A Neural Layered Model for Nested Named Entity Recognition

Meizhi Ju[1,3], **Makoto Miwa**[2,3] and **Sophia Ananiadou**[1,3]
[1]National Centre for Text Mining, University of Manchester, United Kingdom
[2]Toyota Technological Institute, Japan
[3]Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{meizhi.ju, sophia.ananiadou}@manchester.ac.uk
makoto-miwa@toyota-ti.ac.jp

Abstract

Entity mentions embedded in longer entity mentions are referred to as nested entities. Most named entity recognition (NER) systems deal only with the flat entities and ignore the inner nested ones, which fails to capture finer-grained semantic information in underlying texts. To address this issue, we propose a novel neural model to identify nested entities by dynamically stacking flat NER layers. Each flat NER layer is based on the state-of-the-art flat NER model that captures sequential context representation with bidirectional long short-term memory (LSTM) layer and feeds it to the cascaded CRF layer. Our model merges the output of the LSTM layer in the current flat NER layer to build new representation for detected entities and subsequently feeds them into the next flat NER layer. This allows our model to extract outer entities by taking full advantage of information encoded in their corresponding inner entities, in an inside-to-outside way. Our model dynamically stacks the flat NER layers until no outer entities are extracted. Extensive evaluation shows that our dynamic model outperforms state-of-the-art feature-based systems on nested NER, achieving 74.7% and 72.2% on GENIA and ACE2005 datasets, respectively, in terms of F-score.[1]

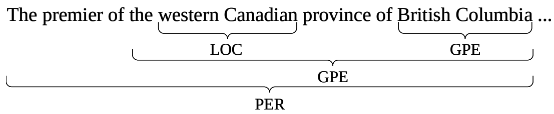

Figure 1: A sentence from ACE2005 (Walker et al., 2006) containing the nested 4 entities nested 3 levels deep.

1 Introduction

The task of named entity recognition (NER) involves the extraction from text of names of entities pertaining to semantic types such as *person (PER)*, *location (LOC)* and *geo-political entity (GPE)*. NER has drawn the attention of many researchers as the first step towards NLP applications such as entity linking (Gupta et al., 2017), relation extraction (Miwa and Bansal, 2016), event extraction (Feng et al., 2016) and co-reference resolution (Fragkou, 2017; Stone and Arora, 2017).

Due to the properties of natural language, many named entities contain *nested entities*: embedded names which are included in other entities, illustrated in Figure 1. This phenomenon is quite common in many domains (Alex et al., 2007; Byrne, 2007; Wang, 2009; Màrquez et al., 2007). However, much of the work on NER copes only with non-nested entities which are also called *flat entities* and neglects nested entities. This leads to loss of potentially important information, with negative impacts on subsequent tasks.

Traditional approaches to NER mainly involve two types of approaches: supervised learning (Ling and Weld, 2012; Marcińczuk, 2015; Leaman and Lu, 2016) and hybrid approaches (Bhasuran et al., 2016; Rocktäschel et al., 2012; Leaman et al., 2015) that combine supervised learning with rules. Such approaches require either domain knowledge or heavy feature-engineering. Recent advances in neural networks enable NER without depending on external knowledge resources through automated learning high-level and abstract features from text (Lample et al., 2016; Ma and Hovy, 2016; Pahuja et al., 2017; Strubell et al., 2017).

