Effective Named Entity Recognition with Boundary-aware Bidirectional Neural Networks

Fei Li[1,2], Zheng Wang[3,*], Siu Cheung Hui[3], Lejian Liao[1,2,*], Dandan Song[1,2] and Jing Xu[1,2]
[1]School of Computer Science and Technology, Beijing Institute of Technology, China
[2]Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, China
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore
{lifei926,liaolj,sdd}@bit.edu.cn,{wang_zheng,asschui}@ntu.edu.sg,bitjingxu@gmail.com

ABSTRACT

Named Entity Recognition (NER) is a fundamental problem in Natural Language Processing and has received much research attention. Although the current neural-based NER approaches have achieved the state-of-the-art performance, they still suffer from one or more of the following three problems in their architectures: (1) boundary tag sparsity, (2) lacking of global decoding information; and (3) boundary error propagation. In this paper, we propose a novel Boundary-aware Bidirectional Neural Networks (Ba-BNN) model to tackle these problems for neural-based NER. The proposed Ba-BNN model is constructed based on the structure of pointer networks for tackling the first problem on boundary tag sparsity. Moreover, we also use a boundary-aware binary classifier to capture the global decoding information as input to the decoders. In the Ba-BNN model, we propose to use two decoders to process the information in two different directions (i.e., from left-to-right and right-to-left). The final hidden states of the left-to-right decoder are obtained by incorporating the hidden states of the right-to-left decoder in the decoding process. In addition, a boundary retraining strategy is also proposed to help reduce boundary error propagation caused by the pointer networks in boundary detection and entity classification. We have conducted extensive experiments based on three NER benchmark datasets. The performance results have shown that the proposed Ba-BNN model has outperformed the current state-of-the-art models.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**.

KEYWORDS

named entity recognition, boundary retraining, bidirectional decoding, pointer networks

ACM Reference Format:
Fei Li, Zheng Wang, Siu Cheung Hui, Lejian Liao, Dandan Song and Jing Xu. 2021. Effective Named Entity Recognition with Boundary-aware Bidirectional Neural Networks. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, Ljubljana, Slovenia, 9 pages. https://doi.org/10.1145/3442381.3449995

*Corresponding author.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449995

	Boundary tag sparsity	Lacking of global decoding information	Boundary error propagation
Softmax	√	√	×
CRF	√	√	×
RNN	×	√	√
PN	×	√	√

Table 1: The current state-of-the-art neural NER approaches and their corresponding problems (√ indicates the problem exists).

1 INTRODUCTION

The task of named entity recognition (NER) is to find and classify the type of a named entity in text, such as *person (PER)*, *location (LOC)* or *organization (ORG)*. It is widely viewed as a fundamental problem in natural language processing (NLP) and serves many downstream applications such as entity linking [7], relation extraction [26], question generation [28] and coreference resolution [2].

NER is typically modeled as a sequence labeling task, where each word in a sentence is assigned a special label. In recent years, many neural-based NER approaches were proposed as end-to-end sequence labeling models. In general, these approaches can be classified into four categories according to their decoding architectures [15]: Multi-Layer Perceptron (MLP) with Softmax [6, 23, 29], Conditional Random Fields (CRF) [3, 9, 14], Recurrent Neural Networks (RNN) [20], and Pointer Networks (PN) [16, 27]. Although these approaches are currently the state-of-the-art techniques for NER, they still suffer from one or more of the following common problems in NER research: (1) boundary tag sparsity; (2) lacking of global decoding information; and (3) boundary error propagation. Table 1 summarizes the problems for each category of the neural-based NER approaches.

The problem on boundary tag sparsity is due to the semantic nature of natural language, in which entities are rare and non-entities are common in a sentence. MLP with Softmax and CRF are two typical approaches that suffer from such problem as they are unable to tackle it with simple classifiers such as Maximum Entropy [10]. The lack of global decoding information is a common problem in existing encoder-decoder architectures. This is mainly due to the individual decoding process (e.g., in Softmax) or the nature

of unidirectional decoding processing from left-to-right (e.g., in CRF, RNN and PN). For example, the RNN-based tag decoder can serve as a language model to condition each output tag based on the previously generated tag. Unfortunately, information on unexploited context is lacking and not available. Intuitively speaking, such global information is important as it follows human cognitive habit in reading and writing.

The problem on boundary error propagation occurs when pointer networks-based decoders [16, 27] are deployed for the NER task. Pointer networks are built on top of encoder-decoder architectures with the encoder extracting contextual representation for an input sequence, and the decoder detecting named entities via a pointer mechanism to find the corresponding boundary positions from the encoder. Although these pointer networks-based models have achieved better performance on tackling the problems of boundary tag sparsity and variable size vocabulary (i.e., models need retraining with respect to different vocabulary sizes) as discussed in [16], they inevitably accumulate errors in the decoding process. The pointer networks-based architecture depends heavily on the accuracy of entity boundary detection. As the input of the current detection will need the output of the previous detection, it will be propagated in the NER process if a boundary detection error occurs. As such, the accuracy of entity classification will be adversely affected. This seems to be the obvious disadvantage of such pointer networks-based approaches.

