Multi-Grained Named Entity Recognition

Congying Xia[1,5], **Chenwei Zhang**[1], **Tao Yang**[2], **Yaliang Li**[3]*,
Nan Du[2], **Xian Wu**[2], **Wei Fan**[2], **Fenglong Ma**[4], **Philip Yu**[1,5]
[1]University of Illinois at Chicago, Chicago, IL, USA
[2]Tencent Medical AI Lab, Palo Alto, CA, USA; [3]Alibaba Group, Bellevue, WA, USA
[4]University at Buffalo, Buffalo, NY, USA; [5]Zhejiang Lab, Hangzhou, China
{cxia8,czhang99,psyu}@uic.edu; yaliang.li@alibaba-inc.com
{tytaoyang,kevinxwu,davidwfan}@tencent.com
nandu2048@gmail.com; fenglong@buffalo.edu

Abstract

This paper presents a novel framework, MGNER, for Multi-Grained Named Entity Recognition where multiple entities or entity mentions in a sentence could be non-overlapping or totally nested. Different from traditional approaches regarding NER as a sequential labeling task and annotate entities consecutively, MGNER detects and recognizes entities on multiple granularities: it is able to recognize named entities without explicitly assuming non-overlapping or totally nested structures. MGNER consists of a Detector that examines all possible word segments and a Classifier that categorizes entities. In addition, contextual information and a self-attention mechanism are utilized throughout the framework to improve the NER performance. Experimental results show that MGNER outperforms current state-of-the-art baselines up to 4.4% in terms of the F1 score among nested/non-overlapping NER tasks.

1 Introduction

Effectively identifying meaningful entities or entity mentions from the raw text plays a crucial part in understanding the semantic meanings of natural language. Such a process is usually known as Named Entity Recognition (NER) and it is one of the fundamental tasks in natural language processing (NLP). A typical NER system takes an utterance as the input and outputs identified entities, such as person names, locations, and organizations. The extracted named entities can benefit various subsequent NLP tasks, including syntactic parsing (Koo and Collins, 2010), question answering (Krishnamurthy and Mitchell, 2015) and relation extraction (Lao and Cohen, 2010). However, accurately recognizing representative entities in natural language remains challenging.

Previous works treat NER as a sequence labeling problem. For example, Lample et al. (2016) achieve a decent performance on NER by incorporating deep recurrent neural networks (RNNs) with conditional random field (CRF) (Lafferty et al., 2001). However, a critical problem that arises by treating NER as a sequence labeling task is that it only recognizes non-overlapping entities in a single, sequential scan on the raw text; it fails to detect nested named entities which are embedded in longer entity mentions, as illustrated in Figure 1.

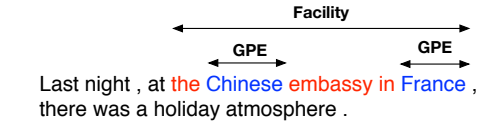

Figure 1: An example from the ACE-2004 dataset (Doddington et al., 2004) in which two GPEs (Geographical Entities) are nested in a Facility Entity.

Due to the semantic structures within natural language, nested entities can be ubiquitous: *e.g.* 47% of the entities in the test split of ACE-2004 (Doddington et al., 2004) dataset overlap with other entities, and 42% of the sentences contain nested entities. Various approaches (Alex et al., 2007; Lu and Roth, 2015; Katiyar and Cardie, 2018; Muis and Lu, 2017; Wang and Lu, 2018) have been proposed in the past decade to extract nested named entities. However, these models are designed explicitly for recognizing nested named entities. They usually do not perform well on non-overlapping named entity recognition compared to sequence labeling models.

To tackle the aforementioned drawbacks, we propose a novel neural framework, named MGNER, for Multi-Grained Named Entity Recognition. It is suitable for tackling both Nested NER and Non-overlapping NER. The idea

*Work was done when the author Yaliang Li was at Tencent America.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1430–1440
Florence, Italy, July 28 - August 2, 2019.

of MGNER is natural and intuitive, which is to first detect entity positions in various granularities via a Detector and then classify these entities into different pre-defined categories via a Classifier. MGNER has five types of modules: Word Processor, Sentence Processor, Entity Processor, Detection Network, and Classification Network, where each module can adopt a wide range of neural network designs.

In summary, the contributions of this work are:

- We propose a novel neural framework named MGNER for Multi-Grained Named Entity Recognition, aiming to detect both nested and non-overlapping named entities effectively in a single model.
- MGNER is highly modularized. Each module in MGNER can adopt a wide range of neural network designs. Moreover, MGNER can be easily extended to many other related information extraction tasks, such as chunking (Ramshaw and Marcus, 1999) and slot filling (Mesnil et al., 2015).
- Experimental results show that MGNER is able to achieve new state-of-the-art results on both Nested Named Entity Recognition tasks and Non-overlapping Named Entity Recognition tasks.

