Hierarchical Contextualized Representation for Named Entity Recognition

Ying Luo,[1,2,3] **Fengshun Xiao,**[1,2,3] **Hai Zhao**[1,2,3,*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
[3]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China
{kingln, felixxiao}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Named entity recognition (NER) models are typically based on the architecture of Bi-directional LSTM (BiLSTM). The constraints of sequential nature and the modeling of single input prevent the full utilization of global information from larger scope, not only in the entire sentence, but also in the entire document (dataset). In this paper, we address these two deficiencies and propose a model augmented with hierarchical contextualized representation: sentence-level representation and document-level representation. In sentence-level, we take different contributions of words in a single sentence into consideration to enhance the sentence representation learned from an independent BiLSTM via label embedding attention mechanism. In document-level, the key-value memory network is adopted to record the document-aware information for each unique word which is sensitive to similarity of context information. Our two-level hierarchical contextualized representations are fused with each input token embedding and corresponding hidden state of BiLSTM, respectively. The experimental results on three benchmark NER datasets (CoNLL-2003 and Ontonotes 5.0 English datasets, CoNLL-2002 Spanish dataset) show that we establish new state-of-the-art results.

Introduction

Named Entity Recognition (NER) is one of the fundamental tasks in natural language processing (NLP) that intends to identify words or phrases as the proper names of PER (Person), ORG (Organization), LOC (Location), etc. Currently, most state-of-the-art NER systems (Huang, Xu, and Yu 2015; Lample et al. 2016; Ma and Hovy 2016; Chiu and Nichols 2016) employ BiRNNs, specially BiLSTM (Hochreiter and Schmidhuber 1997) as the encoder to extract the sequential information.

BiLSTM architectures exist limitations in making full use of global information. First, at each time step, BiLSTM takes current word embedding and past summary states as inputs, making it difficult to capture sentence-level information. (Zhang, Liu, and Song 2018) simultaneously model the sub-states for individual words and an overall sentence-level state. (Liu et al. 2019b) use a global contextual encoder and mean pooling strategy to capture sentence-level features, though they ignore the different importance of words in the same sentence. Second, though BiLSTM updates the parameters with the iteration of all training instances, it only consumes one instance during both training and predicting. This nature prevents the model from effectively capturing document (dataset)-level information, e.g. for a unique token, its representations in training instances are indicative for recognizing the concerned token. (Akbik, Bergmann, and Vollgraf 2019) use a pooling operation on different contextualized embeddings to generate global word representations. While they only consider the changes of embeddings for each unique word.

In this paper, we propose a hierarchical contextualized representation architecture to enhance NER modeling. For sentence-level representation, inspired by (Wang et al. 2018), we embed labels in the same space with word embeddings, and label embeddings are learned from attention mechanism computed with word embeddings to ensure that, each word embedding is much closer to their corresponding label embedding, and farther to other label embeddings.

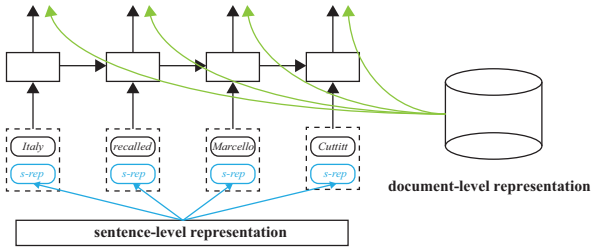

Figure 1: Incorporating hierarchical contextualized representation for NER. The sentence-level representation is assigned to each token and fed to the sequence labeling BiLSTM encoder. The document-level representation is fused with the hidden state of the BiLSTM and fed to the decoder.

*Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), Key Projects of National Natural Science Foundation of China (U1836222 and 61733011).

Then, the similarity between a word embedding and its nearest label embedding is regarded as a confidence score for this word. Hence, words with higher confidence scores contribute more to sentence-level representation. The sentence-level representations are then assigned to each token and fed to the encoder as shown in Figure 1. For document-level representation, we adopt a key-value memory component (Miller et al. 2016) which memorizes all the word embeddings of training instances and their corresponding representations. The attention mechanism is adopted to compute the output of the memory component. The retrieved document-level representation is fused with the original hidden state and fed to the decoder as shown in Figure 1. In this case, the training instances are not only used to train the model parameters, but also involved in inference.

To verify the effectiveness our model, we conduct extensive experiments on three benchmark NER datasets. Experimental results on these benchmarks suggest that our model can achieve state-of-the-art performance on CoNLL-2003 (91.96 F_1 without external knowledge and 93.37 F_1 with BERT), OntoNotes 5.0 (87.98 F_1 without external knowledge and 90.30 F_1 with BERT), 87.08 F_1 on CoNLL-2002, meaning that our model truly learns and benefits from useful contextualized representations.

Our contributions in this paper are summarized as follows.

