NeurSEG: A Segment Driven Deep Neural Model for Nested Named Entity Recognition

Zheng Wang[1,*,†] , Fei Li[2,*], Cheng Long[1]
[1]SCALE@Nanyang Technological University, Singapore
[2]Baidu Inc., Beijing, China
wangzheng155@huawei.com,lifei21@baidu.com,c.long@ntu.edu.sg

ABSTRACT

Named Entity Recognition (NER) is a fundamental problem in natural language processing (NLP). Apart from flat entities, nested entities are also commonly existed in real-life textual data. However, the current methods are not capable of handling nested structures for NER effectively. In this paper, we propose a novel segment driven modeling method (NeurSEG) for the nested NER problem, which can effectively extract entities from the nested structures in complex nesting scenarios. The proposed NeurSEG model first finds the nested label of each word in a sentence and determines the positional relationships between neighbouring words, and then extracts the entities and predicts the corresponding entity types. In addition, we also propose an augmented training method for further improving the performance. We have conducted extensive experiments based on both flat and nested NER benchmark datasets. The performance results have shown that our proposed NeurSEG model has achieved promising performance while retaining its runtime efficiency for the nested NER task. Moreover, the proposed model has also achieved very competitive results when compared with the existing models for the flat NER task, demonstrating its capability for tackling both nested and flat NER tasks.

CCS CONCEPTS

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

KEYWORDS

nested NER, segment driven, positional relationship, augmented training

ACM Reference Format:
Zheng Wang[1,*,†] , Fei Li[2,*], Cheng Long[1]. 2023. NeurSEG: A Segment Driven Deep Neural Model for Nested Named Entity Recognition. In *Proceedings of the 1st International Workshop on Large Generative Models Meet Multimodal Applications (LGM3A).* ACM, Washington, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.
†Zheng Wang is currently working at Huawei Singapore Research Center.

LGM3A, 28 October – 3 November, 2023, Ottawa, Canada

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

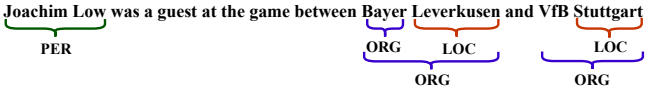

Figure 1: A sentence from GermEval 2014 with 1 flat level, 2 nested levels and 6 entities.

1 INTRODUCTION

The task of named entity recognition (NER) is to find and classify entities such as *person (PER)*, *location (LOC)* and *organization (ORG)* in a sentence. It is widely viewed as the first step towards natural language processing (NLP) applications such as entity linking [12], relation extraction [33], question generation [37] and coreference resolution [1]. Due to the semantic nature of natural language, nested entities are commonly existed in real-life textual data. Figure 1 shows an example sentence taken from GermEval2014 [3]. It is common that a sentence may contain several entities with complex nesting relationships; however, most NER research focus mainly on non-nested entities or *flat entities* such as "Joachim Low", and ignore nested entities such as "Bayer" and "Leverkusen". As such, it causes the loss of fine-grained semantic information and thereby affecting the corresponding downstream tasks.

The core problem of nested NER is to find the nested levels with assigned labels. For example, "Bayer Leverkusen" is a nested level because "Leverkusen" is a *LOC* entity and "Bayer Leverkusen" is an *ORG* entity. In general, nested NER is non-trivial because it involves complex nesting scenarios. While many existing methods have some merits in their solutions to this problem, they suffer from some of the issues such as modeling, effectiveness and efficiency.

For modeling, the existing flat NER solutions assume that there is no overlapping relationship between entities [8] and then model the NER problem as a sequence tagging task [6, 14, 19, 31]. In other words, the words in a sentence will not be involved with more than one label, and such modeling method is apparently not suitable for nested NER. As for the current nested NER solutions, the task is modeled as a classification problem for finding nested entities. For example, Zheng et al. [36] proposed to classify the start-points and end-points of entities in a nested level. However, the start-point and end-point may coexist for certain words in a sentence. For example, "Stuttgart" is the end-point of the entity "VfB Stuttgart", while "Stuttgart" is also a single entity, which means that the word "Stuttgart" is both the start-point and end-point. Such cases are quite common in real-life data, and the current classification methods cannot handle such problem effectively.

For effectiveness in terms of performance, it is highly regarded as the primary goal for NER problems. To the best of our knowledge, almost all existing methods tackle the nested NER task by first

finding the nested boundaries and then identifying the entity types such as *PER* or *LOC*. It means that if entity boundaries are detected wrongly, then it will not be possible to achieve good performance.

For efficiency, as the amount of textual data required to process is generally quite huge and therefore an efficient NER model is highly desirable. However, most of the existing methods require long runtime for extracting entities. Sohrab and Miwa [28] proposed a deep exhaustive model to enumerate all possible combination sequences in a sentence by a predefined maximum entity length. Fisher et al. [10] and Yang et al. [34] proposed merge models to form nested structure iteratively and then label each structure independently. Ju et al. [15] extracted nested entities by dynamically stacking flat NER layers. These methods are not efficient in tackling the NER problem.