In this paper, we propose a novel dynamic neural model for nested entity recognition, without relying on any external knowledge resources or linguistics features. Our model enables sequentially

[1]Code is available at https://github.com/meizhiju/layered-bilstm-crf

Proceedings of NAACL-HLT 2018, pages 1446–1459
New Orleans, Louisiana, June 1 - 6, 2018. ©2018 Association for Computational Linguistics

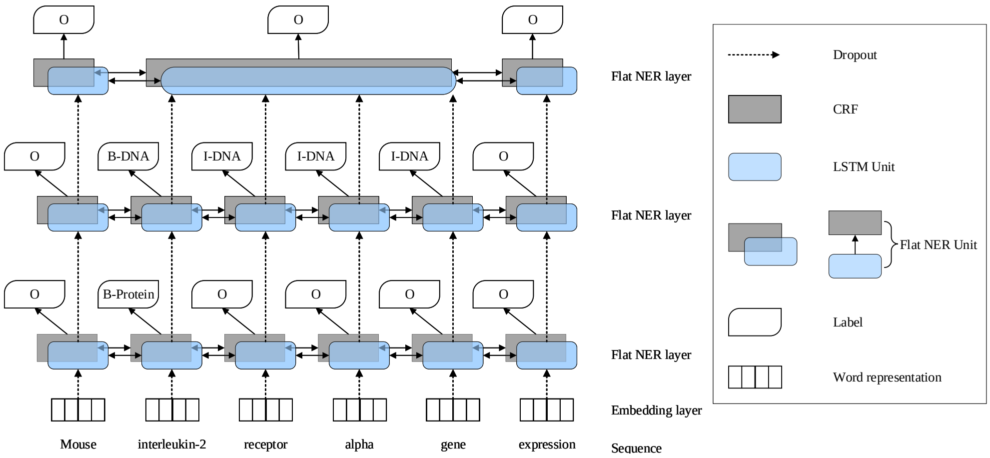

Figure 2: Overview of our layered model architecture. “interleukin-2” and “interleukin-2 receptor alpha gene” are nested entities.

stacking flat NER layers from bottom to up and identifying entities in an end-to-end manner. The number of stacked layers depends on the level of entity nesting and dynamically adjusts to the input sequences as the nested level varies from different sequences.

Given a sequence of words, our model first represents each word using a low-dimensional vector concatenated from its corresponding word and character sequence embeddings. Taking the sequence of the word representation as input, our flat NER layer enables capturing context representation by a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) layer. The context representation is then fed to a CRF layer for label prediction. Subsequently, the context representation from the LSTM layer is merged to build representation for each detected entity, which is used as the input for the next flat NER layer. Our model stops detecting entities if no entities are predicted by the current flat NER layer. Through stacking flat NER layers in order, we are able to extract entities from inside to outside with sharing parameters among the different LSTM layers and CRF layers.

We gain 3.9 and 9.1 percentage point improvements regarding F-score over the state-of-the-art feature-based model on two nested entity corpora: GENIA (Kim et al., 2003) and ACE2005 (Walker et al., 2006), and analyze contributions of inner entities to outer entity detection, drawing several key conclusions.

In addition, experiments are conducted on a flatly annotated corpora JNLPBA (Kim et al., 2004). Our model can be a complete NER model as well for flat entities, on the condition that it is trained on annotations that do not account for nested entities. We obtain 75.55% in terms of F-score that is comparable to the state-of-the-art performance.

2 Neural Layered Model

Our nested NER model is designed based on a sequential stack of flat NER layers that detects nested entities in an end-to-end manner. Figure 2 provides the overview of our model. Our flat NER layers are inspired by the state-of-the-art model proposed in Lample et al. (2016). The layer utilizes one single bidirectional LSTM layer to represent word sequences and predict flat entities by putting one single CRF layer on top of the LSTM layer. Therefore, we refer to our model as Layered-BiLSTM-CRF model. If any entities are predicted, a new flat NER layer is introduced and the word sequence representation of each detected entity by the current flat NER layer is merged to compose a representation for the entity, which is then passed on to the new flat NER layer as its input. Otherwise, the model terminates stacking and hence finishes entity detection.

In this section, we provide a brief description of the model architecture: the flat NER layers and their stacking, the embedding layer and their training.

2.1 Flat NER layer

A flat NER layer consists of an LSTM layer and a CRF layer. The LSTM layer captures the bidirectional context representation of sequences and subsequently feeds it to the CRF layer to globally decode label sequences.

