuning is carried out as follows.

1. Word and character embeddings are pre-trained. They are used as initial embedding parameters.
2. The classification model is trained until the loss function is saturated. In this step, only the model parameters are estimated, while the embedding parameters are fixed.
3. The model is trained again, where only the embedding parameters are updated and the model parameters are fixed. the embedding parameters are fine-tuned until the loss function is saturated.

4 Evaluation

4.1 Experimental settings

The news articles in the NE tagged corpus were divided into about 90,000 sentences, then split into training, development and test data-sets. The proportions of the training, development and test data are 80%, 10% and 10% respectively. Each subset contains named entities of almost all 26 classes. Among 52,208 NE types in all the data-sets, 45,097(86%) NEs are ambiguous, i.e. they have two or more NE classes, while 7,111(14%) NEs have one NE class. Table 3 shows the statistics of the data-sets.

Table 2: Definition of models

	Character Encoder	Word Encoder	Tag Decoder	pre-trained	fine-tuning	vector size hidden	pos	jp
Shen's model	BiLSTM	BiLSTM	forward LSTM	no	–	128	–	–
TDf-small	LSTM	BiLSTM	forward LSTM	no	–	128	128	128
TDf	LSTM	BiLSTM	forward LSTM	yes	whole	256	64	64
TDb	LSTM	BiLSTM	backward LSTM	yes	whole	256	64	64
TDfb	LSTM	BiLSTM	forward&backward	yes	whole	256	64	64
TDfb-sep	LSTM	BiLSTM	forward&backward	yes	separate	256	64	64

Table 3: Dataset

Data	Sentence	Token	NE
Training	71,854	1,781,685	187,814
Development	8,980	222,403	23,278
Test	8,980	222,073	23,283

The models were trained on Google Colaboratory, which allows us to use one Tesla K80 GPU. With the development data, we use ADAM(Kingma and Ba, 2014) for the optimization of the hyper parameters in the following configuration:

1. Number of hidden units in LSTM/BiLSTM = 128 or 256
2. Optimizer ADAM learning $\alpha = [10^{-4}, 10^{-7}]$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$

Table 2 summarizes methods compared in this experiment. First, we implemented almost the same model as that of (Shen et al., 2018) as the baseline. Next, TDf-small, our base model with a relatively small neural network, was trained for quick comparison with the baseline. A major difference between them is that inclusion of POS and Japanese particles is only applied by TDf-small.

The rest of the models use pre-trained character and word embeddings, as indicated in the column of "pre-trained" in Table 2. In these models, we increased the number of hidden units from 128 to 256, since this was able to improve the F1-score in our preliminary experiment. On the other hand, we reduced the dimension of the vector of POS (*pos*) and Japanese particles (*jp*) from 128 to 64, considering the additional computational cost for the fine-tuning of the word and character embeddings. TDf, TDb and TDfb use forward, backward and both LSTM in Tag Decoder respectively, where all parameters including word/character embeddings are trained simultaneously. Finally, TDfb-sep was trained with our separate embedding fine-tuning proposed in subsection 3.3.

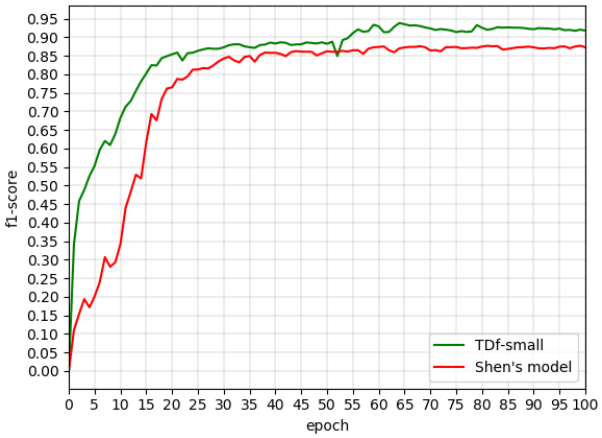

Figure 6: F1-score on the development data of Shen's and our model

The precision, recall and F1-score of each named entity class on the test data were measured in the experiment. However, for the comparison, a micro average of the F1-score for 26 NE classes is used as the major evaluation criterion.