- We are the first to introduce hierarchical contextualized representations, namely sentence-level and document-level representation, for NER to take full advantage of non-local information.
- We introduce the label embedding attention mechanism for sentence-level representation and propose an effective approach to distill document-level information using key-value memory network.
- The evaluation results on three benchmark NER datasets show that our model outperforms all previously reported results without external knowledge. Furthermore, with pre-trained language model BERT, we establish new state-of-the-art results on CoNLL-2003 and ontonotes 5.0 datasets.

Related work

Neural Named Entity Recognition Recently, with the development of deep neural network in a wide range of NLP tasks (He et al. 2018; He, Li, and Zhao 2019; Zhou and Zhao 2019; Xiao et al. 2019; Zhang et al. 2020a; 2020b), neural network based models build reliable NER systems without hand-crafted features or task-specific knowledge. (Huang, Xu, and Yu 2015) firstly proposed the BiLSTM-CRF architecture, which is used by most state-of-the-art models. Later, character-level embeddings are concatenated to enhance the representation of rare and out-of-vocabulary words, these embeddings are generated with LSTM (Lample et al. 2016), CNN (Ma and Hovy 2016), and recently IntNet (Xin et al. 2018). (Tran, MacKinlay, and Jimeno Yepes 2017) stack BiLSTMs with residual connections between different layers of BiLSTM to add more representational power. More recently, pre-trained language models from huge corpus are adopted to enhance the representation of words (Peters et al. 2018; Akbik, Bergmann, and Vollgraf 2019; Devlin et al. 2019).

Sentence-level Representation has been adopted to eliminate the limitations of RNNs due to their sequential nature. (Yang, Zhang, and Dong 2017) leverage RNN models to learn sentence-level patterns for NER reranking. (Zhang, Liu, and Song 2018) model the sub-states for individual words and an overall sentence-level state simultaneously to capture local and non-local contexts. (Chen et al. 2019) use contextual layer and relation layer to model the relations between words in sentences, and then use gates to fuse local context features into global ones. (Liu et al. 2019b) simplify sentence-level state to average of the hidden states of each individual word from an independent global contextual encoder. Inspired by (Wang et al. 2018) which use label information to construct text-sequence representations, we adopt label embedding attention to enhance the sentence-level representation learned from an independent BiLSTM. The use of sentence-level information in (Liu et al. 2019b) can be seen as a special case of our model where the attention weight vector is a uniform distribution that assigns equal probabilities to all the words in the sentence.

Document-level Representation (Qian et al. 2019) considers the dependency structure of word sequence as the global information. (Akbik, Bergmann, and Vollgraf 2019) dynamically aggregates contextualized embeddings for each unique string and then use a pooling operation to generate a global word representation from these contextualized instances for NER. Different from their work which only uses the contextual word embeddings, our memory component uses the key-value memory networks to memorize the word representations (key) and the hidden states (value) from the sequence labeling encoder. The attention mechanism is then called to calculate the document-level representation.

Model

This section presents our NER model in detail. The overall model architecture is shown in Figure 2, which consists of four components: a decoder (top part), a sequence labeling encoder (upper right part), a sentence-level encoder (bottom right part), and a document-level encoder (left part).

Baseline Model

We adopt the IntNet-LSTM-CRF model proposed by (Xin et al. 2018) as our baseline, which consists of three parts: representation module, sequence labeling encoder and decoder module.

Token Representation Given a sequence of N tokens $X = \{x_1, x_2, ..., x_N\}$, for each word x_i, we concatenate the word-level and character-level embedding as the joint word representation $x_i = [w_i; c_i]$. w_i is the pre-trained word embedding. The character-level embedding c_i is learned from IntNet, which is a funnel-shaped wide convolutional neural architecture for learning representations of the internal structure of words. The network comprises of L convolutions, which implies $(L-1)/2$ convolutional blocks. In each convolutional block, the first layer is the $N \times 1$ convolution

