Pyramid: A Layered Model for Nested Named Entity Recognition

Jue Wang [†], **Lidan Shou** [‡†*], **Ke Chen** [†], **Gang Chen** [‡†]
[‡]State Key Laboratory of CAD&CG,
[†]College of Computer Science and Technology,
Zhejiang University, China
{zjuwangjue,should,chenk,cg}@zju.edu.cn

Abstract

This paper presents `Pyramid`, a novel layered model for Nested Named Entity Recognition (nested NER). In our approach, token or text region embeddings are recursively inputted into L flat NER layers, from bottom to top, stacked in a pyramid shape. Each time an embedding passes through a layer of the pyramid, its length is reduced by one. Its hidden state at layer l represents an l-gram in the input text, which is labeled only if its corresponding text region represents a complete entity mention. We also design an inverse pyramid to allow bidirectional interaction between layers. The proposed method achieves state-of-the-art F1 scores in nested NER on ACE-2004, ACE-2005, GENIA, and NNE, which are 80.27, 79.42, 77.78, and 93.70 with conventional embeddings, and 87.74, 86.34, 79.31, and 94.68 with pre-trained contextualized embeddings. In addition, our model can be used for the more general task of Overlapping Named Entity Recognition. A preliminary experiment confirms the effectiveness of our method in overlapping NER.

1 Introduction

Named Entity Recognition (NER), which aims at identifying text spans as well as their semantic classes, is an essential and fundamental Natural Language Processing (NLP) task. It is typically modeled as a sequence labeling problem, which can be effectively solved by RNN-based approach (Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016). However, such formulation oversimplifies the problem and is based on a very strong assumption that entity mentions do not overlap with each other, which is certainly not the real case. In real-world languages, entities might be deeply nested or overlapping, calling for better models to handle such complexity.

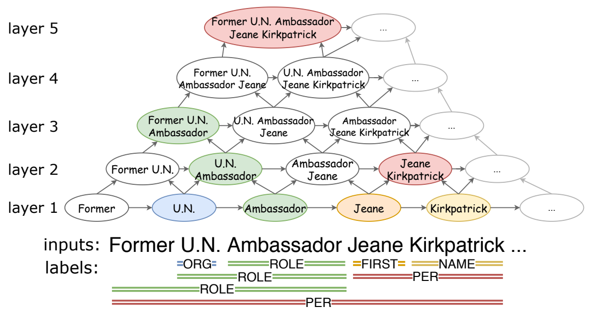

Figure 1: Pyramid output of a sentence from NNE (Ringland et al., 2019) containing 8 nested entities.

Many previous studies have focused on recognizing nested entity mentions. A few works use proprietary structures, such as constituency graph (Finkel and Manning, 2009) or hypergraph (Lu and Roth, 2015; Muis and Lu, 2017), to explicitly capture nested entities. These structures, however, do not produce satisfactory performance results.

Some other works handle nested entity mentions in a *layered* model, which employs multiple flat NER layers(Alex et al., 2007; Ju et al., 2018; Fisher and Vlachos, 2019). Each layer is usually responsible for predicting a group of nested entities having the same nesting level.

Unfortunately, conventional layered schemes do not address the more general *overlapping* setting, and also suffer from *layer disorientation*. The latter is a problem arising when the model might output a nested entity from a wrong layer. For example, entity "U.N. Ambassador" is labeled as a second-layer entity (containing "U.N." and "Ambassador"). Thus, prediction of it from the first layer is considered an error. Generally, a false positive prediction with the correct span and class but from a wrong layer produces an over-estimated loss (despite the correct entity itself), causing the entire model reluctant to predict positive, and eventually harming the recall. This problem occurs quite often, as the

*Corresponding author

target layer for a nested entity is determined by *the nesting levels of its composing entities* rather than by its own semantics or structure. A recent study on a layered model (Ju et al., 2018) also reports the *error propagation* issue, i.e. errors in the first few layers are propagated to the next layers.

In this paper, we propose a novel layered model called `Pyramid` for nested NER. The model consists of a stack of inter-connected layers. Each layer l predicts whether a text region of certain length l, i.e. an l-gram, is a complete entity mention. Between each two consecutive layers of our model, the hidden state sequence is fed into a convolutional network with a kernel of two, allowing a text region embedding in the higher layer to aggregate two adjacent hidden states from the lower layer, and thus forming the pyramid look (as the length of the sequence in the higher layer is one token shorter than the lower layer). Such process enumerates all text spans without breaking the sequence structure.

Figure 1 shows a sentence containing eight nested entities being fed into the `Pyramid` model. These entities are separated into 5 layers according to their number of tokens. The job of each decoding layer is simple and clear – it needs to output entity type when it encounters a complete entity.