In this paper, we propose a novel Boundary-aware Bidirectional Neural Networks (Ba-BNN) to tackle the three problems encountered by the current neural-based NER approaches as follows. Firstly, the proposed Ba-BNN model is constructed on top of encoder-decoder architecture and integrated with the pointer mechanism [24] into a sequence labeling framework to deal with the boundary tag sparsity problem. Secondly, we propose a boundary-aware bidirectional decoding mechanism to capture the global decoding information. In particular, we first use a binary classifier to predict whether the word in the sentence is a boundary or not with supervised training. This enables the hidden states of these words carry more context-aware boundary features during the process of network optimization. A Self-Attention function is then applied on these hidden states to obtain global information as input to the decoders. In addition, we also use two decoders to process the information in two different directions (i.e., from left-to-right and right-to-left). The final hidden states of the left-to-right decoder are obtained by incorporating the hidden states of the right-to-left decoder in the decoding process. Thirdly, we propose a boundary retaining strategy to train the boundaries that were wrongly predicted due to the pointer mechanism. As the input of decoder is now with boundary-aware global information, the error propagation problem would be alleviated.

To summarize, the main contributions of this paper are as follows:

- We classify the current neural-based NER approaches into four main categories and identify three major problems encountered by them. We propose a novel Boundary-aware Bidirectional Neural Networks (Ba-BNN) which integrates a suite of techniques including the pointer networks mechanism, boundary-aware bidirectional decoding and boundary retraining strategy in order to tackle the current NER problems. These techniques benefit each other from the sharing of information with multitask training. Thus, our proposed Ba-BNN model achieves superior performance than the current neural-based NER approaches.
- We conduct extensive experiments on three NER benchmark datasets, namely CoNLL2003, WNUT2017 and JNLPBA. The experimental results have shown that our Ba-BNN model has achieved the state-of-the-art performance and outperforms the current neural-based NER models.

The rest of the paper is organized as follows. We review the related work in Section 2 and present our proposed model in Section 3. The experimental results are discussed in Section 4. Finally, we conclude the paper in Section 5.

2 RELATED WORK

In this section, we review the related work on the current approaches for named entity recognition. These approaches can be categorized into Multi-Layer Perceptron (MLP) with Softmax, Conditional Random Fields (CRF), Recurrent Neural Networks (RNN) and Pointer Networks (PN).

2.1 Multi-Layer Perceptron with Softmax

Applying a Multi-Layer Perceptron (MLP) with Softmax as the label decoder layer, the sequence labeling task is acting as a multi-class classification problem. The label for each word is independently predicted based on the context-dependent representations without taking into consideration of its neighbors. Tomori et al. [23] used deep neural networks as an encoder to learn the contextual representations of words, which were concatenated with the state embeddings and fed into a Softmax layer for predicting the named entities. Instead of deep neural networks, Transformer was used by Devlin et al. [6] as an encoder to learn the contextual word representations, and then Multi-Layer Perceptron with Softmax was applied as a decoder for labeling the words. Gregoric et al. [29] employed multiple independent bidirectional LSTM [8] to learn different features, and then these features are concatenated and fed into a Softmax layer for named entity prediction.

2.2 Conditional Random Fields

Conditional random fields (CRF) is a random field that is globally conditioned on the observation sequence [12]. Huang et al. [9] utilized the bidirectional LSTM as an encoder to learn the contextual representation of words, and then conditional random fields was used as a decoder to label words as a sequence labeling task. It has achieved the state-of-the-art results on various datasets. Inspired by the success of BiLSTM-CRF, the related models with some adaptations have been widely used in the field of NER for the past few years. Specifically, Chiu and Nichols [3] used convolutional neural network (CNN) to capture spelling features, and the characters and word characteristics are concatenated as the input of BiLSTM with CRF network. Additionally, Lample et al. [14] used RNN-BiLSTM-CRF as an alternative, and Strubell et al. [21] proposed an improved CNN model called ID-CNNs, which can encode words concurrently for a large amount of textual data.

2.3 Recurrent Neural Networks

With recurrent neural networks (RNN) as the decoder layer for labeling tasks, this network structure serves as a language model to greedily produce a sequence with the traditional seq2seq structure. Shen et al. [20] reported that RNN decoders can achieve close to the state-of-the-art performance and perform better when training with a large number of entity types. Specifically, at the first step, a special [GO]-symbol is provided as the first input y_1 to the RNN decoder. Subsequently, at each time step, the RNN decoder computes the current decoder's hidden state h_{i+1}^{Dec} for decoding word $i+1$ according to the previous tag y_i, the previous decoder's hidden state h_i^{Dec} and the encoder's hidden state h_{i+1}^{Enc} of the current word. The current generated h_{i+1}^{Dec} is decoded via a Softmax function and is then fed in as an input to the next step. Finally, the labeled sequence is obtained over all steps.

2.4 Pointer Networks

Pointer networks [24] was proposed to solve combinatorial optimization problems. It uses the attention mechanism as a pointer to select one of the words in the input sequence as the output. In [27], Zhai et al. used the pointer networks for the sequence chunking task. The decoder first detects the possible positions with a predefined maximum chunk length at each time step, and then these chunks are labeled with a Softmax function. Subsequently, Li et al. [16] used the pointer networks for named entity boundary detection. At each time step, the starting boundary word in an entity is trained to point to the corresponding ending boundary word, and the non-boundary word is trained to point to a specific position. This method has achieved promising results.

In this paper, our proposed model has also adopted the pointer networks-based architecture. However, our proposed model has incorporated some important features to tackle the shortcomings of the current approaches. Different from the current approaches, our proposed model has the following features: (1) We explore the bidirectional decoding method and propose a boundary-aware binary classifier to tackle the problem on the lacking of global decoding information as discussed in Section 1. (2) A boundary retraining strategy is proposed to help reduce boundary error propagation caused by the pointer networks in boundary detection and entity classification .