LSTM is a variant of recurrent neural networks (RNNs) (Goller and Kuchler, 1996) that incorporates a memory cell to remember the past information for a long period of time. This enables capturing long dependencies, thus reducing the gradient vanishing/explosion problem existing in RNNs. We employ bidirectional LSTM with no peephole connection. We refer the readers to Hochreiter and Schmidhuber (1997) for more details of LSTM used in our work.

CRFs are used to globally predict label sequences for any given sequences. Given an input sequence $X = (x_1, x_2, \ldots, x_n)$ which is the output from the LSTM layer, we maximize the log-probability during training. In decoding, we set transition costs between illegal transitions, e.g., transition from O to I-PER, as infinite to restrict illegal labels. The expected label sequence $y = (y_1, y_2, \ldots, y_n)$ is predicted based on maximum scores in decoding.

2.2 Stacking flat NER layers

We stack a flat NER layer on the top of the current flat NER layer, aiming to extract outer entities. Concretely, we merge and average current context representation of the regions composed in the detected entities, as described in the following equation:

$$m_i = \frac{1}{end - start + 1} \sum_{i=start}^{end} z_i, \quad (1)$$

where z_i denotes the representation of the i-th word from the flat NER layer, and m_i is the merged representation for an entity. The region starts from a position $start$ and ends at a position end of the sequence. This merged representation of detected entities allows us to treat each detected entity as a single token, and hence we are able to make the most of inner entity information to encourage outer entity recognition. If the region is detected as a non-entity, we keep the representation without any processing. The processed context representation of the flat NER layer is used as the input for the next flat NER layer.

2.3 Embedding layer

The input for the first NER layer is different from the remaining flat NER layers since the first layer has no previous layers. We thus represent each word by concatenating character sequence embeddings and word embeddings for the first flat NER layer. Figure 3 describes the architecture of the embedding layer to produce word representation.

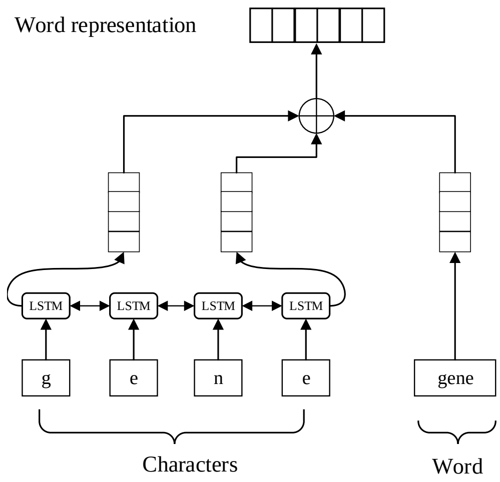

Figure 3: Word representation of a word ‘gene’. We concatenate the outputs of character embedding from LSTM and word embedding to obtain its final word representation.

Following the successes of Ma and Hovy (2016) and Lample et al. (2016) in utilizing character embeddings on the flat NER task, we also represent each word with its character sequence to capture the orthographic and morphological features of the word. Each character is mapped to a randomly initialized vector through a character lookup table. We feed the character vectors comprising a word to a bidirectional LSTM layer and concatenate the forward and backward representation to obtain the word-level embedding.

Differently from the character sequence embeddings, the pretrained word embeddings are used to initialize word embeddings. When evaluating or applying the model, words that are outside of the pretrained embeddings and training dataset are mapped to an *unknown* (UNK) embedding, which is randomly initialized during training. To train the UNK embedding, we replace words whose frequency is 1 in the training dataset with the UNK embedding with a probability 0.5.

2.4 Training

We prepare the gold labels based on the conventional BIO (Beginning, Inside, Out of entities) tagging scheme to represent a label attached to each word.