In this paper, we propose a segment driven deep neural model called NeurSEG. The proposed model treats the nested NER task as a segment covering problem. First, we obtain each word representation in a given sentence and feed each representation into the BiLSTM network to extract features. Next, in order to detect nested structures, we find the nested label of each word and determine the positional relationships between neighbouring words. Then, we utilize the detected positional relationships to extract the entities and predict their types. In addition, we also propose a new augmented training method to further improve the performance of our model.

In summary, the main contributions of this paper are ad follows:

- We propose a new segment driven modeling method called NeurSEG that models entities as segments. It transforms the complex nested NER task to a solvable segment covering problem by first finding the nested label of each word in a sentence and determining the positional relationships between neighbouring words, and then extracting the entities and predicting the corresponding entity types.
- We study an augmented training method to further improve entity type prediction by augmenting the training dataset with data in which the entity boundaries were wrongly predicted due to entity extraction.
- We conduct extensive experiments based on both flat and nested NER benchmark datasets. The performance results have shown that our proposed model has achieved promising performance while retaining its runtime efficiency for the nested NER task, and achieves very competitive results when compared with the existing models for the flat NER task, demonstrating its capability for tackling both nested and flat NER tasks. A case study is also discussed to give a better intuition to the good performance of our model.

2 RELATED WORK

Flat NER. Huang et al. [14] utilized the bidirectional long short-term memory (BiLSTM) [13] networks as an encoder to learn the contextual representation of words, and then the conditional random fields (CRF) [17] was used as a decoder to label words in the sequence labeling task. It has achieved the superior results on various datasets. Along with their success, this structure has been widely used in the field of NER and adopted by many works for the past few years. Chiu and Nichols [6] used convolutional neural network (CNN) to capture spelling features, and the characters and word characteristics are concatenated as the input of BiLSTM with CRF network. After that, Lample et al. [19] used RNN-BiLSTM-CRF as an alternative. However, the BiLSTM network can only compute token representation one by one, which hinders the full exploitation of GPU's parallelism. Furthermore, Strubell et al. [29] proposed an improved CNN model called ID-CNNs, which can encode words concurrently for large amount of textual data. Vaswani et al. [31] proposed a fully-connected attention-based model with Transformer which relies entirely on self-attention to compute representations of its input and output. It has become a popular model in various NLP applications. Devlin et al. [9] proposed a pre-trained model based on Transformer which has achieved the superior results on eleven natural language processing tasks. Li et al. [20] proposed a boundary-aware Bidirectional Neural Networks (Ba-BNN) model, which addresses key challenges in the NER task by effectively handling boundary tag sparsity, global decoding information, and boundary error propagation. They further enhanced the NER performance with a proposed Modularized Interaction Network (MIN) model, which combines segment-level information and word-level dependencies, and incorporating an interaction mechanism between boundary detection and type prediction [21]. However, all these works only focus on flat NER and ignore nested entities. This leads to the loss of potentially important information and degrade the performance of the downstream tasks.

Nested NER. Lu and Roth [24] introduced a hypergraph-based structure to represent overlapping entities in a sentence, but there is a big gap when mapping the practical nesting structure to the graph structure. After that, the graph model was further improved by Muis and Lu [26]. They assigned tags between each pair of consecutive words to prevent the model from learning spurious structures and structural ambiguity. Wang et al. [32] proposed a transition-based model that learns each sentence with nested mentions and forms a forest. Ju et al. [15] proposed a neural layered model that first extracts the inner entities in a nested structure by the flat NER system and then feeds them to the next layer to extract outer entities. However, one problem of this method is that when an outer entity is detected at the first time, it cannot be extended to detect its inner entity.

Sohrab and Miwa [28] enumerated all possible entity regions as potential entities by a predefined maximum region size, and then classified them into the corresponding entity types or non-entity. This model considers too many irrelevant non-entity regions, which results in too many misclassification and long runtime. Similarly, Tan et al. [30] used two boundary classifiers to identify the start-points and end-points of entities to form regions within a predefined maximum region length. However, the predefined length may loss many potential long entities. Fisher and Vlachos [10] introduced a merge model that first merges tokens and/or entities into entities, and then labels each of them independently. This model has the same problem with the method proposed by Sohrab and Miwa. Zheng et al. [36] combined the methods of Ju et al. and Sohrab et al., and proposed a boundary-aware neural model that first detects the start-points and end-points of entities to form regions. Then, they were classified into the corresponding entity types or non-entity. However, this model cannot tackle the cases when two entities share the same boundary such as the case of "VfB Stuttgart"