2 Related Work

Existing approaches for recognizing non-overlapping named entities usually treat the NER task as a sequence labeling problem. Various sequence labeling models achieve decent performance on NER, including probabilistic graph models such as Conditional Random Fields (CRF) (Ratinov and Roth, 2009), and deep neural networks like recurrent neural networks or convolutional neural networks (CNN). Hammerton (2003) is the first work to use Long Short-Term Memory (LSTM) for NER. Collobert et al. (2011) employ a CNN-CRF structure, which obtains competitive results to statistical models. Most recent works leverage an LSTM-CRF architecture. Huang et al. (2015) use hand-crafted spelling features; Ma and Hovy (2016) and Chiu and Nichols (2016) utilize a character CNN to represent spelling characteristics; Lample et al. (2016) employ a character LSTM instead. Moreover, the attention mechanism is also introduced in NER to dynamically decide how much information to use from a word or character level component (Rei et al., 2016).

External resources have been used to further improve the NER performance. Peters et al. (2017) add pre-trained context embeddings from bidirectional language models to NER. Peters et al. (2018) learn a linear combination of internal hidden states stacked in a deep bidirectional language model, ELMo, to utilize both higher-level states which capture context-dependent aspects and lower-level states which model aspects of syntax. These sequence labeling models can only detect non-overlapping entities and fail to detect nested ones.

Various approaches have been proposed for Nested Named Entity Recognition. Finkel and Manning (2009) propose a CRF-based constituency parser which takes each named entity as a constituent in the parsing tree. Ju et al. (2018) dynamically stack multiple flat NER layers and extract outer entities based on the inner ones. Such model may suffer from the error propagation problem if shorter entities are recognized incorrectly.

Another series of approaches for Nested NER are based on hypergraphs. The idea of using hypergraph is first introduced in Lu and Roth (2015), which allows edges to be connected to different types of nodes to represent nested entities. Muis and Lu (2017) use a multigraph representation and introduce the notion of mention separator for nested entity detection. Both Lu and Roth (2015) and Muis and Lu (2017) rely on the hand-crafted features to extract nested entities and suffer from structural ambiguity issue. Wang and Lu (2018) present a neural segmental hypergraph model using neural networks to obtain distributed feature representation. Katiyar and Cardie (2018) also adopt a hypergraph-based formulation and learn the structure using an LSTM network in a greedy manner. One issue of these hypergraph approaches is the spurious structures of hypergraphs as they enumerate combinations of nodes, types and boundaries to represent entities. In other words, these models are specially designed for the nested named entities and are not suitable for the non-overlapping named entity recognition.

Xu et al. (2017) propose a local detection method which relies on a Fixed-size Ordinally Forgetting Encoding (FOFE) method to encode utterance and a simple feed-forward neural network to either reject or predict the entity label for each