In the above scheme, the higher decoding layer relies on the output of the lower decoding layer in a bottom-up manner (from layer 1 to 5 in Figure 1). It is also desirable to construct an *inverse pyramid*, where a lower decoding layer receives input from a higher layer (from layer 5 to 1), allowing information to flow in the opposite way.

`Pyramid` outperforms the previous methods in nested NER while addressing all the aforementioned problems with layered model. First, it can be used for more general overlapping NER. Second, it prevents layer disorientation as an l-length entity in the input is only predicted on layer l. Third, it mitigates the error propagation problem, as predictions in one layer do not dictate those in other layers. Our main contributions are as follows:

- We propose a novel layered model called `Pyramid` for nested NER. The model recognizes entity mentions by its length without layer disorientation and error propagation. The proposed model can also address the more general overlapping NER task.

- Besides the normal pyramid, we design an inverse pyramid to allow bidirectional interactions between neighboring layers.

- We evaluate the proposed method on four datasets, namely ACE-2004 (Doddington et al., 2004), ACE-2005 (Walker et al., 2006), GENIA (Kim et al., 2003) and NNE (Ringland et al., 2019). The results suggest that our model significantly outperforms the previous methods, and achieves state-of-the-art performance with and without pre-trained language model embeddings (ALBERT (Lan et al., 2019), BERT (Devlin et al., 2019), and Flair (Akbik et al., 2018)).

- Additionally, we construct a small dataset that contains overlapping but non-nested entities. Preliminary results on this dataset show the potential of our model for handling overlapping entities.

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 regular NER, including probabilistic graph models such as Conditional Random Fields (CRF) (Ratinov and Roth, 2009), and deep neural networks like recurrent neural networks (RNN) and convolutional neural networks (CNN). Recently, LSTM-CRF has become a standard architecture for sequence labeling tasks. Huang et al. 2015 uses hand-crafted spelling features; Ma and Hovy 2016 uses CNN to capture character features; Lample et al. 2016 utilizes LSTM instead. These sequence labeling models can only detect non-overlapping entities and fail to handle nested ones.

Nested NER has been intensively studied recently. Finkel and Manning 2009 proposes a CRF-based constituency parser and use a constituency tree to represent a sentence. Lu and Roth 2015 introduces the idea of hypergraph which allows edges to connect to multiple nodes to represent nested entities. Muis and Lu 2017 uses a multigraph representation and introduces the notion of mention separator for nested entity detection. Wang and Lu 2018 presents a neural segmental hypergraph model using neural networks to obtain distributed feature representation. Katiyar and Cardie 2018 also adopts a hypergraph-based formulation but instead uses neural networks to learn the structure. Lin et al. 2019 borrows the Anchor Region Networks (ARNs) architecture to predict nested entity mentions. All the above works design proprietary structures to explicitly capture nested entities.

Layered models are common solution for nested NER. Alex et al. 2007 stacks multiple flat NER layers, where the first recognizes the innermost (or outermost) mentions, then the following taggers are used to incrementally recognize next-level mentions. Ju et al. 2018 dynamically stacks multiple flat NER layers and extract outer entities based on the inner ones. Fisher and Vlachos 2019 can also be considered as a layered model with a novel neural network architecture. Our method differs from the above layered models in that (1) it is able to handle overlapping NER, and (2) it does not suffer the layer disorientation or error propagation problem.

Exhaustive region classification model enumerates all possible regions of the input sentence. Byrne 2007; Xu et al. 2017; Sohrab and Miwa 2018; Zheng et al. 2019 aggregate all possible adjacent tokens into potential spans. These spans, together with their left and right contexts, are fed into a classifier - a maximum entropy tagger (Byrne, 2007) or a neural network (Xu et al., 2017; Sohrab and Miwa, 2018; Zheng et al., 2019). Unfortunately, all these works fail to take advantage of the dependencies among nested entities, but perform prediction merely on individual text fragments, thus limiting the performance. Luan et al. 2019 uses propagation layers to capture relation and coreference between spans. Our method also potentially enumerates all possible spans, while maintaining the sequence structure, which leads to better performance.

Pre-trained word embeddings, e.g. Glove (Pennington et al., 2014), have proved to be effective in improving NER performance. Recently, with the rapid development of language model techniques, the performance of NER models has been pushed to a new height. The recent pre-trained language model embeddings include ELMo (Peters et al., 2018), Flair (Akbik et al., 2018), BERT (Devlin et al., 2019), ALBERT (Lan et al., 2019), etc. In our experiments, we leverage these embeddings and observe significant performance improvements.

3 Proposed Method

In this section, we describe the proposed model and its architecture, which includes an encoder, a pyramid, an inverse pyramid, and a logits layer. Figure 2 shows a toy model with a pyramid (5 bottom-up decoding layers in blue) and its inverse counterpart (5 top-down layers in pink). As shown in the blue pyramid, each decoding layer contains a convolutional network with a kernel of two to reduce the sequence length in its output, so that all possible mention spans can potentially be enumerated. The top-down inverse pyramid will be described later.