3 BOUNDARY-AWARE BIDIRECTIONAL NEURAL NETWORKS

In this paper, we propose a novel Boundary-aware Bidirectional Neural Networks called Ba-BNN, which integrates a suite of techniques for tackling the problems that occurred in the current neural-based NER approaches. Figure 1 shows the architecture of the proposed Ba-BNN model which adopts the pointer networks as the decoder layer. The proposed model is operated as follows. First, each word in the sentence is mapped into its embedding and the Input Encoder encodes the embedding into a context-aware representation. Next, the Entity Boundary Detection deploys bidirectional decoders (i.e., Left Decoder and Right Decoder) with boundary-aware binary classifier to detect entity boundaries in two different directions via the pointer mechanism. Then, the Entity Chunk Generation generates candidate entity chunks from the bidirectional decoders.

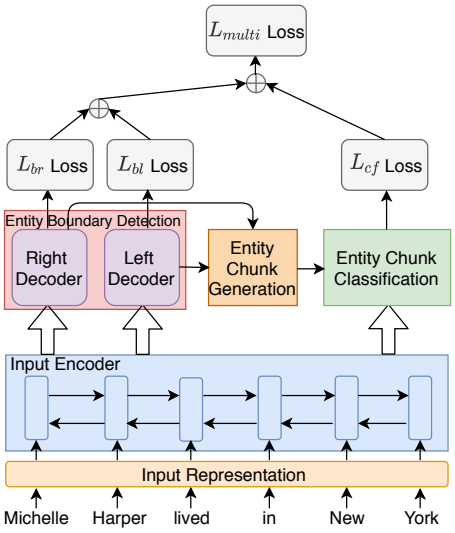

Figure 1: The architecture of our proposed Ba-BNN.

Finally, the Entity Chunk Classification classifies each candidate entity chunk into the corresponding entity type or non-entity with the boundary retraining strategy. Note that as Entity Boundary Detection and Entity Chunk Classification share the same encoder, we apply multitask training when training the proposed Ba-BNN model.

3.1 Input Encoder

Given an input sentence $S =< w_1, w_2, \cdots, w_n >$, each word $w_i (1 \leq i \leq n)$ is represented as

$$x_i = [x_i^w; x_i^c; x_i^u], \tag{1}$$

by using a concatenation of a pre-trained word embedding x_i^w, a character-level word embedding x_i^c and a word feature embedding x_i^u. The pre-trained word embedding x_i^w is obtained from Glove [18]. The character-level word embedding x_i^c is obtained with a bidirectional LSTM to capture the orthographic and morphological information. It considers each character in the word as a vector, and then inputs them to a bidirectional LSTM to learn hidden states. The final hidden states from the forward and backward outputs are concatenated as the character-level word information. For word feature embedding x_i^u, we consider word w_i with four types of characteristics: (1) all capital characters; (2) starting with a capital character; (3) all lower case characters; and (4) all digit characters. These word feature embedding and character embedding are randomly initialized and learned during training. After that, the distributed representations of the word embeddings $< x_1, x_2, ..., x_n >$ are fed into an Input Encoder with bidirectional LSTM to compute the hidden sequences in forward $\overrightarrow{h} =< \overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_n} >$ and backward $\overleftarrow{h} =< \overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_n} >$. Finally, we concatenate $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ as an output at each word, i.e., $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$.

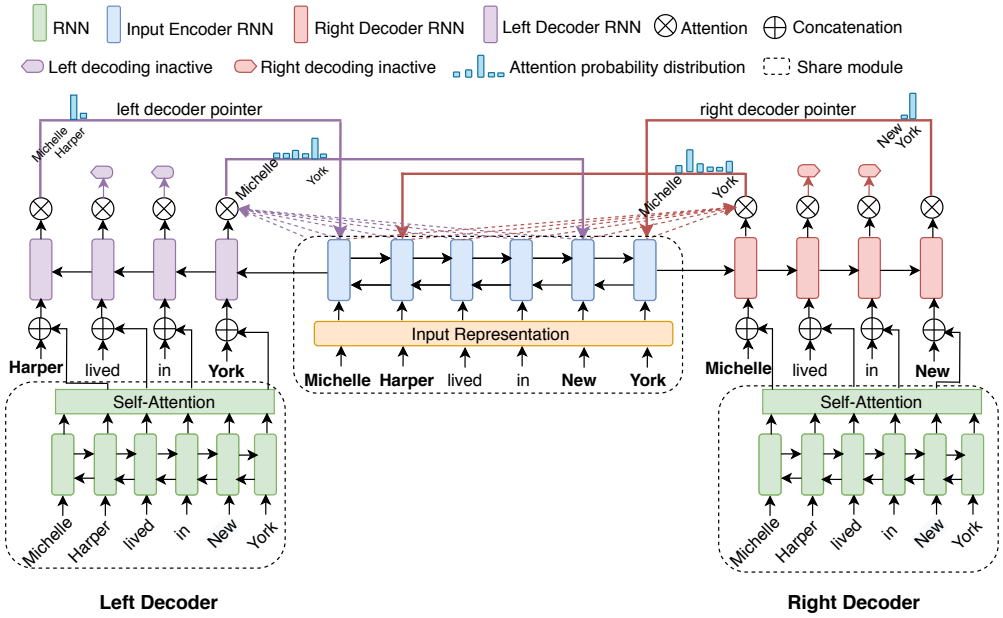

Figure 2: The entity boundary detection process using bidirectional pointer networks with boundary-aware binary classifier.