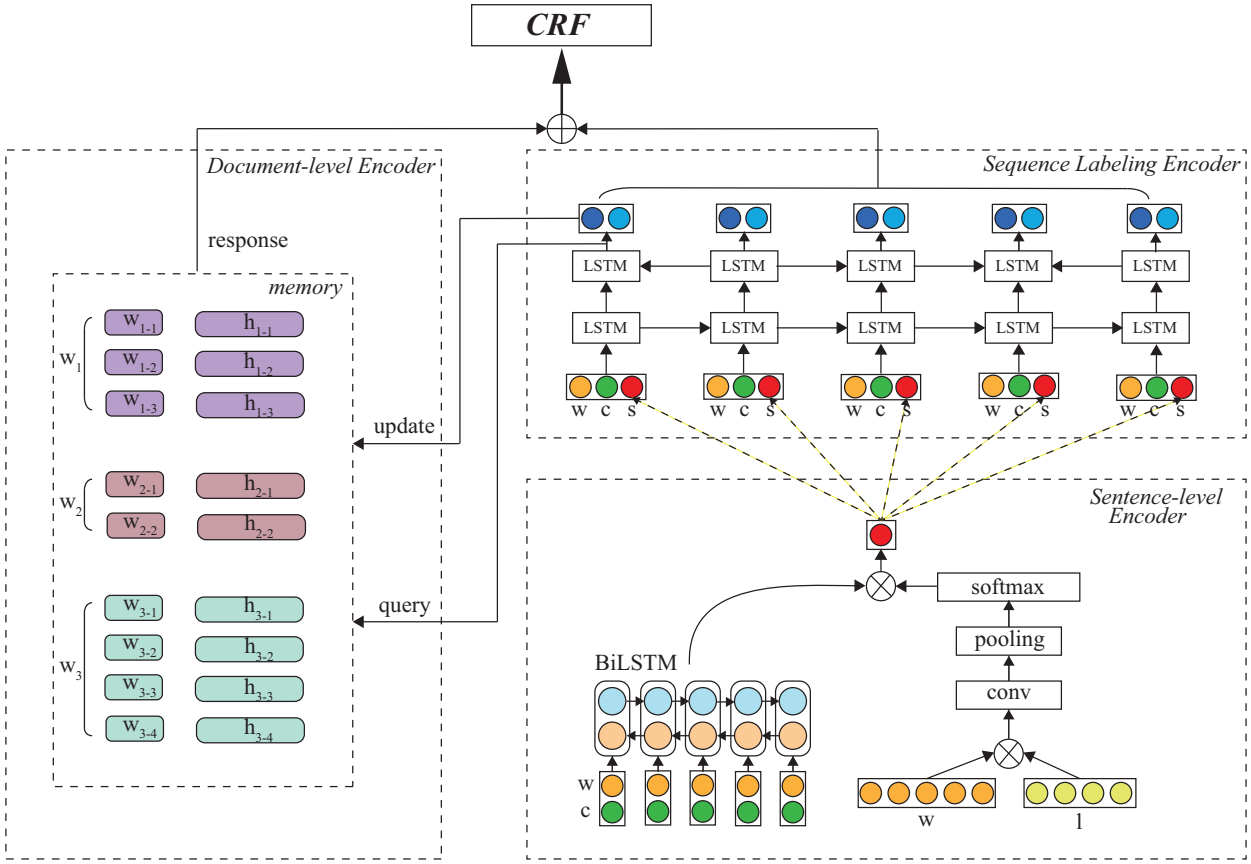

Figure 2: The main architecture of our NER model. The sequence labeling encoder (upper right) generates representations for the decoder and updates the memory component. The sentence-level encoder (bottom right part) generates the sentence-level representation, and the memory network (left part) computes the document-level contextualized information, in which the same color represents the memory slots for the unique word.

which transforms the input, then the concatenation of convolutions with different kernel sizes in the second layer is fed to the next convolutional block. Direct connections from every other layer to all subsequent layers are used like dense connections.

Sequence Labeling Encoder The concatenation of word-level and character-level embeddings $x_i = [w_i; c_i]$ is then fed into the sequence labeling BiLSTM, which represents the sequential information at each step.

$$\begin{aligned} h_i &= [\overrightarrow{h_i}; \overleftarrow{h_i}] \\ \overrightarrow{h_i} &= LSTM(x_i, \overrightarrow{h}_{i-1}; \overrightarrow{\theta}) \\ \overleftarrow{h_i} &= LSTM(x_i, \overleftarrow{h}_{i-1}; \overleftarrow{\theta}) \end{aligned} \tag{1}$$

where $\overrightarrow{\theta}$ and $\overleftarrow{\theta}$ are trainable parameters, respectively.

Decoder Conditional random field (CRF) (Lafferty, Mccallum, and Pereira 2001) has been widely used in state-of-the-art NER models (Lample et al. 2016; Ma and Hovy 2016) to help make decisions when considering strong connections between output tags. During decoding, the Viterbi algorithm is applied to search the label sequence with the highest probability. For $y = \{y_1, ..., y_N\}$ being a predicted sequence of labels with same length as x. We define its score as:

$$sc(x, y) = \sum_{i=0}^{N-1} Tr_{y_i, y_{i+1}} + \sum_{i=1}^{N} P_{i, y_i} \tag{2}$$

where $Tr_{y_i, y_{i+1}}$ represents the transmission score from the y_i to y_{i+1}, P_{i,y_i} is the score of the j^{th} tag of the i^{th} word from the sequence labeling encoder.