We shall use the following notations:

$Embed$	the embedding layer
$LSTM$	the bidirectional LSTM layer
LM	the language model embedder
$Linear$	the fully-connected layer
$LayerNorm$	layer normalization

The mentioned layers with the same notation, superscript and subscript share the same parameters. For the sake of brevity, we omit the dropout layer in this section.

3.1 The Input and Output

The input is a T-length textual sentence. After the encoder, embedding sequences are recursively fed into flat NER decoding layers, producing L tag sequences in the IOB2-format[1] with length T, $T-1$, ..., $T-L+1$, where L is the number of decoding layers. Note we only label n-grams that are complete mentions, so `I-{class}` usually does not appear.

Given the running example in Figure 1, input sentence "Former U.N. Ambassador Jeane Kirkpatrick ..." contains eight entity mentions, namely (U.N., ORG), (Ambassador, ROLE), (Jeane, FIRST), (Kirkpatrick, NAME), (U.N. Ambassador, ROLE), (Jeane Kirkpatrick, PER), (Former U.N. Ambassador, ROLE), and (Former U.N. Ambassador Jeane Kirkpatrick, PER).

The output from the pyramid would contain layered tag sequences ($l = 1, \ldots, 5$) as follows:

```
l=5: B-PER ...
l=4: O     O      ...
l=3: B-PER O      O      ...
l=2: O     B-ROLE O      B-PER   ...
l=1: O     B-ORG  B-ROLE B-FIRST B-NAME ...
```

Unfortunately, the above layered sequences cannot include any entities of more than 5 tokens. Generally, a stack of L layers cannot predict entities containing more than L tokens!

To address this issue, we propose a *remedy solution*: to predict all entities longer than L tokens on the topmost flat NER layer. Specifically, the bottom $L-1$ layers predict `B-{class}` tags for

[1]Label the first token of a mention as `B-{class}`; other tokens inside a mention as `I-{class}`; tokens outside any mention as `O`.

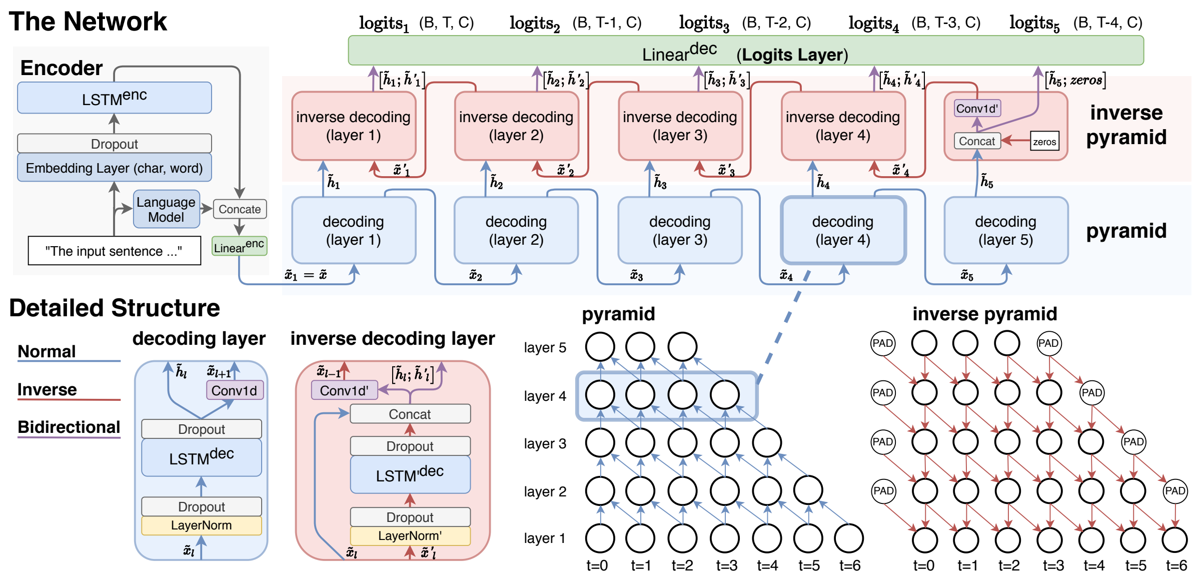

Figure 2: Overview of a toy network with 5 decoding layers. The upper half shows the overall structure, while the lower half shows the details. B is the batch size; T represents the length of original text; C is the class number.

complete entity mentions; and the topmost layer predicts both B-{class} and I-{class} tags. This stipulates that when two entities are nested, if one of them is longer than L, the other one cannot be longer than $L-1$.