3.2 Entity Boundary Detection

After obtaining the representation of each word, the next step is to detect entity boundaries. To achieve this, we have adopted the pointer mechanism to sequentially detect boundaries [16]. Specifically, we pad the hidden states of the encoder with a sentinel word representing *inactive*. If the current input is not an entity boundary, the pointer points to the sentinel word *inactive*. In this research, we use bidirectional pointer networks for entity boundary detection as shown in Figure 2. There are two decoders based on the pointer networks. For the Right Decoder, we pad a *right decoding inactive* at the last position of the hidden states in the Input Encoder. If the input is not an entity boundary in left-to-right decoding, we train the *right decoder pointer* to point to the *right decoding inactive*, and vice versa for the Left Decoder.

More specifically, we first pad the hidden states h of the Input Encoder with two sentinel vectors at the first and last positions as follows:

$$h = [0; h; 0], \tag{2}$$

where $h \in \mathbb{R}^{(n+2)\times D}$, n is the length of the original sentence, and D is the dimension of the hidden states in the Input Encoder. Then, two LSTMs are employed as the Right Decoder and Left Decoder to output the decoder hidden states hr and hl, respectively. For the input of the decoders, we concatenate the word embedding of the current focus word w_i with global boundary features, which are obtained via another bidirectional LSTM in order to capture context-aware representation H, and Self-Attention is applied on H to incorporate global features. To do this, we use a binary classifier on H to predict whether the current word is a boundary or not as follows:

$$d_i = Softmax(WH_i + b), \tag{3}$$

where W and b are trainable parameters which can be optimized, and d_i is a predicted label (i.e., 0 or 1) for the i-th word. Intuitively, the binary classifier enables the hidden states to carry more context-aware boundary features during the process of network optimization, thereby enhancing the accuracy of entity boundary detection by the pointer networks.

Next, we generate a feature representation for each possible boundary position i at time step j. Moreover, to provide chunk level feature, we follow [27] and add the information on the length of chunk. We use the attention mechanism and the chunk length to construct the feature representation for left-to-right decoding as follows:

$$\begin{aligned} u_i^j = {} & {v_1}^T tanh(W_1 h_i + U_1 hr_j) \\ & + {v_2}^T LE(i - j + 1), \text{for } i \in [j, n+2] \end{aligned} \tag{4}$$

Similarly, the feature representation for right-to-left decoding is constructed as follows:

$$\begin{aligned} u_i^j = {} & {v_3}^T tanh(W_2 h_i + U_2 hl_j) \\ & + {v_4}^T LE(j - i + 1), \text{for } i \in [0, j] \end{aligned} \tag{5}$$

Then, the $Softmax$ function is used to obtain the probability of word w_i for determining an entity boundary:

$$p(w_i|w_j) = Softmax(u_i^j) \tag{6}$$

where v_1, v_2, v_3, v_4, W_1, W_2, U_1 and U_2 are learnable parameters, $LE(\cdot)$ is a chunk length embedding, $i \in [j, n+2]$ and $i \in [0, j]$ indicate a possible position in left-to-right and right-to-left decoding respectively, and $p(w_i|w_j)$ denotes the probability of word w_i of entity ending (or starting) boundary given the entity starting (ending) boundary w_j. When w_j is not a boundary of any entity, the pointers are trained to point to the padded sentinel words.

The Right Decoder processes the input in a left-to-right decoding manner. When an input word is denoted as a starting boundary in an entity, the attention probability distribution will be computed. The index of the ending boundary is always equal to or greater than the index of the starting boundary in the Right Decoder. The possible word position for computing the attention probability distribution may vary with each decoding step. Once an ending boundary is identified, a candidate entity chunk is determined. Then, the Right Decoder starts processing the next word after the ending boundary from the last decoding step. Other words in the entity do not need to be processed by the Right Decoder. For example, in Figure 2, "Harper" is the ending boundary of "Michelle Harper", and the word "Harper" will not be processed by the Right Decoder. Similarly, the Left Decoder processes the input in the right-to-left decoding manner.

We describe the process of Right Decoder with the example sentence "Michelle Harper lived in New York" as follows:

- The Right Decoder starts with the input word "Michelle", which is the starting boundary of the entity "Michelle Harper". Thus, Ba-BNN computes an attention probability distribution over all positions (i.e., from "Michelle" to "York") in the input sentence. The word "Harper" with the maximum probability from the attention probability distribution is identified and then assigned to "Michelle". That is, a *right decoder pointer* "Michelle⟶ Harper" is constructed.
- Then, "lived" is processed as the input word to the Right Decoder. It needs to compute the probability distribution from "lived" to "York". However, it is identified as an non-entity word from the distribution. Then, the *right decoder pointer* points to *right decoding inactive*.
- Similarly, the *right decoder pointer* also points to *right decoding inactive* for the word "in".
- Next, "New" is processed as the next input word to the Right Decoder, and the proposed Ba-BNN computes the attention probability distribution from "New" to "York". As a result, "York" is identified as the ending boundary and assigned to "New", and a *right decoder pointer* "New⟶ York" is constructed.