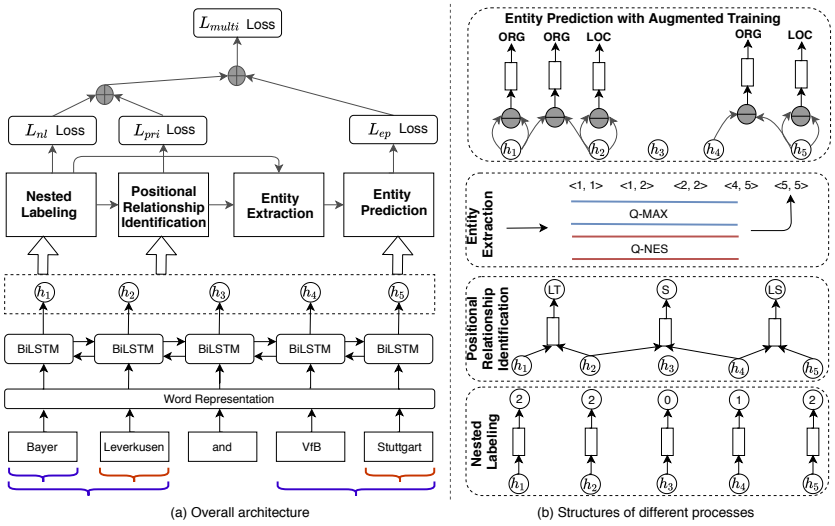

Figure 2: The architecture of the proposed NeurSEG model.

shown in Figure 1. Li et al. [23] modeled NER as a machine reading comprehension (MRC) task, in which the entities are detected as the answers for the MRC queries. In addition, Li et al. [22] studied a segment-enhanced span-based model, which enhances boundary supervision and reduces negative samples. Different from the existing methods, our work models the nested NER task as a segment covering problem, which carefully designs for two level nested entities and detects the candidate entities with arbitrary lengths in near linear complexity.

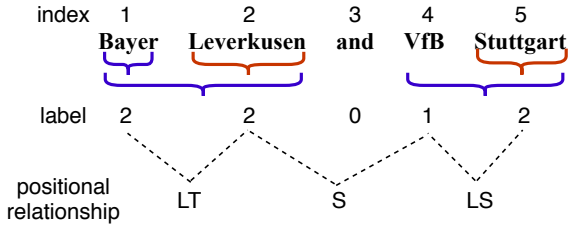

Figure 3: An example for Nested Labeling and Positional Relationship Identification.

3 OUR PROPOSED MODEL

A sentence can be viewed as a sequence of words, and entities in the sentence are considered as entity segments in the given sequence. In particular, entity boundaries are mapped to the boundaries of segments. Thus, we model the nested named entity recognition task as a segment covering problem on a given sequence. Specifically, a given sentence S has its form as $< w_1, w_2, \cdots, w_n >$, where w_i denotes the i-th word, and n denotes the number of words in the sentence. We denote an entity $E[i, j]$ as a maximum continuous portion of S with an entity type such as *LOC* or *PER*, which starts from the i-th word and ends at the j-th word, i.e., $E[i, j] =< w_i, w_{i+1}, \cdots, w_j >$ and $1 \leq i \leq j \leq n$. Therefore, our goal for the nested NER task is to identify all segments and then classify them into the corresponding entity types in a complex nested scenario. Figure 2 shows the architecture of our proposed NeurSEG model for nested NER.

Word Representation. Given an input sentence $S =< w_1, w_2, \cdots, w_n >$, we represent each word w_i as a vector representation for subsequent processing. We follow the previous studies [19, 25] to represent each word by concatenating word embeddings and character-based word representations.

Feature Extraction. Recurrent neural networks (RNNs) are widely used in the field of NER tasks to extract features. In particular, BiLSTM [14, 19] has received great success in recent years. More specifically, given an input sentence S, we obtain the distributed representation of each word $< x_1, x_2, ..., x_n >$ and the representations are fed into a bidirectional LSTM layer to compute the hidden sequences in forward $\overrightarrow{h} =< \overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_n} >$ and backward $\overleftarrow{h} =< \overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_n} >$. Finally, we concatenate $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ as an output at each word, i.e., $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$.

Nested Labeling. As discussed before, we model an entity as a segment. Each word in a sentence may be used to form one or more entities which have the boundaries at the word. Thus, in a nested scenario, the segments may be overlapping. For example, in Figure 3, the word "Bayer" can be used for two entities. One is the single *ORG* entity "Bayer", and the other is the start-point of the *ORG* entity "Bayer Leverkusen". Here, we assign a label to each word in the sentence to denote the number of entities that the word can be used to form. The sentence "Bayer Leverkusen and VfB Stuttgart" in Figure 3 has a label sequence $< 2, 2, 0, 1, 2 >$. The label 0 means the word is non-entity. Similarly, the label 1 (or 2) means there is one entity (or two entities) that can be formed at the word. Therefore, our task is transformed into a classification task for identifying these labels. In particular, for each hidden vector h_i with its feature representation, we feed it into a neural layer with ReLU activation function and then pass it to a softmax classifier such that $o_i = Wh_i + b$ and $d_i = \text{softmax}(o_i)$, where W and b are trainable parameters which can be optimized, and d_i is a predicted label for the i-th word.