In the running example, suppose we had only 4 decoding layers ($l=1,\dots,4$), then the longest mention (Former U.N. Ambassador Jeane Kirkpatrick) would be recognized in the fourth decoding layer as following:

```
l=4: B-PER I-PER   ...
l=3: B-PER O       O       ...
l=2: O     B-ROLE  O       B-PER   ...
l=1: O     B-ORG   B-ROLE  B-FIRST B-NAME ...
```

With the remedy solution, our model is able to handle entities longer than L. As most entity mentions are not too long (99% are no longer than 15 tokens), and it is even rarer for both two nested mentions to be longer than 15, we set the default number of flat decoder layers to $L=16$ to minimize the impact of the remedy. Parameter L can be tuned for balance between accuracy and inference speed.

3.2 The Encoder

We represent each word by concatenating character sequence embeddings and word embeddings. First, the character embeddings are dynamically generated by a LSTM (Lample et al., 2016) to capture the orthographic and morphological features of the word. It is suggested that with the introduction of character embeddings the model can better handle out-of-vocabulary (OOV) words. Second, the word embeddings are initialized with pre-trained word vectors. For OOV words, we randomly initialize an embedding for [UNK], which is tuned during training. The concatenated character and word embeddings are fed into a bidirectional LSTM encoding layer to further leverage contextual information.

Formally, given the input sentence x:

$$\tilde{x}^{char} = LSTM^{char}(Embed^{char}(x)) \quad (1)$$

$$\tilde{x}^{word} = Embed^{word}(x) \quad (2)$$

$$\tilde{x} = LSTM^{enc}([\tilde{x}^{char}; \tilde{x}^{word}]) \quad (3)$$

For better performance, we adopt the popular pre-trained contextualized language model embeddings, such as BERT (Devlin et al., 2019). These embeddings are concatenated to the output of $LSTM^{enc}$, followed by a linear layer to reduce the embedding dimension. i.e.:

$$\tilde{x} = Linear^{enc}([\tilde{x}; LM(x)]) \quad (4)$$

3.3 The Pyramid

The pyramid recognizes entities in a bottom-up manner. It consists of L decoding layers, each of which corresponds to a flat named-entity recognizer. Each decoding layer has two main components, a LSTM and a CNN with a kernel of two. In layer l, the LSTM recognizes l-length entity mentions, and the CNN aggregates two adjacent hidden states and then feeds the text region embeddings enriched with layer information to the higher ($l+1$-th) decoding layer. By passing through l decoding layers

		ACE-2004			ACE-2005			GENIA			NNE		
		train	dev	test	train	dev	test	train	dev	test	train	dev	test
sentences	# total	6,198	742	809	7,285	968	1,058	15,022	1,669	1,855	43,457	1,989	3,762
	# nested	2,718	294	388	2,797	352	339	3,222	328	448	28,606	1292	2489
		(44%)	(40%)	(48%)	(38%)	(36%)	(32%)	(21%)	(20%)	(24%)	(66%)	(65%)	(66%)
entities	# total	22,195	2,514	3,034	24,700	3,218	3,029	47,006	4,461	5,596	248,136	10,463	21,196
	# nested	10,157	1,092	1,417	9,946	1,191	1,179	8,382	818	1212	20,6618	8,487	17,670
		(46%)	(43%)	(47%)	(40%)	(37%)	(39%)	(18%)	(18%)	(22%)	(83%)	(81%)	(83%)
	max length	57	35	43	49	31	27	20	20	15	16	15	15

Table 1: Statistics of the datasets used in the experiments. A sentence is considered nested if any two mentions in it are nested. An entity mention is considered nested if it contains any mention or is contained by any mention.

with $l-1$ CNNs, each hidden state (at t) actually represents the region of l original tokens (from t to $t+l-1$). Therefore, the l-th decoding layer enumerates text spans of length l. And all these L layers together produce all possible entity spans.

One may notice that the pyramid structure intrinsically provides useful inductive bias: The higher the layer, the shorter the input sequence, forcing the model to capture high-level information for predicting long entities and low-level information for predicting short entities. Moreover, as the length of each span representation is reduced to one on its target decoding layer, the prediction task on each layer is simple and clear - to predict entities whose representation length is one in this layer.

Since the input of the first decoding layer is from the encoder while the others are from the output of their lower neighboring layers, the input bias and scale may differ among layers. This is detrimental to training. To address this issue, we apply *layer normalization* (Ba et al., 2016) before feeding the region embeddings into the decoding LSTM.