4.2 Results and discussion

Figure 6 shows the F1-score on the development data of the baseline and our model. It shows that TDf-small clearly outperforms Shen's model, which is one of the state-of-the-art models. The number of the epochs for training the models was determined so that the F1-score becomes the highest on the development data, then the NER performance of two models in the test data was measured. The F1-score of TDF-small

Table 4: Performance of NER on the test data

Model	F1-score	statistical test
Shen's model	0.8839	–
TDf	0.9316	–
TDb	0.9302	$p < 0.05$ (vs TDf)
TDfb	0.9337	$p < 0.05$ (vs TDf)
TDfb-sep	0.9440	$p < 0.01$ (vs TDfb)

Table 5: F1-score comparison between TDf and TDb

NE class	TDf	TDb	dif.*
Product	0.942	0.946	−0.004
Person	0.957	0.958	−0.001
Timex	0.985	0.983	+0.002
Countx	0.940	0.929	+0.011
Organization	0.903	0.901	+0.002
Periodx	0.967	0.966	+0.001
Ordinal_Number	0.891	0.890	+0.001
Location	0.951	0.946	+0.005
Facility	0.865	0.858	+0.007
Event	0.884	0.877	+0.007
Age	0.987	0.985	+0.002
Percent	0.990	0.987	+0.003
Natural_Object	0.853	0.817	+0.036
Disease	0.947	0.900	+0.047
Multiplication	1.00	1.00	0
Rank	0.896	0.912	−0.016
Numex_Other	0.731	0.686	+0.045
Frequency	0.667	0.511	+0.156
Point	0.891	0.871	+0.020
Measurement	0.966	0.942	+0.024
Money	0.991	0.994	−0.003
School_Age	0.962	0.905	+0.057
Color	0.895	0.729	+0.166
God	0.571	0.333	+0.238
Name_Other	0.769	0.566	+0.203
Latitude_Longtitude	0.00	0.00	0

* dif. means TDf − TDb

was 0.9254, which was obviously better than Shen's model, namely 0.8839. These results prove that the inclusion of POS and Japanese particles is effective.

Next, we evaluated our proposed methods using forward, backward, and both LSTM in Tag Decoder. Table 4 presents the F1-score of our models on the test data, while Table 5 compares TDf and TDb for each NE class. The model with forward LSTM in Tag Decoder slightly outperforms the backward model in the overall F1-score in Table 4 and most of the NE classes in Table 5. However, TDb is better than TDf for some NE classes, namely, Product, Person, Money and Rank. When forward and backward LSTM are combined, the F1-score is slightly improved from the model using only forward LSTM. Although the difference is small, it is confirmed by McNemar's test that TDfb is significantly better than TDb at the 95% confidence level. From the above results, we can conclude that combining forward and backward LSTM in Tag Decoder is effective.

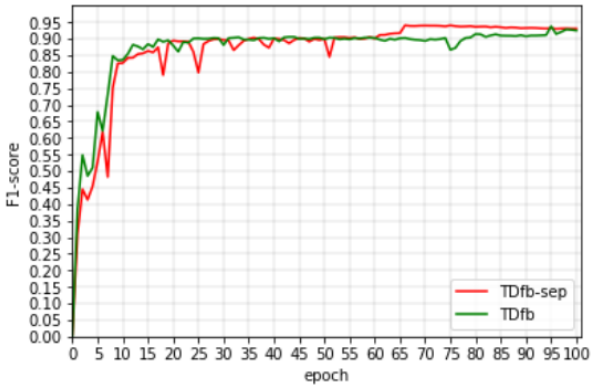

Figure 7: F1-score on the development data of TDfb and TDfb-sep

We evaluated the separate embedding fine-tuning by comparing TDfb and TDfb-sep. Figure 7 shows the change of F1-score on the development data of these two models. In the training of TDfb-sep, the parameters of the character and word embeddings were fixed until the 60^{th} epoch where the loss function saturates in the training. After the 61^{th} epoch, the parameters of the character and word embeddings were fine-tuned, while the model parameters were fixed. It can be seen in Figure 7 that the F1-score is sharply improved at the 61^{th} epoch. Furthermore, in the comparison on the test data in Table 4, the F1-score of TDfb-sep is significantly better than that of TDfb, at the 99% confidence level. These results indicate that the idea of updating the model parameters and embedding parameters separately is effective for training the neural based NER model.

Finally, the precision(P), recall(R) and F1-score(F) of our best model, TDfb-sep, for each NE class are shown in Table 6. The last column "NE" shows the number of named entities in the test data. Among the 26 NE classes, the F1-scores for 17 classes are higher than 90%, which are satisfying results for a practical NLP system. The F1-score is lower than 80% when the number of named entities in the test data (also in the training data) is small, as

Table 6: Performance of NER for each class

NE class	P	R	F	NE
Product	0.961	0.949	0.955	5580
Person	0.970	0.969	0.969	3021
Timex	0.976	0.990	0.983	2215
Countx	0.955	0.932	0.943	1333
Organization	0.940	0.910	0.925	2649
Periodx	0.968	0.978	0.973	495
Ordinal_Number	0.940	0.915	0.927	328
Location	0.971	0.939	0.955	3413
Facility	0.855	0.943	0.894	916
Event	0.908	0.885	0.897	766
Age	0.985	0.992	0.988	384
Percent	0.981	0.997	0.988	303
Natural_Object	0.931	0.773	0.845	436
Disease	0.900	0.960	0.929	150
Multiplication	1.00	1.00	1.00	13
Rank	0.974	0.912	0.942	205
Numex_Other	0.773	0.699	0.734	73
Frequency	0.476	0.769	0.588	13
Point	0.892	0.916	0.904	154
Measurement	0.937	0.966	0.951	292
Money	0.988	0.990	0.989	411
School_Age	0.916	0.974	0.944	78
Color	0.780	0.914	0.842	35
God	0.400	0.500	0.444	4
Name_Other	0.600	0.800	0.686	15
Latitude_Longtitude	0.00	0.00	0.00	0
micro average	0.944	0.944	0.944	23282
macro average	0.845	0.868	0.854	23282