Positional Relationship Identification. After obtaining the label for each word by Nested Labeling, we need to identify the different segments from the label sequence. For example, in Figure 3, the subsequence $< 2, 2 >$ denotes three entities "Bayer", "Leverkusen" and "Bayer Leverkusen". To do this, we consider the nested structures as layered structures. For each layer, we aim to find the positional relationship between two neighbouring words which are entity boundaries. Specifically, there are three types of positional relationships between two neighbouring words as follows: Link (L), Tangency (T) and Separation (S). The Link relationship denotes that two words are linked in an entity. For example, "Bayer" and "Leverkusen" have the link relationship in the entity "Bayer Leverkusen". The Tangency relationship denotes that the start-point (or start-word) of an entity is located next to the end-point (or end-word) of another entity. For example, the single entity "Bayer" has the tangency relationship with another single entity "Leverkusen". The Separation relationship considers two situations. First, the start-point of an entity is separated from the end-point of another entity by one or more non-entities. For example, "Leverkusen" and "VfB" have the separation relationship as there is an non-entity "and" between them. Second, a non-entity is separated from a neighbouring entity boundary in multiple nested levels. For example, "VfB" and "Stuttgart" have the separation relationship at the second level as "VfB" is a non-entity.

However, in a complex nesting scenario, positional relationships can coexist. For example, the positional relationships between "Bayer" and "Leverkusen" are both Link and Tangency (LT). Note that the three kinds of positional relationships cannot coexist at the same time in only two levels of nesting. Figure 3 shows the positional relationships for the example sentence of "Bayer Leverkusen and VfB Stuttgart". We only consider positional relationship between neighbouring words which are entity boundaries. The word "and" is a non-entity, and thus it is not part of any segments. In addition, the positional relationships for ("VfB", "Stuttgart") are Link and Separation (LS) as "VfB" is in the entity "VfB Stuttgart" at the first layer; however, it is a non-entity at the second layer, which means it is separated from the single entity "Stuttgart". Thus, the positional relationships of the example sentence are $< LT, S, LS >$, which are then used for entity extraction. In order to identify the positional relationships, we train a classifier based on the hidden vectors of each segment's boundaries.

Entity Extraction. After finding all positional relationships between neighbouring words which are entity boundaries, the next step is to determine the entities. To achieve this, we propose a well-designed data structure called entity queue for extracting entities from an input sequence pattern $< w_1(idx_1, lab_1), pr_1, ..., pr_{k-1}, w_k(idx_k, lab_k) >$ consisting of the following elements: word, word index, label and positional relationship. w denotes a word, idx denotes the index of the word in the sentence, lab denotes the nested label (0, 1 or 2) and pr denotes the positional relationship between two neighbouring entity boundary words. For example, the input sequence of the sentence in Figure 3 is $< Bayer(1, 2), LT, Leverkusen(2, 2), S, VfB(4, 1), LS, Stuttgart(5, 2) >$.

For illustration, we use two entity queues, namely Q-MAX and Q-NES, to extract entities for two nested levels. Q-MAX is used for the maximal nested layer (e.g., "Bayer Leverkusen") and it always contains the maximal segment during the process. Q-NES is used for the nested entity that is nested in the maximal entity (e.g., "Bayer" or "Leverkusen"). Note that the two-level nesting is the most common case occurred in the nested NER problem [27], and our data structure can be extended to tackle higher level nesting problems by adding more entity queues. There are two important operations on the entity queues, namely *push* (or *enqueue*) and *pop* (or *dequeue*). The push operation enqueues a word to the queue tail and the pop operation dequeues a word from queue head. During the processing of the input sequence, we process the words and relationships accordingly. When processing each word, we use the push operation to append the word to the queue. When processing each relationship, we use the relationship to determine whether the pop operation is needed to dequeue a word from the queue Q-MAX or Q-NES.

We discuss how to use the positional relationships (i.e., L, T, S, LS, LT and TS) to extract entities. When the positional relationship pr_i is read, we determine whether the pop operation is needed to perform on Q-MAX or Q-NES . The steps are discussed as follows:

- L Relationship - It does not need to perform the pop operation, because it means an entity has not been formed.
- LT and LS Relationships - It performs the pop operation on Q-NES as there is a nesting case with the next incoming word. The Tangency (T) and Separation (S) relationships indicate the entity in Q-NES has been formed, and we need to pop it from Q-NES. Additionally, the Link (L) relationship in LT and LS indicates that the entity has not been formed in Q-MAX, so we do not need to pop the word from Q-MAX.
- T, TS and S Relationships - It performs the pop operation from Q-MAX and Q-NES as entities have been formed, and we need to pop them from all queues to form the entities.

Figure 4 illustrates the entity extraction process for the sentence "Bayer Leverkusen and VfB Stuttgart". The input sequence $< Bayer(1, 2), LT, Leverkusen(2, 2), S, VfB(4, 1), LS, Stuttgart(5, 2) >$ is processed with the following steps:

1. Read the word "Bayer(1, 2)" and push "Bayer(1, 2)" into Q-MAX and then Q-NES - Note that we perform the push operation two times for the word "Bayer(1, 2)", because there are two boundaries at the word "Bayer".
2. Read the positional relationship LT and pop "Bayer(1, 2)" from Q-NES - One entity $< 1, 1 >$ (i.e., "Bayer") is formed, where $< 1, 1 >$ denotes an extracted segment with the starting index 1 and the ending index 1.
3. Read the word "Leverkusen(2, 2)" and push "Leverkusen(2, 2)" into Q-NES and then Q-MAX.
4. Read the positional relationship S, pop "Bayer(1, 2)" and "Leverkusen(2, 2)" from Q-MAX, and pop "Leverkusen(2, 2)" from Q-NES - Two entities $< 1, 2 >$ (i.e., "Bayer Leverkusen") and $< 2, 2 >$ (i.e., "Leverkusen") are formed.
5. Read the word "VfB(4, 1)" and push it into Q-MAX.
6. Read the positional relationship LS and perform the pop operation on Q-NES - However, as it is empty now, no word will be popped.
7. Read the word "Stuttgart(5, 2)" and push it into Q-NES and then Q-MAX.