Let $\tilde{x}_1 = \tilde{x}$, for each decoding layer l:

$$h_l = \underline{LSTM^{dec}(LayerNorm(\tilde{x}_l))} \quad (5)$$

$$\tilde{x}_{l+1} \underline{= Conv1d(h_l)} \quad (6)$$

3.4 The Inverse Pyramid

Each decoding layer in the bottom-up pyramid takes into account layer information from lower layers. However, a layer cannot get feedback from its higher neighbors, which could potentially help. Moreover, for long entities, their embeddings need to go through numerous lower layers and tend to lose important information.

Therefore, we add an inverse pyramid, which recognizes entity mentions in a top-down manner, to address the above issues. While in the pyramid, sequences pass through a CNN to reduce sequence length before being fed into the higher decoding layer, in the inverse pyramid, however, we use another CNN with zero paddings and a kernel of two to reconstruct the lower-level text region embeddings. Specifically, to reconstruct the text region embeddings at the $l-1$-th decoding layer, we concatenate the hidden states of the l-th *normal* and *inverse* decoding layers, and feed it to the inverse CNN (see bottom-left pink box in Figure 2).

There are two benefits for using the top-down inverse pyramid: (1) It gives the feedback from higher decoding layers, allowing bidirectional interaction between neighboring decoding layers; (2) Since the inverse pyramid needs to reconstruct lower-level sequence, it requires the pyramid to retain as much original information as possible, thereby mitigating the information loss for long entities.

Formally we have the following output from the inverse decoding layers:

$$h'_l = LSTM'^{dec}(LayerNorm'(\tilde{x}'_l)) \quad (7)$$

$$\tilde{x}'_{l-1} = Conv1d'([h_l; h'_l]) \quad (8)$$

For the top inverse decoding layer, we cannot compute h'_L, so we use zeros instead.

Finally, with the concatenation of the hidden states of both the normal and inverse decoding layers, we use a feed-forward layer to predict their class:

$$logits_l = Linear^{dec}([h_l; h'_l]). \quad (9)$$

4 Experiment

4.1 Datasets

We evaluate our model on four nested entity recognition corpora: ACE-2004 (Doddington et al., 2004), ACE-2005 (Walker et al., 2006), GENIA (Kim et al., 2003), and NNE (Ringland et al., 2019). For ACE-2004 and ACE-2005, we adopt the train/dev/test split of Lu and Roth 2015[2], as

[2]https://statnlp-research.github.io/publications/

Setting	Value
batch size	32,32,64,32
optimizer	SGD
momentum	0.9
learning rate (lr)	0.01
dropout rate	0.3,0.4,0.4,0.2
hidden dim	200
# stacked layers	16
token emb dim	100,100,200,100
char emb dim	30,30,60,30
gradient clipping	5.0

Table 2: Hyperparameters used in our experiments. If 4 values are given, they correspond to ACE-2004, ACE-2005, GENIA and NNE respectively.

used in most previous studies. For GENIA, we use GENIAcorpus3.02p[3], and follow the train/dev/test split of previous works (Finkel and Manning, 2009; Lu and Roth, 2015) i.e.: (1) split first 81%, subsequent 9%, and last 10% as train, dev and test set, respectively; (2) collapse all DNA, RNA, and protein subtypes into DNA, RNA, and protein, keeping cell line and cell type, and (3) removing other entity types, resulting in 5 entity types. For NNE, we keep the original dataset split and pre-processing. The statistics of each dataset are shown in Table 1.

4.2 Training Details

We denote by `Pyramid-Basic` the model using the normal bottom-up pyramid only; and by `Pyramid-Full` the one with both the normal and inverse pyramids. We try to use as similar settings as possible on all datasets, and Table 2 describes the settings used in our experiments. For the word embeddings, we use 100-dimensional GloVe word embeddings trained on 6B tokens[4] as initialization. We disable updating the word embeddings during training. Besides, character-based embeddings are generated by a LSTM (Lample et al., 2016). We set the hidden dimension to 200 (100 for each direction in bidirectional LSTM). We use inverse time learning rate decay: $\hat{lr} = lr/(1 + \text{decay\_rate} * \text{steps}/\text{decay\_steps})$, with decay rate 0.05 and decay steps 1000. All results are averaged on 4 runs to ensure reproducibility.

The GENIA corpus significantly differs from the others in its distribution, as it belongs to medical domain. So for GENIA, we initialize word embeddings with word vectors pre-trained on biomedical corpus (Chiu et al., 2016)[5], which are in 200 dimensions.

We also evaluate our method with pre-trained language model embeddings:

- **[Flair]** (Akbik et al., 2018): Pre-trained contextualized character-level embeddings. Here, we use the concatenation of `news-forward` and `news-backward`, forming embeddings of dimension 4096. For GENIA, we use `pubmed-forward` and `pubmed-backward`.