3.3 Entity Chunk Generation

After Entity Boundary Detection, we obtain two sets of chunks (in boundary pairs) from Right Decoder and Left Decoder. Next, we introduce the following three strategies for entity chunk generation, which are shown in Figure 3:

Naive: It simply chooses the entity chunks that are formed from the Right Decoder. Intuitively, the Right Decoder is more natural in language processing, i.e., processing a sentence from left to right. The Left Decoder can be considered as auxiliary to the Right Decoder as they share the same encoder and are trained together as a multitask. This strategy is shown in Figure 3(a).

Selection-aware Generation (SG): It merges all the candidate entity chunks for chunk generation. However, based on empirical findings, the performance is not very good. Therefore, we keep the entity chunks from the Right Decoder and select the entity chunks from the Left Decoder that have partial chunk overlap with the Right Decoder. This strategy is shown in Figure 3(b).

Context-aware Generation (CG): In order to capture the contextual information, we concatenate the hidden states from hl to hr as a new hidden state to generate the entity chunks from the Right Decoder. This strategy is shown in Figure 3(c).

As shown in Figure 3, we assume that the set of candidate entity chunks detected by the Right Decoder is $\{(1,2),(3,3),(5,6)\}$, and the set of candidate entity chunks detected by the Left Decoder is $\{(1,1),(4,4),(5,6)\}$. The candidate entity chunks which are indicated in red color inside the oval shape are the final generated chunks of the corresponding strategy.

3.4 Entity Chunk Classification

After determining the entity chunks, the next step is to classify them into their corresponding entity types. Figure 4 shows the Entity Chunk Classification process. We use a Bi-LSTM network to obtain the representations of entity chunks. Given an entity chunk $ch(i,j)$, where i is the starting boundary and j is the ending boundary. We first get the context-aware word representation sequence of each word in the entity chunk $ch(i,j)$ as $T = h[i:j] \in \mathbb{R}^{(j-i+1)\times D}$, where h refers to the hidden states of the Input Encoder discussed in Section 3.1. Then, the Chunk Encoder learns to map T to a fixed-sized vector. To do this, we pass T to a Bi-LSTM network to compute the hidden sequence, and concatenate the first hidden state and last hidden state as the representation of the entity chunk. Additionally, we also append the information on the length of chunk. After that, a multi-layer perception (MLP) is used to obtain the dense vector representations for the entity chunks. Finally, the $Softmax$ function is adopted to predict the types for the candidate entity chunks.

In order to further improve the prediction performance, we introduce the Boundary Retraining strategy. In general, the entity chunk classifier can be trained with the ground truth labels following the ideas of teacher forcing [13]. However, it is ineffective because the entity chunks generated by Entity Chunk Generation are not all correctly identified, which may cause error propagation. To resolve the issue, we assign a new class "O" to assign the incorrect entity chunks to non-entity chunks. Specifically, we collect the negative boundary cases produced during the process of entity boundary detection, and extend the original training data with the collected cases to train our model.

However, as appending all negative cases to the training data may cause the imbalance problem, i.e., the negative cases with label "O" are much larger than the positive cases with entity labels, it will degrade the recall performance accordingly. As such, we consider to extend the training data only when the model tends to converge. Specifically, we use ground truth labels to train the classifier at the early stage of model training. When the model tends to converge, i.e., the change in F1(%) on the development dataset is lower than a given threshold θ, we append negative cases to the training data to fine-tune the classifier.

3.5 Multitask Training

There are two main processes in our Ba-BNN model: Entity Boundary Detection and Entity Chunk Classification. The loss L_{br} and L_{bl} are for detecting entity boundaries from the Right Decoder and Left Decoder, respectively. The loss L_{ba} is for boundary-aware binary

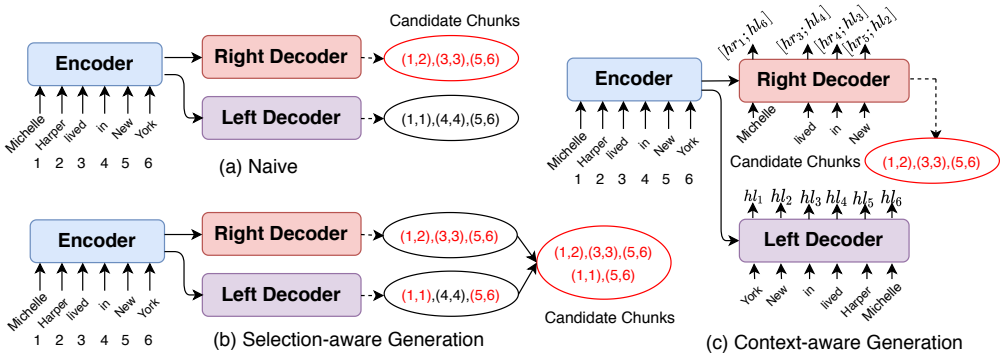

Figure 3: The Entity Chunk Generation process.