As our model detects entities from inside to outside, we keep the same order in preparing the gold labels for each word sequence. We call it the *detection order rule*. Meantime, we define that each entity region in the sequence can only be tagged once with the same entity type, referred to as the *non-duplicate rule*. For instance, in Figure 2, "interleukin-2" is tagged first while "interleukin-2 receptor alpha gene" is subsequently tagged following the above two rules. When assigning the label O to non-entity regions, we only follow the detection order rule. As a result, two gold label sequences {O, B-Protein, O, O, O, O} and {O, B-DNA, I-DNA, I-DNA, I-DNA, O} are assigned to the given word sequence "Mouse interleukin-2 receptor alpha gene expression" as shown in Figure 2. With these rules, the number of labels for each word equals the nested level of entities in the given word sequence.

We employ mini-batch training and update the model parameters using back-propagation through time (BPTT) (Werbos, 1990) with Adam (Kingma and Ba, 2014). The model parameters include weights, bias, transition costs, and embeddings of characters. We disable updating the word embeddings.[2] During training, early stopping, L2-regularization and dropout (Hinton et al., 2012) are used to prevent overfitting. Dropout is employed to the input of each flat NER layer. Hyper-parameters including batch size, number of hidden units in LSTM, character dimensions, dropout rate, Adam learning rate, gradient clipping and weight decay (L2) are all tuned with Bayesian optimization (Snoek et al., 2012).

3 Evaluation Settings

We employed three datasets for evaluation: GENIA[3] (Kim et al., 2003), ACE2005[4] (Walker et al., 2006) and JNLPBA[5] (Kim et al., 2004). We briefly explain the data and task settings and then introduce model and experimental settings.

3.1 Data and Task Settings

We performed nested entity extraction experiments on GENIA and ACE2005 while we conducted flat entity extraction on the JNLPBA dataset. For the details of data statistics and preprocessing, please refer to the supplementary materials.

GENIA involves 36 fine-grained entity categories among total 2,000 MEDLINE abstracts. Following the same task settings as in Finkel and Manning (2009) and Lu and Roth (2015), we collapsed all DNA subcategories as DNA. The same setting was applied to RNA, protein, cell line and cell type categories. We used same test portion as Finkel and Manning (2009), Lu and Roth (2015) and Muis and Lu (2017) for the direct comparison.

ACE2005 contains 7 fine-grained entity categories. We made same modifications described in Lu and Roth (2015) and Muis and Lu (2017) by keeping files from bn, bw, nw and wl and spitting them into training, development and testing datasets at random following same ratio 8:1:1, respectively.

JNLPBA defines both training and testing datasets. These two datasets are composed of 2,000 and 404 MEDLINE abstracts, respectively. JNLPBA is originally from the GENIA corpus. However, only flat and topmost entities in JNLPBA are kept while nested and discontinuous entities are removed. Like our preprocessing on the GENIA corpus, subcategories are collapsed and only 5 entity types are finally reserved. We randomly chose the 90% sentences of the original training dataset as our training dataset and the remaining as our development dataset.

Precision (P), recall (R) and F-score (F) were used for the evaluation metrics in our tasks. We define that if the numbers of gold entities and predictions are all zeros, the evaluation metrics all equal one hundred percent.

3.2 Model and Experimental Settings

Our model was implemented with Chainer[6] (Tokui et al., 2015). We initialized word embeddings in GENIA and JNLPBA with the pretrained embeddings trained on MEDLINE abstracts (Chiu et al., 2016). For ACE2005, we initialized each word with the pretrained embeddings which are trained by Miwa and Bansal (2016). Except for the word embeddings, parameters of word embeddings were initialized with a normal distribution. For LSTM, we initialized hidden states, cell state and all the bias terms as 0 except for the forget gate

[2] We tried updating and disabling updating word embeddings. The former trial did not work.

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

[4] https://catalog.ldc.upenn.edu/ldc2006t06

[5] http://www.nactem.ac.uk/tsujii/GENIA/ERtask/report.html

[6] https://chainer.org/

bias that was set as 1. For other hyper-parameters, we chose the best hyper-parameters via Bayesian optimization. We refer the readers to the supplemental material for the settings of the hyper-parameters of the models and Bayesian optimization.