- **[BERT]** (Devlin et al., 2019): Transformer based pre-trained contextual word embeddings. Here we use the `bert-large-uncased` checkpoint, with embeddings of dimension 1024. For each token, we generate the contextualized word embedding by averaging all BERT subword embeddings in the last four layers without fine-tuning. For GENIA, we use BioBERT v1.1 (Lee et al., 2020)[6].

- **[ALBERT]** (Lan et al., 2019): A lite BERT with shared transformer parameters. Here we use the `albert-xxlarge-v2` checkpoint, with embeddings of dimension 4096. For each token, we average all ALBERT subword embeddings in the last four layers without fine-tuning.

We generate Flair embeddings with the library provided by Akbik et al. 2019[7]. We use the implementation by Wolf et al. 2019[8] to generate BERT and ALBERT embeddings.

With pre-trained contextualized embeddings, the model is more prone to overfitting. So we increase the dropout rate by 0.05 for these settings.

4.3 Results of Comparison

Table 3 presents the comparison of our model with existing methods. Our method outperforms all previous methods by a large margin. With conventional word embeddings, our method achieves 80.27, 79.42, 77.78, and 93.70 in terms of F1-score,

[3]http://www.geniaproject.org/genia-corpus/pos-annotation

[4]https://nlp.stanford.edu/projects/glove/

[5]https://github.com/cambridgeltl/BioNLP-2016

[6]https://github.com/naver/biobert-pretrained

[7]https://github.com/zalandoresearch/flair

[8]https://github.com/huggingface/transformers

	ACE-2004			ACE-2005			GENIA			NNE		
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Finkel and Manning 2009	-	-	-	-	-	-	75.4	65.9	70.3	-	-	-
Lu and Roth 2015	70.0	56.9	62.8	66.3	59.2	62.5	74.2	66.7	70.3	-	-	-
Muis and Lu 2017	72.7	58.0	64.5	69.1	58.1	63.1	75.4	66.8	70.8	-	-	-
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	79.8	68.2	73.6	-	-	-
Ju et al. 2018	-	-	-	74.2	70.3	72.2	78.5	71.3	74.7	-	-	-
Wang et al. 2018	74.9	71.8	73.3	74.5	71.5	73.0	78.0	70.2	73.9	77.4	70.1	73.6
Wang and Lu 2018	78.0	72.4	75.1	76.8	72.3	74.5	77.0	73.3	75.1	91.8	91.0	91.4
Sohrab and Miwa 2018	-	-	-	-	-	-	93.2	64.0	77.1	-	-	-
Fisher and Vlachos 2019	-	-	-	75.1	74.1	74.6	-	-	-	-	-	-
Lin et al. 2019	-	-	-	76.2	73.6	74.9	75.8	73.9	74.8	-	-	-
Straková et al. 2019	-	-	77.1	-	-	75.4	-	-	76.4	-	-	-
Pyramid-Basic	80.83	78.86	79.83	79.27	79.37	79.32	77.91	77.20	77.55	93.37	93.91	93.64
Pyramid-Full	81.14	79.42	**80.27**	80.01	78.85	**79.42**	78.60	77.02	**77.78**	93.44	93.95	**93.70**
LM-based												
Xia et al. 2019 [ELMO]	81.7	77.4	79.5	79.0	77.3	78.2	-	-	-	-	-	-
Fisher and Vlachos 2019 [ELMO]	-	-	-	79.7	78.0	78.9	-	-	-	-	-	-
Fisher and Vlachos 2019 [BERT]	-	-	-	82.7	82.1	82.4	-	-	-	-	-	-
Shibuya and Hovy 2019 [BERT]	-	-	-	83.0	82.4	82.7	76.3	74.7	75.5	-	-	-
Luan et al. 2019 [ELMO]	-	-	84.7	-	-	82.9	-	-	76.2	-	-	-
Straková et al. 2019 [BERT]	-	-	84.3	-	-	83.4	-	-	78.2	-	-	-
Straková et al. 2019 [BERT+Flair]	-	-	84.4	-	-	84.3	-	-	78.3	-	-	-
Pyramid-Basic [BERT]	86.08	86.48	86.28	83.95	85.39	84.66	79.45	78.94	79.19	93.97	94.79	94.37
Pyramid-Basic [BERT+Flair]	87.01	86.55	86.78	84.90	86.08	85.49	79.98	78.51	79.24	93.97	94.98	94.47
Pyramid-Basic [ALBERT]	86.54	87.44	86.99	85.20	86.56	85.87	80.07	77.60	78.82	94.11	94.91	94.51
Pyramid-Basic [ALBERT+Flair]	86.63	87.15	86.89	85.10	87.22	86.15	78.48	79.39	78.93	94.18	94.79	94.48
Pyramid-Basic [ALBERT+BERT]	87.65	87.74	87.70	85.24	87.32	86.27	80.12	77.82	78.95	94.28	94.99	94.63
Pyramid-Full [BERT+Flair]	-	-	-	-	-	-	80.31	78.33	**79.31**	-	-	-
Pyramid-Full [ALBERT+BERT]	87.71	87.78	**87.74**	85.30	87.40	**86.34**	-	-	-	94.30	95.07	**94.68**

Table 3: Results of nested NER. Ju et al. 2018 used different dataset split. Straková et al. 2019 introduces two methods, here we report the better one. Bold and underline indicate the best and the second best F1 respectively.

even compatible with some LM-based baselines. A close one is from Straková et al. 2019, which employs many extra features including input forms, lemmas and POS, whereas our method does not. Additionally, our method brings much higher recall values than the other methods.