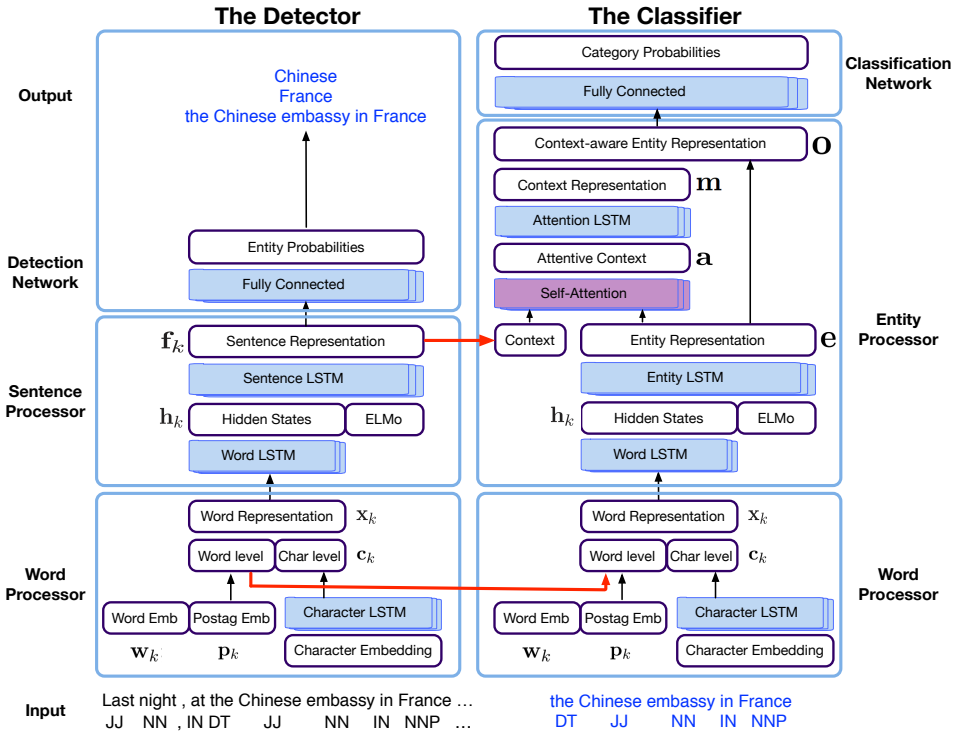

Figure 2: The framework of MGNER for Multi-Grained Named Entity Recognition. It consists of a Detector and a Classifier.

individual text fragment (Luan et al., 2018; Lee et al., 2017; He et al., 2018). Their model is in the same track with the framework we proposed whereas the difference is that we separate the NER task into two stages, *i.e.*, detecting entity positions and classifying entity categories.

3 The Proposed Framework

An overview of the proposed MGNER framework for multi-grained entity recognition, is illustrated in Figure 2. Specifically, MGNER consists of two sub-networks: the Detector and the Classifier. The Detector detects all the possible entity positions while the Classifier aims at classifying detected entities into pre-defined entity categories. The Detector has three modules: 1) Word Processor which extracts word-level semantic features, 2) Sentence Processor that learns context information for each utterance and 3) Detection Network that decides whether a word segment is an entity or not. The Classifier consists of 1) Word Processor which has the same structure as the one in the Detector, 2) Entity Processor that obtains entity features and 3) Classification Network that classifies entity into pre-defined categories. In addition, a self-attention mechanism is adopted in the Entity Processor to help the model capture and utilize entity-related contextual information.

Each module in MGNER can be replaced with a wide range of different neural network designs. For example, BERT (Devlin et al., 2018) can be used as the Word Processor and a capsule model (Sabour et al., 2017; Xia et al., 2018) can be integrated into the Classification Network.

It is worth mentioning that in order to improve the learning speed as well as the performance of MGNER, the Detector and the Classifier are trained with a series of shared input features, including the pre-trained word embeddings and the pre-trained language model features. Sentence-level semantic features trained in the Detector are also transferred into the Classifier to introduce and utilize the contextual information. We present the key building blocks and the properties of the Detector in Section 3.1 and the Classifier in Section 3.2, respectively.

3.1 The Detector

The Detector is aimed at detecting possible entity positions within each utterance. It takes an utterance as the input and outputs a set of entity candidates. Essentially, we use a semi-supervised neural network inspired by (Peters et al., 2017)

to model this process. The architecture of the Detector is illustrated in the left part of Figure 2. Three major modules are contained in the Detector: Word Processor, Sentence Processor and Detection Network. More specifically, pre-trained word embeddings, POS tag information and character-level word information are used for generating semantically meaningful word representations. Word representations obtained from the Word Processor and the language model embeddings—ELMo (Peters et al., 2018), are concatenated together to produce context-aware sentence representations. Each possible word segment is then examined in the Detection Network and to be decided whether accepted it as an entity or not.

3.1.1 Word Processor

Word Processor extracts semantically meaningful word representation for each token. Given an input utterance with K tokens $(t_1, ..., t_K)$, each token $t_k (1 \leq k \leq K)$ is represented as

$$\mathbf{x}_k = [\mathbf{w}_k; \mathbf{p}_k; \mathbf{c}_k],$$

by using a concatenation of a pre-trained word embedding $\mathbf{w}_k$, POS tag embedding $\mathbf{p}_k$ if it exists, and a character-level word information $\mathbf{c}_k$. The pre-trained word embedding $\mathbf{w}_k$ with a dimension D_w is obtained from GloVe (Pennington et al., 2014). The character-level word information $\mathbf{c}_k$ is obtained with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) layer to capture the morphological information. The hidden size of this character LSTM is set as D_{cl}. As shown in the bottom of Figure 2, character embeddings are fed into the character LSTM. The final hidden states from the forward and backward character LSTM are concatenated as the character-level word information $\mathbf{c}_k$. Those POS tagging embeddings and character embeddings are randomly initialized and learned within the learning process.