The CRF model defines a family of conditional probability $p(y|x)$ over all possible tag sequences y:

$$p(y|x) = \frac{\exp^{sc(x,y)}}{\sum_{\tilde{y} \in y} \exp^{sc(x,\tilde{y})}} \tag{3}$$

during training, we consider the maximum log probability of the correct sequence of tags. While decoding, we search the label sequence with maximum score:

$$y^* = \arg\max_{\tilde{y} \in y} sc(x, \tilde{y}) \tag{4}$$

Sentence-level Representation

Sentence-level information has been shown highly useful for model sequence (Zhang, Liu, and Song 2018; Liu et al. 2019b). We adopt an independent BiLSTM to generate contextualized features, which takes word representation $x_i = [w_i; c_i]$ as the input, we denote the hidden states of this BiLSTM as $v \in \mathbb{R}^{N \times d_s}$, where N is the length of the sequence and d_s is the hidden size. Considering that words may contribute differently to the sentence-level representation, we adopt label embedding attention (Wang et al. 2018) to get an attention score for the entire sentence and then transform the hidden states $v \in \mathbb{R}^{N \times d_s}$ into a fixed-sized sentence-level representation $s \in \mathbb{R}^{d_s}$.

As shown in Figure 2, we embed all the label types (e.g. LOC, PER, etc.) in the same space as the word embeddings. We denote the label embeddings as $l = [l_1, l_2, ...l_P], l \in \mathbb{R}^{P \times d_w}$, where P is the number of labels, d_w is the dimension of the word embeddings. We train on the training instances to ensure that, each word embedding is more closer to their corresponding label embedding, and farther to other label embeddings. For example, the token *Italy* is labeled as LOC type in the example of Figure 1, we attempt to make it closer to the label embedding of LOC, and farther to other label embeddings (e.g. the label embedding of PER). The cosine similarity [1] $e(x_i, l_j)$ between the word embeddings x_i and the label embedding l_j can be taken to measure the confidence score of this word-label pair:

$$e(x_i, l_j) = \frac{x_i^T l_j}{\|x_i\| \|l_j\|} \tag{5}$$

We use convolutional neural network (CNN) to capture the relative spatial information among consecutive words in the sentence. Further, the largest confidence score $m_i \in \mathbb{R}^P$ between the i-th word and all labels is obtained by a max-pooling operation:

$$m_i = max(W^T \begin{bmatrix} e(i - \frac{k-1}{2}, :) \\ \cdots \\ e(i + \frac{k-1}{2}, :) \end{bmatrix} + b) \tag{6}$$

where $W \in \mathbb{R}^k$ and $b \in \mathbb{R}^P$ are trainable parameters, k is the kernel size, max denotes max pooling.

The attention (confidence) score $\beta \in \mathbb{R}^N$ for the entire sentence is:

$$\beta = softmax(m) \tag{7}$$

The sentence-level representation $s \in \mathbb{R}^{d_s}$ can be simply obtained via averaging the hidden states $v \in \mathbb{R}^{N \times d_s}$, weighted by the attention score calculated above:

$$s = \sum_{i=1}^{N} \beta_i v_i \tag{8}$$

The sentence-level representation $s \in \mathbb{R}^{d_s}$ is then concatenated with the word representation $x_i' = [x_i; s]$ and fed to the sequence labeling encoder. Note that the training and test process are the same. We use the label embeddings $l \in \mathbb{R}^{P \times d_w}$ of all labels, not the ground-truth label embedding. The intuition for using the label embedding attention is that each word in the sentence contributes differently to sentence-aware representation. The similarity between each word embedding and its nearest label embedding can be regarded as the confidence score of this word-label pair. Words with higher confidence score should contribute more to the sentence-level contextualized representation.

[1] The reason for using cosine similarity is the same as the next subsection, and will be analyzed later.

Document-level Representation

In terms of memory network, we introduce document-aware representations of the unique word in training instances as an extra knowledge source to help the prediction. Memory network was originally proposed by (Weston, Chopra, and Bordes 2014) in the domain of question answering (QA) for prediction, where the long-term memory acts as a dynamic knowledge base. (Miller et al. 2016) further introduces a key-value memory networks, which utilize different encodings in the addressing and output stages. The keys are designed to help match the question, while the values are to generate the response.

We adopt the key-value memory component ***M*** to memorize document-level contextualized representation. Memory slots are defined as pairs of vectors $(k_1, v_1), ..., (k_m, v_m)$. In each single slot, the key represents the word embedding w_i, and the value is the corresponding hidden states h_i from sequence labeling encoder for each token in training instances. So the same word may occupy in many different slots because of changing embeddings and representations under different contexts. Table 1 shows an example of using training instances to help indicate the NE type of queried token.

Memory Update The word embeddings are fine-tuned during training and used to update the key part of the memory. The sequence labeling encoder generates the hidden states to update the value part. Supposing the states of the i-th token is changed after computation, the i-th slot in the memory M will be rewritten. Each memory slot will be updated once in one epoch.

Memory Query For the i-th word in the sentence, we distill all the contextualized representations for this word in the memory M through an inverted index that finds a subset $(k_{sub_1}, v_{sub_1}), ..., (k_{sub_T}, v_{sub_T})$ of size T, where the inverted index records the positions of the unique word in the memory M as shown in Table 1. T represents the number of occurrences of this word among the training instances.