is the case with Numex_Other, Frequency, God, and Name_Other. This might be caused by the insufficiency of the training data. However, the F1-scores of the named entities that are often regarded as important reach over or around 90%, such as Timex (98.3%), Person (96.9%), Product(95.5%), Location (95.5%), Organization (92.5%), and Event (89.7%).

5 Conclusion

This paper proposed the novel method of deep learning for Named Entity Recognition in Japanese. Our model consisted of three neural network modules: Character Encoder, Word Encoder and Tag Decoder. In addition to word embedding and character embedding, two important features were added. One was the part-of-speech that is widely used in various NLP tasks, the other was the Japanese particles, which play a significant grammatical role in Japanese.

The first contribution of this paper was to combine forward and backward LSTM in Tag Decoder. The information of both left and right contexts was thought to be necessary for an accurate NER. However, since the NER tag of one or the other of either the previous or succeeding words is inapplicable in a sequential labeling model, BiLSTM could not be simply applied. To use both the previous and succeeding NE tags for classification, models using forward LSTM and backward LSTM in Tag Decoder were separately trained, then, in the test phase, the NE tag of each word was determined by the sum of the probability distributions of the NE tags of the two models. Our model using both directions of LSTM slightly outperformed the models with forward or backward LSTM in our experiment.

The second contribution was to propose a method of fine-tuning of the word and character embeddings. Although fine-tuning an embedding has been a promising approach to improve the performance of deep learning models for NLP, it increases the number of parameters considerably. In our approach, the model parameters were first trained with pre-trained and fixed word and character embeddings, then the parameters of the word and character embeddings were fine-tuned with the fixed model parameters. This method was able to improve the F1-score by 0.01 point in our experiment. Furthermore, our best model was obviously better than the baseline, and achieved an F1-score of 0.944.

In the future, we will explore other effective features to be added to the neural network in order to improve the model performance. We will also investigate various adjustment methods for the hyper parameter estimation. Another important line of future research is to investigated whether the model with two directional tag decoder is effective for NER of other languages such as English. We plan to evaluate our model on CoNLL-2003 English dataset.

Acknowledgements

This research is financially supported by the National Science and Technology Development Agency (NSTDA), National Research University Project, Thailand Office of the Higher Education Commission, Japan Advanced Institute of Technology (JAIST), and Infrastructure Engineering Research Unit, Sirindhorn International Institute of Technology (SIIT), Thammasat University (TU).

References

Raghavendra Chalapathy, Ehsan Zare Borzeshi, and Massimo Piccardi. 2016. Bidirectional LSTM-CRF for clinical concept extraction. *ClinicalNLP 2016*, page 7.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Proceedings of the Empiricial Methods in Natural Language Processing*.

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

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. In *NLPCC/ICCPOL*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5–6):602–610.

Taiichi Hasimoto, Takashi Inui, and Koji Murakami. 2008. Constructing extended named entity annotated corpora. *Technical Report*, 008-NL-188:113–120. Written in Japanese.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

John J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, 79(8):2554–2558.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, volume abs/1412.6980.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1064—-1074, Berlin, Germany.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775.

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

Shotaro Misawa, Motoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. 2017. Character-based bidirectional LSTM-CRF with words and characters for Japanese named entity recognition. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 97–102.

Thien Huu Nguyen, Avirup Sil, Georgiana Dinu, and Radu Florian. 2016. Toward mention detection robustness with recurrent neural networks. In *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence*, volume abs/1602.07749, New York, USA.

Hoang Pham and Le-Hong Phuong. 2017. End-to-end recurrent neural network models for Vietnamese named entity recognition: Word-level vs. character-level. In *The 15th International Conference of the Pacific Association for Computational Linguistics*.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, dictionaries and tagger for extended named entity hierarchy. In *Proceedings in the 4th International Conference on Language Resources and Evaluation*, pages 1977–1980.

Yan Shao, Christian Hardmeier, and Joakim Nivre. 2016. Multilingual named entity recognition using hybrid neural networks. In *The Sixth Swedish Language Technology Conference*.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *International Conference on Learning Representations*.

Yuji Matsumoto Taku Kudo. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 63–69.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*, volume 4, pages 142–147. Association for Computational Linguistics.

Kai Xu, Zhanfan Zhou, Tianyong Hao, and Wenyin Liu. 2018. A bidirectional LSTM and conditional random fields approach to medical named entity recognition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 355–365.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270.