3.1.2 Sentence Processor

To learn the contextual information from each sentence, another bidirectional LSTM, named word LSTM, is applied to sequentially encode the utterance. For each token, the forward hidden states $\overrightarrow{\mathbf{h}}_k$ and the backward hidden states $\overleftarrow{\mathbf{h}}_k$ are concatenated into the hidden states $\mathbf{h}_k$. The dimension of the hidden states of the word LSTM is set as D_{wl}.

$$\begin{aligned} \overrightarrow{\mathbf{h}}_k &= \text{LSTM}_{fw}(\mathbf{x}_k, \overrightarrow{\mathbf{h}}_{k-1}), \\ \overleftarrow{\mathbf{h}}_k &= \text{LSTM}_{bw}(\mathbf{x}_k, \overleftarrow{\mathbf{h}}_{k+1}), \\ \mathbf{h}_k &= [\overrightarrow{\mathbf{h}}_k; \overleftarrow{\mathbf{h}}_k]. \end{aligned} \tag{1}$$

Besides, we also utilize the language model embeddings pre-trained in an unsupervised way as the ELMo model in (Peters et al., 2018). The pre-trained ELMo embeddings and the hidden states in the word LSTM $\mathbf{h}_k$ are concatenated. Hence, the concatenated hidden states $\mathbf{h}_k$ for each token can be reformulated as:

$$\mathbf{h}_k = [\overrightarrow{\mathbf{h}}_k; \overleftarrow{\mathbf{h}}_k; \mathbf{ELMo}_k], \tag{2}$$

where $\mathbf{ELMo}_k$ is the ELMo embeddings for token t_k. Speficially, a three-layer bi-LSTM neural network is trained as the language model. Since the lower-level LSTM hidden states have the ability to model syntax properties and higher-level LSTM hidden states can capture contextual information, ELMo computes the language model embeddings as a weighted combination of all the bidirectional LSTM hidden states:

$$\mathbf{ELMo}_k = \gamma \sum_{l=0}^{L} u_j \mathbf{h}_{k,l}^{LM}, \tag{3}$$

where γ is a task-specified scale parameter which indicates the importance of the entire ELMo vector to the NER task. L is the number of layers used in the pre-trained language model, the vector $\mathbf{u} = [u_0, \cdots, u_L]$ represents softmax-normalized weights that combine different layers. $\mathbf{h}_{k,l}^{LM}$ is the language model hidden state of layer l at the time step k.

A sentence bidirectional LSTM layer with a hidden dimension of D_{sl} is employed on top of the concatenated hidden states $\mathbf{h}_k$. The forward and backward hidden states in this sentence LSTM are concatenated for each token as the final sentence representation $\mathbf{f}_k \in \mathbb{R}^{2D_{sl}}$.

3.1.3 Detection Network

Using the semantically meaningful features obtained in $\mathbf{f}_k$, we can identify possible entities within each utterance. The strategy of finding entities is to first generate all the word segments as entity proposals and then estimate the probability of each proposal as being an entity or not.

To enumerate all possible entity proposals, different lengths of entity proposals are generated

surrounding each token position. For each token position, R entity proposals with the length varies from 1 to the maximum length R are generated. Specifically, it is assumed that an input utterance consists of a sequence of N tokens $(t_1, t_2, t_3, t_4, t_5, t_6, ..., t_N)$. To balance the performance and the computational cost, we set R as 6. We take each token position as the center and generate 6 proposals surrounding it. All the possible $6N$ proposals under the max-length of 6 will be generated. As shown in Figure 3, the entity proposals generated surrounding token t_3 are: (t_3), (t_3, t_4), (t_2, t_3, t_4), (t_2, t_3, t_4, t_5), $(t_1, t_2, t_3, t_4, t_5)$, $(t_1, t_2, t_3, t_4, t_5, t_6)$. Similar entity proposals are generated for all the token positions and proposals that contain invalid indexes like (t_0,t_1,t_2) will be deleted. Hence we can obtain all the valid entity proposals under the condition that the max length is R.

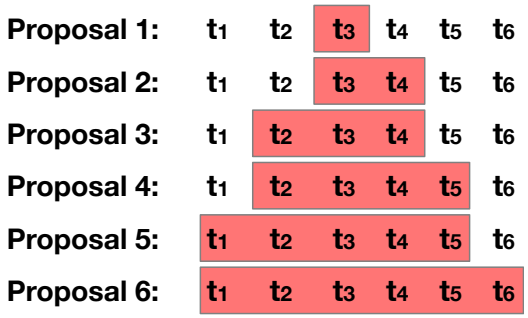

Figure 3: All possible entity proposals generated surrounding token t_3 when the maximum length of an entity proposal R is set as 6.