The attention operation is called to compute the weight of document-level representation. For the unique word, the memory key $k_j \in [k_{sub_1}; ...; k_{sub_T}]$ is used as the attention key, the memory value $v_j \in [v_{sub_1}; ...; v_{sub_T}]$ is used as the attention value. Then the embedding w_{q_i} of the queried word serves as the attention query q_i. Here, we consider three compatibility functions $u_{ij} = o(q_i, k_j)$:

(1) dot-product attention

$$o_1(q_i, k_j) = q_i k_j^T \tag{9}$$

(2) scaled dot-product attention (Vaswani et al. 2017)

$$o_2(q_i, k_j) = \frac{q_i k_j^T}{\sqrt{d_w}} \tag{10}$$

Test instance
Italy *recalled Marcello Cuttitt.*
Training instances
1. *ORVIETO* (0), ***Italy*** **(1)** *1996-08-24* (2). 2. *Rohrabacher* (3) *had* (4) *recently* (5) *visited* (6) ***Italy*** **(7)** 3. *Andrea* (8) *Ferrigato* (9) *of* (10) ***Italy*** **(11)** *sprinted* (12)...
Inverted index
Italy: [1, 7, 11, ...]

Table 1: Query operation for the word *Italy*. The numbers in parentheses indicate slot index of tokens in memory M. The memory slots of these bold tokens in training instances are retrieved according to the inverted index for *Italy*.

and (3) cosine similarity

$$o_3(q_i, k_j) = \frac{q_i k_j^T}{\|q_i\|\|k_j\|} \tag{11}$$

where d_w represents the dimension of word embeddings.

Memory Response The document-level representation is computed as:

$$\begin{aligned} \alpha_{ij} &= \frac{\exp(u_{ij})}{\sum_{z=1}^{T} \exp(u_{iz})} \\ r_i &= \sum_{j=1}^{T} \alpha_{ij} v_j \end{aligned} \tag{12}$$

Then the fusion representation $g_i \in \mathbb{R}^{d_h}$ of the original hidden representation and this document-level representation are fed to the CRF layer, where d_h is the hidden size of the sequence labeling encoder.

$$g_i = \lambda h_i + (1 - \lambda) r_i \tag{13}$$

where λ is a hyperparameter, indicating how much document-aware information is adopted, 0 for document-level representation only and 1 for discarding all document-level information at all.

Experiment

Dataset

Our proposed representations are evaluated on three benchmark NER datasets: CoNLL-2003 (Sang and De Meulder 2003) and OntoNotes 5.0 (Pradhan et al. 2013) English NER datasets, CoNLL-2002 Spanish NER (Tjong Kim Sang 2002) dataset.

- **CoNLL-2003 English NER** consists of 22,137 sentences totally and is split into 14,987, 3,466 and 3,684 sentences for the training, development set and test sets, respectively. It is tagged with four linguistic entity types (PER, LOC, ORG, MISC).
- **CoNLL-2002 Spanish NER** consists of 11,752 sentences totally and is split into 8,322, 1,914 and 1,516 sentences for the training, development and test sets, respectively. It is also tagged with four linguistic entity types (PER, LOC, ORG, MISC).
- **OntoNotes 5.0** consists of 76,714 sentences from a wide variety of sources (magazine, telephone conversation, newswire, etc.). Following (Chiu and Nichols 2016; Chen et al. 2019), we use the portion of the dataset with gold-standard named entity annotations, and thus exclude the New Testaments portion. It is tagged with eighteen entity types (PERSON, CARDINAL, LOC, PRODUCT, etc.).

Models	F_1
(Lample et al. 2016)	90.94
(Ma and Hovy 2016)	91.21
(Yang, Zhang, and Dong 2017)	91.62
(Liu et al. 2018)	91.24 ± 0.12
(Yang and Zhang 2018)	91.35
(Zhang, Liu, and Song 2018)	91.57
(Xin et al. 2018)	91.64 ± 0.17
(Liu et al. 2019a)	91.10
(Chen et al. 2019)	91.44 ± 0.10
(Qian et al. 2019)	91.74
(Liu et al. 2019b) [2]	91.54
Ours	**91.96 ± 0.03**
+ Language Models / External knowledge	
(Chiu and Nichols 2016)†	91.62 ± 0.33
(Liu et al. 2018)	91.71 ± 0.10
(Peters et al. 2018) (ELMo)	92.20
(Clark et al. 2018)	92.61
(Devlin et al. 2019) (BERT)	92.80
(Akbik, Blythe, and Vollgraf 2018)†	93.09
(Akbik, Bergmann, and Vollgraf 2019)†	93.18
(Liu et al. 2019b) (BERT) [2]	93.23
Ours + BERT	**93.37 ± 0.04**

Table 2: F_1 scores on CoNLL-2003. † refers to models trained on both training and development datasets.

Metric We use the BIOES sequence labeling scheme instead of BIO for these three datasets during training. As for test, we convert the prediction results back to the BIO scheme and use the standard conlleval script to compute the F_1 score.