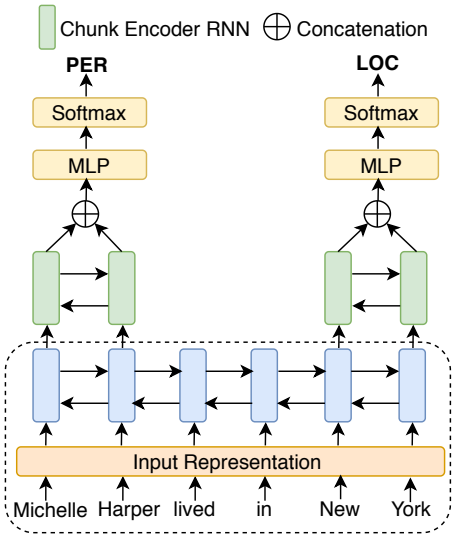

Figure 4: The Entity Chunk Classification process.

classifier to enhance entity boundary detection with global boundary information. The loss L_{cf} is for predicting the entity chunk type. In particular, L_{br} and L_{bl} are used to optimize entity boundary detection, L_{ba} is used to enhance entity boundary detection, and L_{cf} is used to optimize entity chunk type prediction based on the boundaries. As boundary detection and chunk type prediction share the same encoder, we apply a multitask loss for training the Ba-BNN model as follows:

$$L_{multi} = \alpha(L_{br} + L_{bl} + L_{ba}) + (1 - \alpha)L_{cf}, \tag{7}$$

where α is a hyperparameter that is used to balance the importance of each process.

4 EXPERIMENTS

In this section, we first discuss the datasets, baseline models and parameter settings used in the experiments. Then, we present the experimental results on the three benchmark datasets. Moreover, an ablation study is also conducted.

Dataset		train	dev	test
CoNLL2003	#sentences	14,987	3,466	3,684
	#entities	23,499	5,942	5,648
WNUT2017	#sentences	3,394	1,009	1,287
	#entities	3,160	1,250	1,589
JNLPBA	#sentences	16,691	1,853	3,855
	#entities	46,388	4,902	8,657

Table 2: Statistics of CoNLL2003, WNUT2017, and JNLPBA datasets.

4.1 Datasets

We evaluate the proposed model on three benchmark datasets including CoNLL2003 [22], WNUT2017 [5] and JNLPBA [4].

- CoNLL2003 - It is collected from Reuters news articles. Four different types of named entities including *PER*, *LOC*, *ORG* and *MISC* are defined by the CoNLL 2003 NER shared task.
- WNUT2017 - It is a set of noisy user-generated text including YouTube comments, StackExchange posts, Twitter text, and Reddit comments. Six types of entities including *PER*, *LOC*, *Group*, *Creative_work*, *Corporation* and *Product* are annotated.
- JNLPBA - It is collected from MEDLINE abstracts. Five types of entities including *DNA*, *RNA*, *protein*, *cell_line* and *cell_type* are annotated.

Table 2 presents the statistics of these datasets.

4.2 Baseline Models

We evaluate the proposed Ba-BNN model against the following baseline models:

- BiLSTM-Softmax - This model utilizes BiLSTM to learn the contextual representation of words, and then a Multi-Layer Perceptron with Softmax is used as the label decoder layer to infer decoder tags.
- BiLSTM-CRF [14] - This model uses CRF instead of Multi-Layer Perceptron with Softmax in BiLSTM-Softmax.

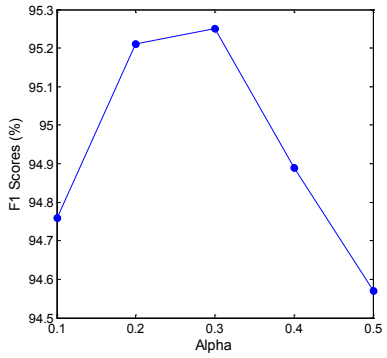

Figure 5: Experimental results on the development set of CoNLL2003 using different values of α.

- BiLSTM-PN[1] - This model uses BiLSTM as the encoder and another unidirectional LSTM with pointer networks as the decoder for entity boundary detection [16]. Then, the entity chunks generated by the decoder are classified with a Softmax classifier.
- HCRA [17] - This model uses sentence-level and document-level representations to augment the contextualized representation.
- ELMo [19] - This model uses a deep bidirectional language model to learn contextualized word representation on a large text corpus.
- BERT [6] - This model learns contextualized word representation based on a bidirectional transformer.
- CS Embeddings [1] - This model uses BiLSTM-CRF with character-level contextualized representations.
- MRC [17] - This model formulates the NER task as a machine reading comprehension task.

4.3 Parameter Settings

Our proposed Ba-BNN model is implemented in the PyTorch framework. We use 300-dimensional pre-trained Glove word embeddings [2] [18]. The char embeddings and word feature embeddings are initialized randomly as 50-dimensional and 25-dimensional vectors, respectively. When training the model, both of the embeddings are updated along with other parameters. We use Adam optimizer [11] for training with a mini-batch. We set the learning rate to 0.001, dropout rate to 0.5, the hidden layer size to 400, and the gradient clipping to 5. The α of multitask loss is tuned during the development process. The value of α is set to 0.3. We report the results based on the best performance on the development set. All of our experiments are conducted on the same machine with 8-cores of Intel(R) Xeon(R) E5-1630 CPU@3.70GHz and two Nvidia GeForce-GTX GPU.

4.4 Parameter Study

We investigate the impact of the hyperparameter α (see Eq. 7) on the development set of CoNLL2003. To this end, we gradually vary α from 0.1 to 0.5 with an increment of 0.1 in each step. As shown in Fig. 5, we find that our model has achieved the best performance when α=0.3 by empirical results. Therefore, we set α=0.3 for all experiments thereafter.