For each token, we simultaneously estimate the probability of a proposal being an entity or not for R proposals. A fully connected layer with a two-class softmax function is used to determine the quality of entity proposals:

$$\mathbf{s}_k = \text{softmax}\left(\mathbf{f}_k \mathbf{W}_p + \mathbf{b}_p\right), \quad (4)$$

where $\mathbf{W}_p \in \mathbb{R}^{2D_{sl} \times 2R}$ and $\mathbf{b}_p \in \mathbb{R}^{2R}$ are weights and the bias for the entity proposal layer; $\mathbf{s}_k$ contains $2R$ scores including R scores for being an entity and R scores for not being an entity at position k. The cross-entropy loss is employed in the Detector as follows:

$$L_p = -\sum_{k=1}^{K} \sum_{r=1}^{R} \mathbf{y}_k^r \log \mathbf{s}_k^r, \quad (5)$$

where $\mathbf{y}_k^r$ is the label for proposal type r at position k and $\mathbf{s}_k^r$ is the probability of being an entity for proposal type r at position k. It is worth mentioning that, most entity proposals are negative proposals. Thus, to balance the influence of positive proposals and negative proposals in the loss function, we keep all positive proposals and use down-sampling for negative proposals when calculating the loss L_p. For each batch, we fix the number of the total proposals, including all positive proposals and sampled negative proposals, used in the loss function as N_b. In the inference procedure of the Detection Network, an entity proposal will be recognized as an entity candidate if its score of being an entity is higher than score of not being an entity.

3.2 The Classifier

The Classifier module aims at classifying entity candidates obtained from the Detector into different pre-defined entity categories. For the nested NER task, all the proposed entities will be saved and fed into the Classifier. For the NER task which has non-overlapping entities, we utilize the non-maximum suppression (NMS) algorithm (Neubeck and Van Gool, 2006) to deal with redundant, overlapping entity proposals and output real entity candidates. The idea of NMS is simple but effective: picking the entity proposal with the maximum probability, deleting conflict entity proposals, and repeating the previous process until all the proposals are processed. Eventually, we can get those non-conflict entity candidates as the input of the Classifier.

To understand the contextual information of the proposed entity, we utilize both sentence-level context information and a self-attention mechanism to help the model focus on entity-related context tokens. The framework of the Classifier is shown in the right part of Figure 2. Essentially, it consists of three modules: Word Processor, Entity Processor and Classification Network.

3.2.1 Word Processor

A same Word Processor as in the Detector is used here to get the word representation for the entity candidates obtained from the Detector. The word-level embedding, which is the concatenation of pre-trained word embedding and POS tag embedding if it is exists, is transferred from the Word Processor in the Detector to improve the performance as well as to speed up the learning process. The character-level LSTM and character embeddings are trained separately in the Detector and the Classifier.

		ACE-2004			ACE-2005			CoNLL-2003		
		TRAIN	DEV	TEST	TRAIN	DEV	TEST	TRAIN	DEV	TEST
sentences	#total	6,799	829	879	7,336	958	1,047	14,987	3,466	3,684
	#overlaps	2,683(39%)	293(35%)	373(42%)	2,683 (37%)	340(35%)	330 (32%)	-	-	-
entities	#total	22,207	2,511	3,031	24,687	3,217	3,027	23,499	5,942	5,648
	#overlaps	10,170 (46%)	1,091(43%)	1,418 (47%)	9,937 (40%)	1,192(37%)	1,184 (39%)	-	-	-
	length >6	1,439 (6%)	179(7%)	199 (7%)	1,343 (5%)	148(5%)	160 (6%)	23(0.1%)	8(0.1%)	0 (0%)
	max length	57	35	43	49	30	27	10	10	6

Table 1: Corpora Statistics for the ACE-2004, ACE-2005 and CoNLL-2003 datasets.

3.2.2 Entity Processor

The word representation is fed into a bidirectional word LSTM with hidden size D_{wl} and the hidden states are concatenated with the ELMo language model embeddings as the entity features. A bidirectional LSTM with hidden size D_{el} is applied to the entity feature to capture sequence information among the entity words. The last hidden states of the forward and backward Entity LSTM are concatenated as the entity representation $\mathbf{e} \in \mathbb{R}^{2D_{el}}$.

The same word in different contexts may have different semantic meanings. To this end, in our model, we take the contextual information into consideration when learning the semantic representations of entity candidates. We capture the contextual information from other words in the same utterance. Denote $\mathbf{c}$ as the context feature vector for these context words, and it can be extracted from the sentence representation $\mathbf{f}_k$ in the Detector. Hence, the sentence features trained in the Detector is directly transferred to the Classifier.