Setup

Pre-trained Word Embeddings. For the CoNLL-2003 and OntoNotes 5.0 English datasets, we use the publicly available pre-trained $100D$ GloVe (Pennington, Socher, and Manning 2014) embeddings. For CoNLL-2002 Spanish dataset, we train $64D$ GloVe embeddings with the minimum frequency of occurrence as 3, and the window size of 5. The

[2]Through personal communication, the authors confirmed that they directly tested the BIOES tagged results with the official conlleval script (which can only works for BIO tagged entities), giving the results reported in their paper 91.96 / 93.47, while our re-evaluation results are 91.54 / 93.23 with strict BIO tag converting from the released file by the authors.

Models	F_1
(Gillick et al. 2015)	82.95
(Lample et al. 2016)	85.75
(Yang, Salakhutdinov, and Cohen 2017)	85.77
(Xin et al. 2018)	86.68 ± 0.35
Ours	**87.08 ± 0.16**

Table 3: F_1 scores on CoNLL-2002.

Models	F_1
(Durrett and Klein 2014)	84.04
(Chiu and Nichols 2016)	86.28 ± 0.26
(Shen et al. 2018)	86.63 ± 0.49
(Strubell et al. 2017)	86.84 ± 0.19
(Ghaddar and Langlais 2018)	87.44
(Chen et al. 2019)	87.67 ± 0.17
Ours	**87.98 ± 0.05**
+ Language Models / External knowledge	
(Ghaddar and Langlais 2018)	87.95
(Clark et al. 2018)	88.88
(Akbik, Bergmann, and Vollgraf 2019)[3]	89.71
Ours + BERT	**90.30**

Table 4: F_1 scores on OntoNotes 5.0.

word embeddings are fine-tuned during training.

Character Embeddings. We train the IntNet character embeddings (Xin et al. 2018). The dimension of character embeddings is 32, which is randomly initialized, the filter size of the initial convolution is 32 and that of other convolutions is 16. Different from (Xin et al. 2018), we set filters as size [3; 5] for all the kernels, and the number of convolutional layers is 7.

Parameters. We follow the work of (Yang and Zhang 2018), and conduct optimization with the stochastic gradient descent [2]. The batch size is set as 10, the initial learning rate is set to 0.015 and will shrunk by 5% after each epoch. The hidden size of sequence labeling encoder and the sentence-level encoder are set as 256 and 128, respectively. We apply dropout to embeddings and hidden states with a rate of 0.5. The λ used to fuse original hidden state and document-level representation is set as 0.3 empirically. For each type of NEs, we randomly select hundreds of NEs, and calculate the average of the word embeddings as its label embedding.

Results and Comparisons

Tables 2, 3, 4 compare our model to existing state-of-the-art approaches on the three benchmark datasets. Our model surpasses previous state-of-the-art approaches on all the three datasets. On CoNLL-2003 dataset, we compare our model with the state-of-the-art models, including the models that use global information to enhance the representation (Yang, Zhang, and Dong 2017; Zhang, Liu, and Song 2018; Qian et al. 2019; Liu et al. 2019b). We also incorporate pre-trained language model BERT (Devlin et al. 2019) for fair comparisons with the models which also use pre-trained language models or other external knowledge. Some of the results (Akbik, Blythe, and Vollgraf 2018; 2018) are not comparable to our results directly, because their final models are trained on both training and development datasets. On CoNLL-2002 Spanish dataset, our model achieves 87.08 F_1 score without external knowledge, which surpasses previous best score by 0.4. Considering that the above two datasets are relatively small, we further conduct experiment on a much more large OntoNotes 5.0 dataset, which also has more entity types. We compare our model with the previous model that also reported results on it (Chiu and Nichols 2016; Shen et al. 2018; Ghaddar and Langlais 2018). As shown in Table 4, our model shows a significant advantage on this dataset, which outperforms previous state-of-the-art results substantially at 87.08 (+0.31) without BERT, and 90.30 (+0.59) with BERT. More notably, our model without external knowledge surpasses the previous model (Ghaddar and Langlais 2018), which use extra lexicon information of 120 entity types from Wikipedia. Overall, the comparisons on these three benchmark datasets well demonstrate that our model truly learns and benefits from useful sentence-level and document-level representation without the support from external knowledge.

[2]Code will be available at https://github.com/cslydia/Hire-NER.

[3]The authors of (Akbik, Bergmann, and Vollgraf 2019) released the result of 89.3 in their github https://github.com/zalandoresearch/flair, 89.71 is our re-implement result.

	CoNLL03	CoNLL02	OntoNotes
base model	91.60	86.65	87.58
+ sentence-level	91.80	86.95	87.86
+ document-level	91.79	86.76	87.81
+ ALL	**91.96**	**87.08**	**87.98**

Table 5: Ablation study on the three benchmark datasets.