4.5 Experimental Results

Table 3 shows the experimental results of Ba-BNN and the baseline models. From Table 3, when comparing with models without using any language models or external knowledge, we observe that our model outperforms all the compared models in terms of precision, recall and F1 scores, and achieves 0.59%, 3.15% and 2.43% improvements on F1 scores for the CoNLL2003, WNUT2017 and JNLPBA datasets, respectively. Among the compared models, the precision and recall of the BiLSTM-Softmax model are generally lower than other models. This is because the Softmax classifier is unable to tackle the problem of boundary tag sparsity through the simple Maximum Entropy [10]. Moreover, as it only uses a single decoding process, it is unable to make use of the global decoding information. In addition, we also observe that BiLSTM-CRF performs slightly better than BiLSTM-PN since the use of pointer networks suffers from the boundary error propagation problem during boundary detection and entity type classification. HCRA is the current state-of-the-art model for the NER task by fusing sentence-level and document-level representations for global contextualized representation. Although the contextualized representation helps reduce the problem on lacking of global information to some extent, it still suffers from the problem on boundary tag sparsity since the model is built basd on the BiLSTM-CRF architecture. Our Ba-BNN model achieves the best performance as it is capable of tackling the common NER problems as discussed in Section 1. Also, we observe that "Ba-BNN-CG" achieves the best performance when compared with "Ba-BNN-Naive" (with 0.46% improvements in F1) and "Ba-BNN-SG" (with 0.02% improvements in F1). It is because the bidirectional decoder in "Ba-BNN-CG" has fully utilized the contextual information to generate more accurate entity boundaries.

When pre-trained language models such as ELMo and BERT are incorporated, all the models have achieved better performance results. In particular, we observe that our Ba-BNN model has achieved 1.17%, 3.43% and 2.9% improvements on the F1 scores for the CoNLL-2003, WNUT2017 and JNLPBA datasets, respectively when compared with the other models. Similarly, "Ba-BNN-CG+BERT" also performs better than the other two strategies (i.e., Naive and SG) with the pre-trained language model. Overall, our proposed model has achieved the best performance results when compared with other models. It is consistent with the performance results discussed earlier on comparison with models without using any pre-trained language models in this section.

4.6 Ablation Study

To show the importance of each component of our proposed model, we conduct an ablation experiment including bidirectional decoder, boundary-aware binary classifier and boundary retraining strategy. We choose the model Ba-BNN-CG+BERT as an example to show the ablation study and the results are reported in Table 4. As shown in Table 4, all these components contribute significantly to the effectiveness of our model.

[1] In [16], the pointer networks is used for detecting entity boundaries only. We reproduce this work and add one Softmax layer for the NER task.
[2] http://nlp.stanford.edu/projects/glove/

Model	CoNLL2003			WNUT2017			JNLPBA		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
BiLSTM-Softmax	88.53	90.21	89.36	48.99	26.07	34.03	72.20	67.74	69.90
BiLSTM-CRF	90.88	90.62	90.75	50.86	35.50	41.81	73.08	71.56	72.31
BiLSTM-PN	90.34	90.31	90.32	54.23	30.43	38.98	67.72	74.90	71.13
HCRA	-	-	91.96	-	-	-	-	-	-
Ba-BNN-Naive (ours)	92.60	91.59	92.09	57.44	34.43	43.05	71.02	75.28	73.09
Ba-BNN-SG (ours)	92.84	92.23	92.53	56.02	36.89	44.49	70.56	77.79	74.00
Ba-BNN-CG (ours)	93.11	91.99	92.55	57.99	36.71	44.96	72.77	76.82	74.74
+ Language Models / External knowledge									
ELMo	-	-	92.22	-	-	45.33	71.18	77.68	74.29
BERT	-	-	92.80	-	-	46.1	70.73	80.36	75.24
CS Embeddings	92.37	93.12	92.74	-	-	-	71.18	77.68	74.29
MRC	92.33	94.61	93.04	-	-	-	-	-	-
BiLSTM-PN+BERT	92.02	92.45	92.23	56.82	36.87	44.72	68.56	77.32	72.68
HCRA+BERT	-	-	93.37	-	-	-	-	-	-
Ba-BNN-Naive+BERT (ours)	93.88	94.01	93.94	59.98	39.51	47.64	75.05	79.47	77.20
Ba-BNN-SG+BERT (ours)	94.09	94.72	94.40	59.17	42.46	49.44	74.55	81.97	78.08
Ba-BNN-CG+BERT (ours)	94.37	94.72	**94.54**	60.24	42.06	**49.53**	75.27	81.23	**78.14**

Table 3: Experimental results on three benchmark datasets.

	CoNLL2003			WNUT2017			JNLPBA		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
Ba-BNN-CG+BERT	94.37	94.72	94.54	60.24	42.06	49.53	75.27	81.23	78.14
w/o BERT	93.11	91.99	92.55	57.99	36.71	44.96	72.77	76.82	74.74
w/o Bi-decoder	93.87	94.33	94.10	59.19	41.16	48.56	73.29	80.33	76.65
w/o Binary classifier	92.12	92.75	92.43	57.31	38.32	45.93	69.72	78.47	73.84
w/o Boundary retraining	94.42	93.89	94.15	60.32	40.37	48.37	75.49	80.14	77.75

Table 4: Experimental results of the ablation study of the Ba-BNN model.