With pre-trained language model embeddings, specifically with ALBERT+BERT for ACE-2004, ACE-2005, NNE and with BERT+Flair for GENIA, our model achieves state-of-the-art F1 scores: 87.74, 86.34, 79.31, and 94.68 respectively.

4.4 Tuning Number of Layers

We evaluate our method with different L on all datasets. Due to space limit, we only present the results of ACE-2005 in Table 4. The findings on the other datasets are similar.

Results From All Layers We report in Table 4 the detailed results for all entity lengths while tuning L on ACE-2005. Obviously 1-word and 2-word entities account for the majority of entities (77%), where we achieve competitive results. Longer entities see reductions in performance. However, due to our remedy strategy, entities longer than L are still recognized with acceptable performance. Note R(N) is the recall of nested entities, i.e. for layer l, entities nested with other entities shorter than l are also counted in.

Inference Speed Table 4 also shows the inference speed with different L for the basic and full models. Although the basic model does not perform as good as the full model, it is significantly faster. Since the time complexity of our method is $O(TL)$ with T being the number of tokens and L the number of stacked layers, we can further speed up the inference by using smaller L value (e.g. $L = 8$ or 4), while achieving F1 scores higher than most baselines.

4.5 Ablation Study

We conduct ablation study to verify the effectiveness of components of `Pyramid`. Likewise, we only present the results on ACE-2005 here.

Character Embeddings: Using character is a standard technique for NER to dynamically capture orthographic and morphological features. It provides some improvements.

Layer Normalization: `LayerNorm` eliminates the bias and scale difference of the inputs of each

Pyramid-Basic		$L = 32$		$L = 16$		$L = 8$		$L = 4$	
len(e)	# entities	F1	R(N)	F1	R(N)	F1	R(N)	F1	R(N)
all	-	79.3	73.6	79.3	74.4	78.8	73.9	77.6	69.5
1	1706 (56%)	84.0	82.3	84.3	82.5	84.0	83.0	83.4	81.4
2	635 (21%)	79.3	77.5	79.7	78.6	78.8	77.7	78.6	76.2
3	248 (8%)	74.9	75.5	75.3	76.8	75.6	77.5	72.9	73.7
4	140 (5%)	72.1	73.1	71.8	75.0	72.0	73.3	65.7	61.1
5	90 (3%)	73.6	77.5	72.3	78.9	69.3	75.5	63.6	60.3
6-8	106 (3%)	57.9	59.3	56.2	59.3	53.4	56.7	47.7	45.9
9-16	81 (3%)	42.0	36.4	43.1	39.9	42.3	39.5	40.0	36.8
17-	25 (1%)	33.8	26.1	23.0	18.8	27.2	21.7	23.6	18.8
Inference Speed (Basic/Full, words per second) on GTX 1080 Ti									
batch size = 1		708 / 445		842 / 545		1116 / 781		1494 / 1153	
batch size = 4		1526 / 955		2085 / 1361		2987 / 2151		4230 / 3280	
batch size = 16		2949 / 2084		4372 / 3282		6660 / 5169		8999 / 7852	

Table 4: Details of tuning L on ACE-2005. len(e) is the length of entities. R(N) is the recall of nested entities. Numbers below the horizontal lines indicate the results where the remedy solution starts working

Pyramid-Basic	P	R	F1
CharEmbs			
with	79.27	79.37	79.32
without	79.54	77.67	78.59 (-0.73)
LayerNorm			
with	79.27	79.37	79.32
without	79.17	78.01	78.59 (-0.73)
LSTM^{dec}			
shared weights	79.27	79.37	79.32
independent	78.19	78.75	78.47 (-0.85)
ReduceLength			
Conv1d	79.27	79.37	79.32
MeanPooling	79.18	77.77	78.47 (-0.85)
MaxPooling	79.69	77.47	78.56 (-0.76)

Table 5: Ablation study on ACE-2005

decoding layer and improve the F1 score. It also substantially accelerates the converging speed.

Sharing LSTM^{dec}: The jobs of decoding layers are similar: inheriting information from previous layers and recognizing entity representations of length one. Therefore, sharing weights maximizes the use of training data and prevents overfitting.