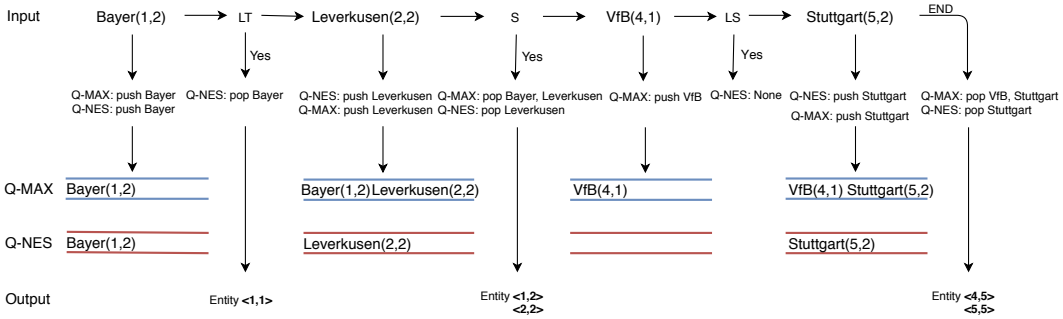

Figure 4: Entity extraction for the sentence "Bayer Leverkusen and VfB Stuttgart".

8. Stop processing when the sentence is finished, and pop all words from Q-MAX and Q-NES - Two entities $< 4, 5 >$ (i.e., "VfB Stuttgart") and $< 5, 5 >$ (i.e., "Stuttgart") are formed.

Entity Prediction. After obtaining an entity, we separate it into entity boundaries and middle-word. For the boundary representation, we take the outputs of the BiLSTM layer corresponding to the boundary words. For the middle-word representation, we average the outputs of the BiLSTM layer in these words. Note that the outputs of the boundary words are included in the middle-word representation to ensure that the middle-word representation will have the corresponding output.

We use the boundaries, middle-word and entity length to represent the entity. Then, we concatenate them as follows:

$$E_{i,j} = \left[h_i; \frac{1}{j-i+1} \sum_{t=i}^{j} h_t; h_j; L_e \right] \quad (1)$$

where $E_{i,j}$ denotes the entity embedding of $E[i, j]$ that starts from the i-th word and ends at the j-th word, and L_e denotes the entity length embedding. We then feed the $E_{i,j}$ to a ReLU activation function and pass it to a softmax output layer to classify its type such as *PER*, *LOC* or *ORG*.

Augmented Training. In this section, we discuss how to train the classifier for entity prediction in details. Specifically, the classifier is guided by ground truth labels. Most of the existing works [35, 36] follow the ideas of teacher forcing [18]. Teacher forcing trains a model using the data with the existing entity boundaries and types in the training set (called D_1). In other words, it ignores the cases that boundaries were wrongly predicted. However, such cases are also informative to help boundary supervision by correcting the misclassified entity regions to improve the performance. Therefore, we assign a new class for the cases with incorrect boundary detection (called D_2). In our training process, we augment our training data with D_2 to form new training data, $D_1 \cup D_2$, to train our model. In particular, D_2 consists of two kinds of incorrect boundary detection cases due to entity extraction as follows: (1) the predicted boundary (i, j) does not have any overlap with an actual entity's boundaries; and (2) the predicted boundary (i, j) has some overlaps with an actual entity's boundaries. Compared with teacher forcing, which considers only D_1 to force the model training, we consider $D_1 \cup D_2$ for augmented training for our model. To alleviate the data imbalance problem for the classifier, we use the focal loss [11] instead of the cross entropy loss. In our empirical evaluation, we found that augmented training can effectively improve the performance of our model.

Recall that there is a loss L_{nl} when classifying the number of entity boundaries in a sentence, a loss L_{pri} for identifying the positional relationships between neighbouring boundaries, and a loss L_{ep} for predicting the entity type. As such, we consider them as multitask training. L_{nl} and L_{pri} are used to optimize boundary detection, and L_{ep} is used to optimize entity type prediction based on the boundaries. Thus, the multitask loss function is defined as follows:

$$L_{multi} = \alpha(\beta \sum L_{nl} + (1-\beta) \sum L_{pri}) + (1-\alpha) \sum L_{ep} \quad (2)$$

where α and β are hyperparameters that are used to balance the importance of each task.