An easy way to model context words is to concatenate all the word representations or average them. However, this naive approach may fail when there exists a lot of unrelated context words. To select high-relevant context words and learn an accurate contextual representation, we propose a self-attention mechanism to simulate and dynamically control the relatedness between the context and the entity. The self-attention module takes the entity representation $\mathbf{e}$ and all the context features $\mathbf{C} = [\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_N}]$ as the inputs, and outputs a vector of attention weights $\mathbf{a}$:

$$\mathbf{a} = softmax(\mathbf{C}\mathbf{W}\mathbf{e}^T), \quad (6)$$

where $\mathbf{W} \in \mathbb{R}^{2D_{sl} \times 2D_{el}}$ is a weight matrix for the self-attention layer, and $\mathbf{a}$ is the self-attention weight on different context words. To help the model focus on entity-related context, the attentive vector $\mathbf{C}^{att}$ is calculated as the attention-weighted context:

$$\mathbf{C}^{att} = \mathbf{a} * \mathbf{C}. \quad (7)$$

The lengths of the attentive context $\mathbf{C}^{att}$ varies in different contexts. However, the goal of the Classification Network is to classify entity candidates into different categories, and thus it requires a fixed embedding size. We achieve that by adding another LSTM layer. An Attention LSTM with the hidden dimension D_{ml} is used and the concatenation of the last hidden states in the forward and backward LSTM layer as the context representation $\mathbf{m} \in \mathbb{R}^{2D_{ml}}$. Hence the shape of the context representation is aligned. We concatenate the context representation and the entity representation together as a context-aware entity representation to classify entity candidates: $\mathbf{o} = [\mathbf{m}; \mathbf{e}]$.

3.2.3 Classification Network

A two-layer fully connected neural network is used to classify candidates into pre-defined categories:

$$\mathbf{p} = \text{softmax}\left(\mathbf{W}_{c2}\left(\sigma\left(\mathbf{o}\mathbf{W}_{c1} + \mathbf{b}_{c1}\right)\right) + \mathbf{b}_{c2}\right), \quad (8)$$

where $\mathbf{W_{c1}} \in \mathbb{R}^{(2D_{ml}+2D_{el}) \times D_h}$, $\mathbf{b_{c1}} \in \mathbb{R}^{D_h}$, $\mathbf{W_{c2}} \in \mathbb{R}^{D_{c1} \times (D_t+1)}$, $\mathbf{b_{c2}} \in \mathbb{R}^{D_t+1}$ are the weights for this fully connected neural network, and D_t is the number of entity types. Actually, this classification function classifies entity candidates into $(D_t + 1)$ types. Here we add one more type as for the scenario that a candidate may not be a real entity. Finally, the hinge-ranking loss is adopted in the Classification Network:

$$L_c = \sum\nolimits_{y_w \in Y_w} max\left\{0, \Delta + \mathbf{p}_{y_w} - \mathbf{p}_{y_r}\right\}, \quad (9)$$

where $\mathbf{p}_w$ is the probability for the wrong labels y_w, $\mathbf{p}_r$ is the probability for the right label y_r, and Δ is a margin. The hinge-rank loss urges the probability for the right label higher than the probability for the wrong labels and improves the classification performance.

4 Experiments

To show the ability and effectiveness of our proposed framework, MGNER, for Multi-Grained

Named Entity Recognition, we conduct the experiments on both Nested NER task and traditional non-overlapping NER task.

4.1 Datasets

We mainly evaluate our framework on ACE-2004 and ACE-2005 (Doddington et al., 2004) with the same splits used by previous works (Luo et al., 2015; Wang and Lu, 2018) for the nested NER task. Specifically, seven different types of entities such as person, facility, weapon and vehicle, are contained in the ACE datasets. For the traditional NER task, we use the CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003) which contains four types of named entities: location, organization, person and miscellaneous. An overview of these three datasets is illustrated in Table 1. It can be observed that most entities are less or equal to 6 tokens, and thus we select the maximum entity length $R = 6$.

4.2 Implementation Details

We performed random search (Bergstra and Bengio, 2012) for hyper-parameter optimization and selected the best setting based on performance on the development set. We employ the Adam optimizer (Kingma and Ba, 2014) with learning rate decay for all the experiments. The learning rate is set as 0.001 at the beginning and exponential decayed by 0.9 after each epoch. The batch size of utterances is set as 20. In order to balance the influence of positive proposals and negative proposals, we use down-sampling for negative ones and the total proposal number N_b for each batch is 128. To alleviate over-fitting, we add dropout regularizations after the word representation layer and all the LSTM layers with a dropout rate of 0.5. In addition, we employ the early stopping strategy when there is no performance improvement on the development dataset after three epochs. The pre-trained word embeddings are from GloVe (Pennington et al., 2014), and the word embedding dimension D_w is 300. Besides, the ELMo 5.5B data[1] is utilized in the experiment for the language model embedding. Moreover, the size of character embedding $\mathbf{c}_k$ is 100, and the hidden size of the Character LSTM D_{cl} is also 100. The size of POS tag embedding $\mathbf{p}_k$ is 300 for the ACE datasets and no POS tag information is used in the CoNLL-2003 dataset. The hidden dimensions of the Word LSTM layer D_{wl}, the Sentence LSTM layer D_{sl}, the Entity LSTM layer D_{el} and the Attention LSTM layer D_{ml} are all set to 300. The hidden dimension of the classification layer D_h is 50. The margin Δ in the hinge-ranking loss for the entity category classification is set to 5. The ELMo scale parameter γ used in the Detector is 3.35 and 3.05 in the Classifier, respectively.