	Strategy	F_1	ERR
base model	-	91.60	-
sentence-level	mean-pooling	91.65	0.60
	label-embedding	91.80	2.23
document-level	dot-product	91.63	1.55
	scaled dot-product	91.75	1.79
	cosine similarity	91.79	2.38
ALL	-	**91.96**	**3.81**

Table 6: Comparison of different strategies on CoNLL-2003 dataset. ERR is the relative error rate reduction of our model compared to the baseline.

Ablation Study

In this experiment, we individually adopt two hierarchical contextualized representations to enhance the representation of tokens: sentence-level representation for assigning the sentence state to each token and document-level representation for inference. Table 5 shows the F_1 score raise and relative error reduction brought by each of the two hierarchical representation on the three benchmark datasets. We discover that both sentence-level and document-level repre-

	Baseline			Ours		
	P	R	F_1	P	R	F_1
IV	94.58	93.16	93.87	94.96	93.58	**94.26**
OOTV	93.46	91.57	92.51	94.07	91.85	**92.95**
OOEV	94.12	94.12	94.12	94.12	94.12	94.12
OOBV	88.42	84.81	86.58	88.51	85.56	**87.01**

Table 7: Detailed results on the CoNLL-2003 dataset for IV, OOTV, OOEV, OOBV.

sentations enhance the baseline. By combing these two representations together, we get a larger gain of 0.36 / 0.43 / 0.40, respectively.

We further analyze the two hierarchical representations by adopting different strategies. (Liu et al. 2019b) perform mean pooling over all the tokens to generate sentence-level representation. We further conduct experiments to investigate the three compatibility functions used to employ memorized information. As shown in Table 6, compared with the mean pooling strategy, our label-embedding attention mechanism raises the F_1 score by 0.25. Among the three compatibility functions to compute the weight of query word and memorized slots, cosine similarity performs best, while dot-product performs worst. (Vaswani et al. 2017) use scaled dot-product to counteract the dot products growth in magnitude, showing better than dot product. Cosine similarity calculates the inner product of word vectors with unit length, and can further solve the inconsistency between the embeddings and the similarity measurement. Thus, we eventually adopt cosine similarity as the compatibility function.

Memory Size and Time Consuming

Figure 3 illustrates our model performance and time proportion compared to the baseline with respect to the max queried subset size T for each unique word in the memory query step. For words occurring more than T times in the corpus (these words are more likely to be stop words when T is large), we only randomly select T slots in the subset to compute the document-level representation. For fair comparisons, we keep the IntNet layer, sequence labeling layer and CRF layer the same for all the experiments. The consumed time of our model is only 19% more than the baseline on CoNLL-2003 dataset even with the max memory size as 500. Therefore, our model brings slight increase on time consumption. When T is less than 500, the larger T may incorporate more useful contextualized representation for practice words and improve the results accordingly, when T is larger than 500, which may involve more stop words, our model drops slightly.

Improvement Discussion

Table 7 presents the F_1 score of in-both-vocabulary words (IV), out-of-training-vocabulary words (OOTV), out-of-embedding-vocabulary words (OOEV), and out-of-both-vocabulary words (OOBV) on CoNLL-2003 datset. According to our statistic, 63.40% / 52.43% / 84.68% of the NEs in the test set of CoNLL-2003, CoNLL-2002, and OntoNotes

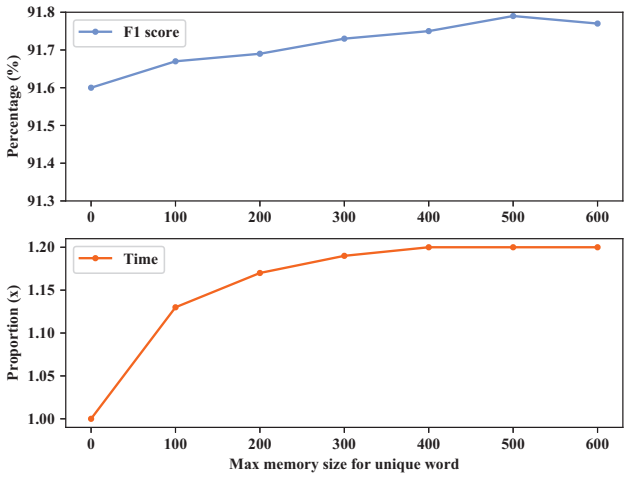

Figure 3: F_1 score and time proportion with respect to the max memory size. Time proportion represents the ratio of our training time compared to baseline.

datasets are located in the IV part, respectively. Therefore, it is of great importance to focus on this part. We adopt memory network to memorize and retrieve the global representations and use the memorized training instances directly to participate in inference, which greatly improves both the precision and recall of the NEs in IV part, in which our model outperforms baseline by 0.39 in terms of F_1 score. For OOV NEs, sentence-level representation can help these concerned tokens aware of the entire sentence, thus enhance the performance. The improvement is 0.44 / 0.43 $F1$ score for OOTV NEs and OOBV NEs, respectively.