4 EXPERIMENTS

4.1 Experimental Setup

Datasets. Following the experiments in [28, 36], we use two benchmark datasets, namely GermEval2014 [1] [3] and JNLPBA [2] [7], to provide empirical evidence for the performance of our proposed model on the nested and flat NER tasks respectively. The GermEval2014 dataset consists of a total of 31,300 sentences corresponding to 591,006 tokens and marked up with 41,005 entity annotations. Additionally, around 7% of the entities are nested and 15% of them are derivations or partly NEs. The JNLPBA dataset is collected from a nested corpus [7] comprising 2,000 training and 404 testing MEDLINE abstracts. However, only flat and top-most entities are kept in JNLPBA and 5 entity types are finally reserved. We preprocess the dataset following the same settings of Zheng et al. [36] to obtain the development dataset from the training dataset.

Compared Models. We compare our NeurSEG model with several existing models for both nested and flat NER tasks.

For the nested NER task, the following models are adopted:

- Flat Stacked [15] - A novel neural model which identifies nested entities by dynamically stacking flat NER layers.
- Exhaustive [28] - An exhaustive model which enumerates all possible entity regions and classifies them into the corresponding entity types.
- Merge [10] - A merge model which merges tokens and/or entities into entities and then classifies them.

[1] https://sites.google.com/site/germeval2014ner/data
[2] http://www.nactem.ac.uk/tsujii/GENIA/ERtask/report.html

Table 1: Performance results for nested NER based on GermEval2014 dataset.

Models	P(%)	R(%)	F(%)
Flat Stacked [15]	72.9	61.5	66.7
Exhaustive [28][3]	75.0	60.8	67.2
Merge [10]	74.5	64.7	69.3
Boundary [36]	74.5	69.1	71.7
Our NeurSEG	**80.7**	**69.5**	**74.7**

Table 2: Performance results for flat NER (JNLPBA dataset).

Models	P(%)	R(%)	F(%)
BiLSTM-CRF [14]	73.5	68.3	70.8
CNN-BiLSTM-CRF [6]	**74.0**	70.5	72.2
SciBERT [2]	70.7	**80.4**	**75.2**
Exhaustive [28]	69.4	71.9	70.6
Merge [10]	68.0	78.6	72.9
Boundary [36]	68.6	79.0	73.4
Our NeurSEG	**71.6**	**79.5**	**75.3**

- Boundary [36] - A boundary-aware neural model which leverages on entity boundaries and then classifies the entity regions formed by the boundaries.

For the flat NER task, the following models are adopted:

- BiLSTM-CRF [14] - A BiLSTM-CRF model which is a classical baseline for the flat NER task.
- CNN-BiLSTM-CRF [6] - A CNN-BiLSTM-CRF model which has achieved strong results for the flat NER task.
- SciBERT [2] - A contextualized embedding model for scientific text based on BERT, which achieves the superior performance on several tasks.

The nested NER models can be used for flat NER as well, and we provide their results on the JNLPBA dataset.

Parameter Setting. Our model is implemented in PyTorch. We employ pre-trained word embeddings trained on MEDLINE abstracts [5] in JNLPBA. For GermEval 2014, we initialize each word based on the pretrained word embeddings computed with the fastText [4], while the char embeddings and the entity length embeddings are initialized as vectors with dimensions 50 and 20 respectively by random. We use Adam optimizer [16] for training with a mini-batch size of 50. We set the learning rate to 0.0005, the dropout rate to 0.5, the hidden layer size to 200, and the gradient clipping to 5. We evaluate the different settings of the coefficients α and β ($0 < \alpha, \beta < 1$) of multitask loss, and the best performance is achieved when α and β are both set to 0.3. All experiments are conducted on the same machine with 8-cores of Intel(R) Xeon(R) E5-1630 CPU @ 3.70GHz and two Nvidia GeForce-GTX GPU.

4.2 Performance Results for Nested NER

We conduct experiments on the GermEval2014 dataset for nested NER. Table 1 shows the performance results in comparison with other baseline models. As shown in Table 1, the Boundary model has performed better than the Merge, Exhaustive and Flat Stacked models in terms of precision, recall and F-score. When our NeurSEG model is compared with the Boundary model, we observe that our NeurSEG model has achieved about 3% improvements in F-score, and the performance in precision and recall is consistently better than the Boundary model. This is because our model does not need to distinguish the start-points and end-points of entities as proposed by the Boundary model and thus it can recognize any entity regions.

[3]This result is reported by [36], consistent with our own re-implemented results.

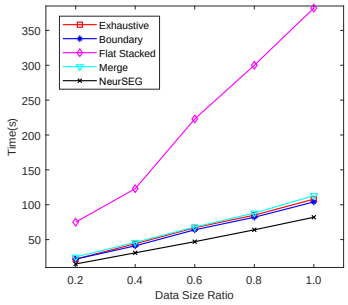

Figure 5: Runtime performance results based on GermEval2014 test dataset.

4.3 Performance Results for Flat NER

We conduct experiments on our NeurSEG model in comparison with three existing flat NER models based on the JNLPBA dataset. The results are given in Table 2. We can observe that SciBERT performs better than the other two flat NER models. Specifically, its F-score is higher than BiLSTM-CRF and CNN-BiLSTM-CRF by about 4.4% and 3.0% respectively. As the nested NER models can be used for the flat NER task, we also conduct experiments on using nested NER models for the flat NER task. Among the nested NER models, our NeurSEG model has achieved the best performance on the flat NER task. When comparing it with the SciBERT model, our NeurSEG model performs slightly better than the SciBERT model by about 0.1% in F-score. It is important to note that SciBERT was designed specifically for the flat NER task, while our NeurSEG model is proposed for the nested NER task. Overall, our model has achieved the promising performance for both flat NER and nested NER tasks.