[1] https://allennlp.org/elmo

MODEL	ACE-2004			ACE-2005		
	P	R	F1	P	R	F1
Lu and Roth (2015)	70.0	56.9	62.8	66.3	59.2	62.5
Lample et al. (2016)	71.3	50.5	58.3	64.1	52.4	57.6
Muis and Lu (2017)	72.7	58.0	64.5	69.1	58.1	63.1
Xu et al. (2017)	68.2	54.3	60.5	67.4	55.1	60.6
Katiyar and Cardie (2018)	73.6	71.8	72.7	70.6	70.4	70.5
Ju et al. (2018)	-	-	-	74.2	70.3	72.2
Wang et al. (2018)	74.9	71.8	73.3	74.5	71.5	73.0
Wang and Lu (2018)	78.0	72.4	75.1	76.8	72.3	74.5
MGNER w/o context	79.8	76.3	78.0	**79.6**	75.6	77.5
MGNER w/o attention	81.5	76.5	78.9	79.4	76.0	77.7
MGNER	**81.7**	**77.4**	**79.5**	79.0	**77.3**	**78.2**

Table 2: Performance on ACE-2004 and ACE-2005 test set for the Nested NER task.

4.3 Results

Nested NER Task. The proposed MGNER is very suitable for detecting nested named entities since every possible entity will be examined and classified. In order to validate this advantage, we compare MGNER with numerous baseline models: 1) Lu and Roth (2015) which propose the mention hypergraphs for recognizing overlapping entities; 2) Lample et al. (2016) which adopt the LSTM-CRF stucture for sequence labelling; 3) Muis and Lu (2017) which introduce mention separators to tag gaps between words for recognizing overlapping mentions; 4) Xu et al. (2017) that propose a local detection method; 5) Katiyar and Cardie (2018) which propose a hypergraph-based model using LSTM for learning feature representations; 6) Ju et al. (2018) that use a layered model which extracts outer entities based on inner ones; 7) Wang et al. (2018) which propose a neural transition-based model that constructs nested mentions through a sequence of actions; 8) Wang and Lu (2018) which adopt a neural segmental hypergraph model.

Experiment results of the Nested NER task on the ACE-2004 and ACE-2005 datasets are reported in Table 2. We can observe from Table 2 that, our proposed framework MGNER outperforms all the baseline approaches. For both datasets, our model improves the state-of-the-art

result by around 4% in terms of precision, recall, as well as the F1 score.

To study the contribution of different modules in MGNER, we also report the performance of two ablation variations of the proposed MGNER at the bottom of Table 2. MGNER w/o attention is a variation of MGNER which removes the self-attention mechanism and MGNER w/o context removes all the context information. To remove the self-attention mechanism, we feed the context feature $\mathbf{C}$ directly into a bi-directional LSTM to obtain context representation $\mathbf{m}$, other than the attentive context vector $\mathbf{C}^{att}$. As for MGNER w/o context, we only use entity representation $\mathbf{e}$ to do classification other than the context-aware entity representation $\mathbf{o}$. By adding the context information, the F1 score improves 0.9% on the ACE-2004 dataset and 0.7% on the ACE-2005 dataset. The self-attention mechanism improves the F1 score by 0.6% on the ACE-2004 dataset and 0.5% on the ACE-2005 dataset.

MODEL	OVERLAPPING			NON-OVERLAPPING		
	P	R	F1	P	R	F1
Lu and Roth (2015)	68.1	52.6	59.4	64.1	65.1	64.6
Muis and Lu (2017)	70.4	55.0	61.8	67.2	63.4	65.2
Wang et al. (2018)	77.4	70.5	73.8	76.1	69.6	72.7
Wang and Lu (2018)	80.6	73.6	76.9	75.5	71.5	73.4
MGNER	**82.6**	**76.0**	**79.2**	**77.8**	**79.5**	**78.6**

Table 3: Results on different types of sentences (ACE-2005).

To analyze how well our model performs on overlapping and non-overlapping entities, we split the test data into two portions: sentences with and without overlapping entities (follow the splits used by Wang and Lu (2018)). Four state-of-the-art nested NER models are compared with our proposed framework MGNER on the ACE-2005 dataset. As illustrated in Table 3, MGNER consistently performs better than the baselines on both portions, especially for the non-overlapping part. This observation indicates that our model can better recognize non-overlapping entities than previous nested NER models.