For ablation tests, we compared with our layered-BiLSTM-CRF model with two models that produce the input for next flat NER layer in different ways. The first model is called *layered-BiLSTM-CRF w/o layered out-of-entities* which uses the input of the current flat NER layer for out-of-entity words. We name the second model as *layered-BiLSTM-CRF w/o layered LSTM* as it skips all intermediate LSTM layers and only uses output of embedding layer to build the input for the next flat NER layer. Please refer to supplemental material for the introduced two models.[7]

To investigate the effectiveness of our model on different nested levels of entities, we evaluated the model performance on each flat NER layer on GENIA and ACE2005 test datasets.[8] When calculating precision and recall measurements, we collected the predictions and gold entities from the corresponding flat NER layer. Since predicted entities on a specific flat NER layer might be from other flat NER layers, we defined extended precision (EP), extended recall (ER) and extended F-score (EF) to measure the performance. We calculated EP by comparing the predicted entities in a specific flat NER layer with all the gold entities, and ER by comparing the gold entities in a specific flat NER layer with all the predicted entities. EF was calculated in the same way with F.

In addition to experiments on nested GENIA and ACE2005 datasets, flat entity recognition was conducted on the JNLPBA dataset. We trained our flat model that only kept the first flat NER layer and removed the following stacking layers. We follow the hyper-parameters settings by Lample et al. (2016) for this evaluation.

[7]We examined the contributions of predicted labels of the current flat NER layer to the next flat NER layer. For this, we introduced label embeddings into each test by combining the embedding with context representation. Experiments show that appending label embedding hurts the performance of our model while gain slight improvements in the rest 2 models on development datasets.

[8]We removed entities which were predicted in previous flat NER layers during evaluation.

4 Results and Analysis

4.1 Nested NER

Table 1 presents the comparisons of our model with related work including the state-of-the-art feature-based model by Muis and Lu (2017). Our model outperforms the state-of-the-art models with 74.7% and 72.2% in terms of F-score, achieving the new state-of-the-art in the nested NER tasks. For GENIA, our model gained more improvement in terms of recall with enabling extract more nested entities without reducing precision. On ACE2005, we improved recall with 12.2 percentage points and obtained 5.1% relative error reductions. Compared with GENIA, our model gained more improvements in ACE2005 in terms of F-score. Two possible reasons account for it. One reason is that ACE2005 contains more deeper nested entities (maximum nested level is 5) than GENIA (maximum nested level is 3) on the test dataset. This allows our model to capture the potentially 'nested' relations among nested entities. The other reason is that ACE2005 has more nested entities (37.45%) compared with GENIA (21.56%).

Table 2 shows the results of models on the development datasets of GENIA and ACE2005, respectively. From this table, we can see that our model, which only utilizes context representation for preparation of input for the next flat NER layer, performs better than the rest two models. This demonstrates that introducing input of the current flat NER layer such as skipping either representation for any non-entity or words or all intermediate LSTM layers hurts performance. Compared with the layered-BiLSTM-CRF model, the drop of the performance in the layered-BiLSTM-CRF w/o layered out-of-entities model reflects the skip of representation for out-of-entity words leads to the decline in performance. This is because the representation of non-entity words didn't incorporate the current context representation as we used the input rather than the output to represent them. By analogy, the layered BiLSTM-CRF w/o layer LSTM model skips representation for both entities and non-entity words, resulting in worse performance. This is because, when skipping all intermediate LSTM layers, input of the first flat NER layer, i.e., word embeddings, is passed to the remaining flat NER layers. Since word embeddings do not contain context representation, we fail to incorporate the context representation when we use

Settings	GENIA			ACE2005		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Finkel and Manning (2009)	75.4	65.9	70.3	-	-	-
Lu and Roth (2015)	72.5	65.2	68.7	66.3	59.2	62.5
Muis and Lu (2017)	75.4	66.8	70.8	69.1	58.1	63.1
Our model	**78.5**	**71.3**	**74.7**	**74.2**	**70.3**	**72.2**

Table 1: Comparisons of our model with the state-of-the-art models on nested NER.