4.4 Runtime Performance

To evaluate the efficiency of the proposed NeurSEG model for the nested NER task, we conduct experiments for evaluating efficiency based on the GermEval 2014 dataset. As the test dataset is small with only 5100 sentences, we use the entire dataset of 31,300 sentences to test the efficiency of different models. The results are shown in Figure 5. Moreover, we also evaluate the scalability of all the models.

Experimental Results. As shown from Figure 5, we can see that our NeurSEG model has achieved the best runtime performance. For example, it takes about 80 seconds to process the entire dataset and achieves a speed of 7388 words per second (w/s). The Flat Stacked model has the worst runtime performance because it dynamically stacks the flat NER layers that normally followed by a decoder with CRF layer. In addition, the Exhaustive, Merge and Boundary models have similar runtime performance. Compared with other models, our model utilizes the positional relationship between neighbouring entity boundaries and entity queue to identify entity regions. The queue operation executes in constant time,

Table 3: Further performance analysis for nested NER based on GermEval2014 dataset.

Setting	Our SeurSEG			w/o Pre-trained Word Embedding			w/o Augmented Training		
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Overall	80.7	69.5	74.7	78.7	69.1	73.6	73.8	70.2	72.0
PER	83.2	77.0	80.0	81.3	76.3	78.7	74.8	81.7	78.1
LOC	83.5	77.3	80.3	83.0	75.6	79.1	77.1	76.9	77.0
ORG	73.2	63.1	67.8	68.1	61.7	64.7	68.2	61.6	64.7

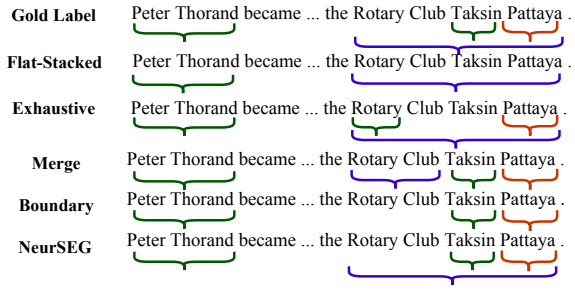

Figure 6: Results for compared models based on an example sentence, with *PER* (green), *ORG* (purple) and *LOC* (red).

and the positional relationship identification is closely related to the number of entities in a sentence. Note that for each sentence, the number of entities is significantly less than the sentence length. Overall, the experimental results confirm with our analysis.

Scalability. We also evaluate the scalability of all the models based on different data size ratios varying from 0.2 to 1.0. As can be seen, all the models scale near linearly with the ratio. We also observe that the Exhaustive, Merge, Boundary and NeurSEG models scale better than the Flat Stacked model, but the difference between them becomes larger as the size of the dataset increases. Additionally, our NeurSEG model is consistently faster than the baseline models.

4.5 Further Performance Analysis

To show the importance of the pre-trained word embedding and augmented training, the performance results based on GermEval2014 are reported in Table 3. The pre-trained word embedding helps improve the overall performance by approximately 2% in terms of precision. This is because the pre-trained word embeddings have a good word distribution representation and it can benefit boundary detection better than random initialization. We also observe that without augmented training, the precision drops by almost 7% in the overall performance. This shows that our augmented training can help correct boundary detection errors, though it affects the recall rate slightly. This is because the use of augmented training can be considered as a tradeoff between precision and recall.

Moreover, we also show the performance results based on three main entity types, namely *PER*, *LOC* and *ORG*, in the dataset which has a total of 12 entity types. As shown in Table 3, we can observe that the overall performance is generally lower than that of *PER* and *LOC* by around 2%-7% in terms of precision, recall and F-score. Furthermore, the recall and F-score of *PER* and *LOC* are higher than that of the overall performance by about 5%. It indicates that the NeurSEG model performs better for the *PER* and *LOC* entity types. The performance of NeurSEG in *ORG* is the worst among the three main entity types, possibly due to the more complex nested relationships of this entity type in the dataset.

4.6 Case Study

To further illustrate the results of our NeurSEG model and the baseline models for nested NER, we present a case study in Figure 6. For Gold Label, the sentence contains one flat level “Peter Thorand” (in green), one nested level “Rotary Club Taksin Pattaya” (in blue), and four entities "Peter Thorand" (*PER*), "Rotary Club Taksin Pattaya" (*ORG*), "Taksin" (*PER*) and "Pattaya" (*LOC*).