Method of Reducing Length: We use CNN to reduce the sequence length at each decoding layer. As shown in Table 5, compared with average pooling and maximum pooling, CNN can effectively retain the original semantic information and capture the boundary information.

Pyramid Layers: Apart from the results shown in Table 5, we emphasize that the performance gain of Pyramid owes a lot to the pyramid layers (both normal and inverse ones). As shown in Table 4, reducing L to 4 leads to a drop of F1 (-1.7). It is clear that when $L = 1$, our method degrades to a flat entity recognizer, which cannot handle nested mentions any more.

Dataset Statistics		train	dev	test
Sentences	# total	1599	400	600
	# nested	1594	400	600
	# overlap	230	54	75
Entities	# total	16202	3978	5989
	# nested	14506	3597	5390
	# overlap	511	115	164
Pyramid-Basic		P	R	F1
overall		87.5	86.9	87.2
nested		-	87.4	-
overlap		-	70.1	-

Table 6: Results of Pyramid-Basic with nested and overlapping entities. The dataset is based on part of NNE, with additional program-generated labels.

4.6 Overlapping Entity Recognition

Overlapping mentions usually occur along with the attributive clause in natural language. For example, sentence "The burial site of Sheikh Abbad, who died 500 years ago, is located." contains two overlapping mentions "The burial site of Sheikh Abbad" and "Sheikh Abbad, who died 500 years ago".

Due to lack of datasets for overlapping NER, we create a small dataset. For all sentences in NNE, we find 2599 which contain ", which" or ", who". We use the ELMo-based constituency parser[9] to find attributive clauses together with their modified noun phrases ("Sheikh Abbad, who ..."), and then see if a bigger noun phrase ("the burial site of Sheikh Abbad") contains the noun phrase. Next, while keeping the original annotations, we add these two mentions to form a new dataset where around 14% sentences have overlapping but non-nested entity mentions. This dataset is split randomly into training, dev, and test sets containing 1599, 400, and 600 sentences respectively. Note the additional annotations are not verified by human, meaning they might contain some errors. However, it is still useful for testing the performance of our model for overlapping NER.

The statistics of the data and the experimental results are shown in Table 6. It can be seen that our method can effectively handle overlapping NER.

5 Conclusion

This paper presented Pyramid, a novel layered neural model for nested entity recognition. Our model relies on a layer-wise bidirectional decoding process (with both normal and inverse pyramids),

[9]Stern et al. 2017 with ELMo: https://allennlp.s3.amazonaws.com/models/elmo-constituency-parser-2018.03.14.tar.gz, implemented by Gardner et al. 2018.

allowing each decoding layer to take into account the global information from lower and upper layers. Pyramid does not suffer from layer disorientation or error propagation, and is applicable for the more general overlapping NER. The proposed method obtained state-of-the-art results on four different nested NER datasets, confirming its effectiveness.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61672455), the Key Research and Development Program of Zhejiang Province of China (No. 2020C01024), the Natural Science Foundation of Zhejiang Province of China (No. LY18F020005), and the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

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.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Kate Byrne. 2007. Nested named entity recognition in historical archive text. In *International Conference on Semantic Computing (ICSC 2007)*, pages 589–596. IEEE.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th workshop on biomedical natural language processing*, pages 166–174.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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. Lisbon.

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*, pages 141–150.

Joseph Fisher and Andreas Vlachos. 2019. Merge and label: A novel neural network architecture for nested ner. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5840–5850.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

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)*, 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)*, pages 861–871.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.

2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2019. Sequence-to-nuggets: Nested entity mention detection via anchor-region networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5182–5192.

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, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046.

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)*, pages 1064–1074.

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.

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 E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

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 (CoNLL-2009)*, pages 147–155.

Nicky Ringland, Xiang Dai, Ben Hachey, Sarvnaz Karimi, Cecile Paris, and James R Curran. 2019. Nne: A dataset for nested named entity recognition in english newswire. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5176–5181.

Takashi Shibuya and Eduard Hovy. 2019. Nested named entity recognition via second-best sequence learning and decoding. *arXiv preprint arXiv:1909.02250*.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested ner through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

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.

Thomas Wolf, L Debut, V Sanh, J Chaumond, C Delangue, A Moi, P Cistac, T Rault, R Louf, M Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv, abs/1910.03771*.

Congying Xia, Chenwei Zhang, Tao Yang, Yaliang Li, Nan Du, Xian Wu, Wei Fan, Fenglong Ma, and S Yu Philip. 2019. Multi-grained named entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1430–1440.

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.

Changmeng Zheng, Yi Cai, Jingyun Xu, Ho-fung Leung, and Guandong Xu. 2019. A boundary-aware neural model for nested named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 357–366.