Conclusions

In this paper, we adopt hierarchical contextualized representations to enhance the performance of named entity recognition (NER). Our model makes full use of the training instances and the spatial information of the embedding space by incorporating sentence-level representation and document-level representation. We consider the importance of words in the sentences and weight their contributions with the label embedding attention for the sentence-level representation. For words shown in training instances, we memorize the representations of these instances, and involve these representations for inference during test. Empirical results on three benchmark datasets (CoNLL-2003 and Ontonotes 5.0 English datasets, CoNLL-2002 Spanish dataset) show that our model outperforms previous state-of-the-art systems with or without pre-trained language models respectively.

References

Akbik, A.; Bergmann, T.; and Vollgraf, R. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL*.

Akbik, A.; Blythe, D.; and Vollgraf, R. 2018. Contextual string embeddings for sequence labeling. In *COLING*.

Chen, H.; Lin, Z.; Ding, G.; Lou, J.; Zhang, Y.; and Karlsson, B. 2019. GRN: Gated relation network to enhance convolutional neural network for named entity recognition. In *AAAI*.

Chiu, J. P., and Nichols, E. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*.

Clark, K.; Luong, M.-T.; Manning, C. D.; and Le, Q. V. 2018. Semi-supervised sequence modeling with cross-view training. In *EMNLP*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Durrett, G., and Klein, D. 2014. A joint model for entity analysis: Coreference, typing, and linking. *TACL*.

Ghaddar, A., and Langlais, P. 2018. Robust lexical features for improved neural network named-entity recognition. In *COLING*.

Gillick, D.; Brunk, C.; Vinyals, O.; and Subramanya, A. 2015. Multilingual language processing from bytes. *Computer Science*.

He, S.; Li, Z.; Zhao, H.; and Bai, H. 2018. Syntax for semantic role labeling, to be, or not to be. In *ACL*.

He, S.; Li, Z.; and Zhao, H. 2019. Syntax-aware multilingual semantic role labeling. In *EMNLP*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Lafferty, J. D.; Mccallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. In *NAACL*.

Liu, L.; Shang, J.; Ren, X.; Xu, F. F.; Gui, H.; Peng, J.; and Han, J. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*.

Liu, P.; Chang, S.; Huang, X.; Tang, J.; and Cheung, J. C. K. 2019a. Contextualized non-local neural networks for sequence learning. In *AAAI*.

Liu, Y.; Meng, F.; Zhang, J.; Xu, J.; Chen, Y.; and Zhou, J. 2019b. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *ACL*.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Miller, A.; Fisch, A.; Dodge, J.; Karimi, A.-H.; Bordes, A.; and Weston, J. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Pennington, J.; Socher, R.; and Manning, C. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL*.

Pradhan, S.; Moschitti, A.; Xue, N.; Ng, H. T.; Björkelund, A.; Uryupina, O.; Zhang, Y.; and Zhong, Z. 2013. Towards robust linguistic analysis using ontonotes. In *CoNLL*.

Qian, Y.; Santus, E.; Jin, Z.; Guo, J.; and Barzilay, R. 2019. GraphIE: A graph-based framework for information extraction. In *NAACL*.

Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Shen, Y.; Yun, H.; Lipton, Z. C.; Kronrod, Y.; and Anandkumar, A. 2018. Deep active learning for named entity recognition. In *ICLR*.

Strubell, E.; Verga, P.; Belanger, D.; and McCallum, A. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *EMNLP*.

Tjong Kim Sang, E. F. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *CoNLL*.

Tran, Q.; MacKinlay, A.; and Jimeno Yepes, A. 2017. Named entity recognition with stack residual LSTM and trainable bias decoding. In *IJCNLP*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Wang, G.; Li, C.; Wang, W.; Zhang, Y.; Shen, D.; Zhang, X.; Henao, R.; and Carin, L. 2018. Joint embedding of words and labels for text classification. In *ACL*.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Xiao, F.; Li, J.; Zhao, H.; Wang, R.; and Chen, K. 2019. Lattice-based transformer encoder for neural machine translation. In *ACL*.

Xin, Y.; Hart, E.; Mahajan, V.; and Ruvini, J.-D. 2018. Learning better internal structure of words for sequence labeling. In *EMNLP*.

Yang, J., and Zhang, Y. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *ACL*.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Yang, J.; Zhang, Y.; and Dong, F. 2017. Neural reranking for named entity recognition. In *RANLP*.

Zhang, S.; Zhao, H.; Wu, Y.; Zhang, Z.; Zhou, X.; and Zhou, X. 2020a. DCMN+: Dual co-matching network for multi-choice reading comprehension. In *AAAI*.

Zhang, Z.; Wu, Y.; Zhao, H.; Li, Z.; Zhang, S.; Zhou, X.; and Zhou, X. 2020b. Semantics-aware BERT for language understanding. In *AAAI*.

Zhang, Y.; Liu, Q.; and Song, L. 2018. Sentence-state LSTM for text representation. In *ACL*.

Zhou, J., and Zhao, H. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *ACL*.