We analyze the results based on CoNLL2003, which will have the similar trend as for other datasets. Overall, the pre-trained word embedding (i.e., BERT), bidirectional decoder, boundary-aware binary classifier and boundary retraining strategy can help improve the effectiveness of the proposed model by approximately 2.0%, 0.4%, 2.1% and 0.4% respectively in terms of F1 score. The discussion on the effectiveness of each component is given as follows:

- The pre-trained BERT embeddings can provide better word representations and improve boundary detection than random initialization in model training. As such, boundaries can be detected more accurately for extracting entities.
- The bidirectional decoder improves the precision and recall by 0.5% and 0.4% respectively. It has shown the capability of the bidirectional decoder in capturing global information.
- The boundary-aware binary classifier has improved the precision by 2.3% and recall by 2.0%. This is because the binary classifier can help model training in two aspects. Firstly, it can capture the global information as the input to decoders. Secondly, the boundary-aware global information can further alleviate the error propagation problem in decoding.
- The boundary retraining strategy has helped the recall to improve by around 0.8% and the precision stays quite stable. This shows that our boundary retraining strategy can help alleviate the problem of boundary error propagation.

Overall, the different components of the proposed model can work effectively with each other with multitask training and enable the model achieve the state-of-the-art performance for the NER task.

4.7 Performance Comparison with Pointer Networks

As our Ba-BNN model is constructed on top of the pointer networks [24] (PN) architecture, we compare the performance of our proposed model with pointer networks which is shown in Figure 6. The experiment is conducted based on the CoNLL2003 test dataset. We group the data according to the number of entities from 1 to >5 in a sentence in the dataset and report the F1 score for each group. We observe that the proposed Ba-BNN model consistently outperforms the pointer networks-based model in each group. The difference gets bigger when the number of entities in a sentence becomes larger. This is due to the error propagation problem in PN that could accumulate when more entities exist in a sentence. In contrast, our boundary-aware bidirectional decoding mechanism can help tackle this problem effectively.

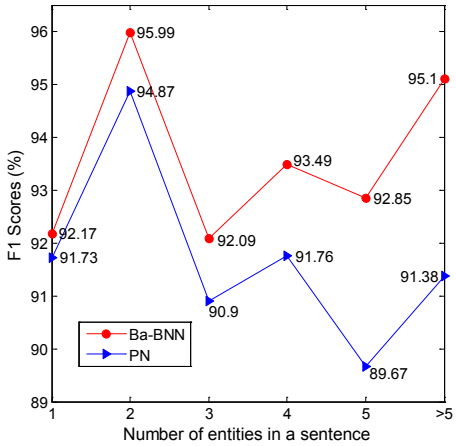

Figure 6: Performance comparison between Ba-BNN and pointer networks mechanism.

5 CONCLUSION

In this paper, we have investigated the problem of the NER task and proposed a novel Boundary-aware Bidirectional Neural Networks (Ba-BNN) which integrates a suite of techniques to help alleviate the three major problems that occurred in the current neural-based NER approaches. We have conducted extensive experiments on three NER benchmark datasets. The experimental results have shown that among the state-of-the-art methods, our proposed Ba-BNN model has achieved the best performance. In the future, we will explore more architectures for the NER task, e.g., reinforcement learning [25], to further improve the performance.

ACKNOWLEDGMENTS

This research has been supported by the National Key R&D Program of China under Grant No. 2020AAA0106600, the National Natural Science Foundation of China under Grants No. 61967003 and 61866038, and the Ministry of Education (MoE) of Singapore under the Academic Research Fund (AcRF) Tier 1 Grant RG135/18.

REFERENCES

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 1638–1649. https://www.aclweb.org/anthology/C18-1139

[2] Shany Barhom, Vered Shwartz, Alon Eirew, Michael Bugert, Nils Reimers, and Ido Dagan. 2019. Revisiting Joint Modeling of Cross-document Entity and Event Coreference Resolution. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 4179–4189.

[3] Jason P.C. Chiu and Eric Nichols. 2016. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4 (2016), 357–370. https://doi.org/10.1162/tacl_a_00104

[4] Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*. 73–78.

[5] Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. 140–147.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*.

[7] Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2681–2690.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[9] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR* abs/1508.01991 (2015). arXiv:1508.01991 http://arxiv.org/abs/1508.01991

[10] Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics* 41, 3 (2015), 385–435.

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[13] Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. 4601–4609.

[14] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).

[15] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[16] Jing Li, Aixin Sun, and Yukun Ma. 2020. Neural Named Entity Boundary Detection. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[17] Ying Luo, Fengshun Xiao, and Hai Zhao. 2020. Hierarchical Contextualized Representation for Named Entity Recognition.. In *AAAI*. 8441–8448.

[18] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[19] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237. https://doi.org/10.18653/v1/N18-1202

[20] 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*.

[21] Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 2670–2680. https://doi.org/10.18653/v1/D17-1283

[22] 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 2003-Volume 4*. 142–147.

[23] Suzushi Tomori, Takashi Ninomiya, and Shinsuke Mori. 2016. Domain Specific Named Entity Recognition Referring to the Real World by Deep Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, 236–242. https://doi.org/10.18653/v1/P16-2039

[24] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*. 2692–2700.

[25] Zheng Wang, Cheng Long, Gao Cong, and Yiding Liu. 2020. Efficient and Effective Similar Subtrajectory Search with Deep Reinforcement Learning. *PVLDB* 13, 11 (2020), 12–25.

[26] Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tie-Yan Liu. 2018. Towards better text understanding and retrieval through kernel entity salience modeling. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 575–584.

[27] Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 3365–3371.

[28] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*. Springer, 662–671.

[29] Andrej Žukov-Gregorič, Yoram Bachrach, and Sam Coope. 2018. Named Entity Recognition With Parallel Recurrent Neural Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, 69–74. https://doi.org/10.18653/v1/P18-2012