The first step in MGNER is to detect entity positions using the Detector, where the effectiveness of proposing correct entity candidates immediately affects the performance of the whole model. To this end, we provide the experiment results of detecting correct entities in the Detector module here. The precision, recall and F1 score are 85.23 , 91.84, 88.41 for the ACE-2004 dataset and 84.95, 89.35, 87.09 for the ACE-2005 dataset.

MODEL	CoNLL-2003	
	DEV	TEST
Lu and Roth (2015)	89.2	83.8
Muis and Lu (2017)	-	84.3
Xu et al. (2017)	-	90.85
Wang and Lu (2018)	-	90.2
Lample et al. (2016)	-	90.94
Ma and Hovy (2016)	94.74	91.21
Chiu and Nichols (2016)	94.03 ± 0.23	91.62 ± 0.33
Peters et al. (2017)	-	91.93 ± 0.19
Peters et al. (2018)	-	92.22 ± 0.10
MGNER w/o context	95.21 ± 0.12	92.23 ± 0.06
MGNER w/o attention	95.23 ± 0.06	92.26 ± 0.09
MGNER	$\mathbf{95.24 \pm 0.13}$	$\mathbf{92.28 \pm 0.12}$

Table 4: F1 scores on CoNLL-2003 devlopement set (DEV) and test set (TEST) for the English NER task. Mean and standard deviation across five runs are reported. Pos tags information are not used.

NER Task. We also evaluate the proposed MGNER framework on the NER task which needs to reorganize non-overlapping entities. Two types of baseline models are compared here: sequence labelling models which are designed specifically for non-overlapping NER task and nested NER models which also provide the ability to detect non-overlapping mentions. The first type of models including 1) Lample et al. (2016) which adopt the LSTM-CRF structure; 2) Ma and Hovy (2016) which use a LSTM-CNNs-CRF architecture; 3) Chiu and Nichols (2016) which propose a CNN-LSTM-CRF model; 4) Peters et al. (2017) which add semi-supervised language model embeddings; and 5) Peters et al. (2018) which utilize the state-of-the-art ELMo language model embeddings. The second types include four Nested models mentioned in the Nested NER section: 1) Luo et al. (2015); 2) Muis and Lu (2017); 3) Xu et al. (2017); 4) (Wang and Lu, 2018).

Table 4 shows the F1 scores of different approaches on CoNLL-2003 devlopement set and test set for the English NER task. Mean and standard deviation across five runs are reported. It can be observed from Table 4 that the proposed MGNER model outperforms all the baselines. The models designed for non-overlapping entity detection usually performs better than Nested NER models for the NER task. Our proposed framework outperforms state-of-the-art results both on the NER and Nested NER task. Xu et al. (2017) is the best baseline model among the Nested models since it shares a similar idea of our proposed framework by individually examin-

ing each entity proposal. From the ablation study, we can observe that by purely adding the context information, the F1 score on the CoNLL-2003 test set improves from 92.23 to 92.26, and by adding the attention mechanism, the F1 score improves to 92.28.

We also provide the performance of detecting non-overlapping entities in the Detector here. The precision, recall and F1 score are 95.33, 95.69 and 95.51 on the CoNLL-2003 dataset.

5 Conclusions

In this work, we propose a novel neural framework named MGNER for Multi-Grained Named Entity Recognition where multiple entities or entity mentions in a sentence could be non-overlapping or totally nested. MGNER is framework with high modularity and each component in MGNER can adopt a wide range of neural networks. Experimental results show that MGNER is able to achieve state-of-the-art results on both nested NER task and traditional non-overlapping NER task.

Acknowledgments

We thank the reviewers for their valuable comments. Special thanks go to Lu Wei from Singapore University of Technology and Design for sharing the datasets split details. This work is supported in part by NSF through grants IIS-1526499, IIS-1763325, and CNS-1626432.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 65–72. Association for Computational Linguistics.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 141–150. Association for Computational Linguistics.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1446–1459.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 861–871.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom M Mitchell. 2015. Learning a compositional semantics for freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3:257–270.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

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 (Volume 1: Long Papers)*, volume 1, pages 1064–1074.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Aldrian Obaja Muis and Wei Lu. 2017. Labeling gaps between words: Recognizing overlapping mentions with mention separators. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2608–2618. Association for Computational Linguistics.

Alexander Neubeck and Luc Van Gool. 2006. Efficient non-maximum suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 850–855. IEEE.

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

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765.

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)*, pages 2227–2237.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866.

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*, pages 142–147. Association for Computational Linguistics.

Bailin Wang and Wei Lu. 2018. Neural segmental hypergraphs for overlapping mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 204–214.

Bailin Wang, Wei Lu, Yu Wang, and Hongxia Jin. 2018. A neural transition-based model for nested mention recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1011–1017.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3090–3099.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1237–1247.