After processing the sentence, our NeurSEG model can extract both boundaries and entity labels correctly. As for the other models, all of them can detect the flat level; however, some errors have occurred on the nested level. Specifically, the Flat Stacked model can only find the outer entity "Rotary Club Taksin Pattaya" because it is considered as the whole entity and cannot be extended to detect its inner entity. Therefore, it is unable to find the nested entities "Taksin" and "Pattaya". The Exhaustive model enumerates all potential entities and classifies them into the corresponding entity types. In this example, some errors have occurred for the words "Rotary" and "Taksin" in the classification step. The Merge model merges words into entities. Here, it is unable to merge the words "Rotary Club", "Taksin" and "Pattaya" correctly for the nested level. Therefore, it is unable to recognize the entity "Rotary Club Taksin Pattaya". The Boundary model performs classification based on the start-points and end-points of entities; however, it has limitations when the start-points and end-points coexist. In this example, "Pattaya" is the end-point of the entity "Rotary Club Taksin Pattaya". As "Pattaya" is a single entity, it is also considered as a start-point by default. Then, after the model has predicted the single entity "Pattaya" successfully, it is unable to recognize the entity "Rotary Club Taksin Pattaya". Overall, our NeurSEG model can extract all the nested boundaries and classify them into the corresponding nested entities correctly.

5 CONCLUSION

In this paper, we study the problem of nested named entity recognition (NER) and propose a segment driven modeling method (NeurSEG). In addition, we also propose an augmented training method for further improving the performance of the proposed model. We have conducted extensive experiments based on both flat and nested NER datasets. The performance results have shown that our proposed NeurSEG model has achieved promising performance for the nested NER task and very competitive results when compared with the state-of-the-art models for the flat NER task. Note that the proposed model is currently targeted at tackling two level nested entities. However, it can be extended to deal with higher nested levels by introducing some additional positional relationships and designing new deterministic rules to the entity queues for entity extraction. With that, the overall performance could further be improved.

Acknowledgements. This study is supported under the RIE2020 Industry Alignment Fund Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in kind contribution from Singapore Telecommunications Limited Singtel, through Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU).

REFERENCES

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

[2] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. 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)*. 3606–3611.

[3] Darina Benikova, Chris Biemann, and Marc Reznicek. 2014. NoSta-D Named Entity Annotation for German: Guidelines and Dataset.. In *LREC*. 2524–2531.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://doi.org/10.1162/tacl_a_00051

[5] 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*. 166–174.

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

[7] Nigel Collier and Jin-Dong Kim. 2004. Introduction to the Bio-entity Recognition Task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*. COLING, Geneva, Switzerland, 73–78. https://www.aclweb.org/anthology/W04-1213

[8] Xiang Dai. 2018. Recognizing complex entity mentions: A review and future directions. In *Proceedings of ACL 2018, Student Research Workshop*. 37–44.

[9] 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[10] 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*. Association for Computational Linguistics, Florence, Italy, 5840–5850. https://doi.org/10.18653/v1/P19-1585

[11] Priyal Goyal, R Girshick, K He, P Dollár, and TY Lin. 2017. Focal loss for dense object detection. In *Proc. IEEE CVPR*. 2980–2988.

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

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

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

[15] 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)*. Association for Computational Linguistics, New Orleans, Louisiana, 1446–1459. https://doi.org/10.18653/v1/N18-1131

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980

[17] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289. http://dl.acm.org/citation.cfm?id=645530.655813

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

[19] 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*. Association for Computational Linguistics, San Diego, California, 260–270. https://doi.org/10.18653/v1/N16-1030

[20] Fei Li, Zheng Wang, Siu Cheung Hui, Lejian Liao, Dandan Song, and Jing Xu. 2021. Effective named entity recognition with boundary-aware bidirectional neural networks. In *Proceedings of the Web Conference 2021*. 1695–1703.

[21] Fei Li, Zheng Wang, Siu Cheung Hui, Lejian Liao, Dandan Song, Jing Xu, Guoxiu He, and Meihuizi Jia. 2021. Modularized interaction network for named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 200–209.

[22] Fei Li, Zheng Wang, Siu Cheung Hui, Lejian Liao, Xinhua Zhu, and Heyan Huang. 2021. A segment enhanced span-based model for nested named entity recognition. *Neurocomputing* 465 (2021), 26–37.

[23] Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A Unified MRC Framework for Named Entity Recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 5849–5859.

[24] 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*. 857–867.

[25] Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354* (2016).

[26] 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*. Association for Computational Linguistics, Copenhagen, Denmark, 2608–2618. https://doi.org/10.18653/v1/D17-1276

[27] 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*. 5176–5181.

[28] 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*. Association for Computational Linguistics, Brussels, Belgium, 2843–2849. https://doi.org/10.18653/v1/D18-1309

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

[30] Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary Enhanced Neural Span Classification for Nested Named Entity Recognition.. In *AAAI*. 9016–9023.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need

[32] 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*. Association for Computational Linguistics, Brussels, Belgium, 1011–1017. https://doi.org/10.18653/v1/D18-1124

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

[34] Zhiwei Yang, Jing Ma, Hechang Chen, Yunke Zhang, and Yi Chang. 2021. HiTRANS: A Hierarchical Transformer Network for Nested Named Entity Recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. 124–132.

[35] Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[36] 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)*. Association for Computational Linguistics, Hong Kong, China, 357–366. https://doi.org/10.18653/v1/D19-1034

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