Settings	GENIA			ACE2005		
	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
Layered-BiLSTM-CRF	78.27	75.97	77.10	75.37	69.41	72.27
Layered-BiLSTM-CRF w/o layered non-entities	76.55	77.01	76.78	72.90	65.54	69.02
Layered-BiLSTM-CRF w/o layered LSTM	75.76	74.60	75.18	69.94	61.94	65.70

Table 2: Performances of ablation tests on development datasets.

Entity type	P (%)	R (%)	F (%)
DNA	74.43	69.68	71.98
RNA	90.29	79.48	84.54
Protein	80.48	73.20	76.67
Cell Line	77.83	65.65	71.22
Cell Type	76.36	68.07	71.97
Overall	78.59	71.33	74.79

Table 3: Results of all entities for each type in GENIA test dataset.

Entity type	P (%)	R (%)	F (%)
PER	78.82	77.37	78.09
LOC	54.54	43.47	48.38
ORG	63.25	54.20	58.38
GPE	76.92	78.98	77.94
VEH	61.53	48.48	54.23
WEA	66.66	53.73	59.50
FAC	49.19	35.26	41.07
Overall	74.27	70.34	72.25

Table 4: Results of all entities for each type in ACE2005 test dataset.

the word embeddings as the input for the flat NER layers. Therefore, we have no chance to take advantage of the context representation and instead we only manage to use the word embeddings as the input for flat NER layers in this case.

Table 3 and Table 4 describe the performance for each entity type in GENIA and ACE2005 test datasets, respectively. In GENIA, our model performed best in recognizing entities with type *RNA*. This is because most of the entities pertaining to *RNA* mainly end up either with "mRNA" or *RNA*. These two words are informative indicators of *RNA* entities. For entities in rest entity types, their performances are close to the overall performance. One possible reason is that there are many instances to model them. This also accounts for the high performances of entity types such as *PER*, *GPE* in ACE2005. The small amounts of instances of entity types like *FAC* in ACE2005 is one reason for their under overall performances. We refer readers to supplemental material for statistics details.

When evaluating our model on top level which contains only outermost entities, the precision, recall and F-score were 78.19%, 75.17% and 76.65% on GENIA test dataset. For ACE2005, the corresponding precision, recall and F-score were 68.37%, 68.57% and 68.47%. Compared with the overall performance listed in Table 1, we obtained higher top level performance on GENIA but lower performance in ACE2005. We discuss details of this phenomena in the following tables.

Table 5 shows the performances of each flat NER layer in GENIA test dataset. Among all the stacking flat NER layers, our model resulted in the best performance regarding standard evaluation metrics on the first flat NER layer which contains the predictions for the gold innermost entities. When the model went to deeper flat NER layers, the performance dropped gradually as the number of gold entities decreased. However, the performance for predictions on each flat

Layer	P (%)	R (%)	F (%)	EP (%)	ER (%)	EF (%)	#Predictions	#Gold Entities
Layer 1	72.86	69.82	71.31	78.46	71.06	74.57	4,783	4,991
Layer 2	56.88	27.59	37.15	81.15	73.98	77.39	276	569
Layer 3	0.00	0.00	0.00	0.00	60.00	0.00	1	15

Table 5: Results of layer evaluation on GENIA test dataset.

Layer	P (%)	R (%)	F (%)	EP (%)	ER (%)	EF (%)	#Predictions	#Gold Entities
Layer 1	74.46	73.39	73.92	75.84	73.77	74.79	2,894	2,936
Layer 2	60.28	50.49	54.95	66.19	58.41	62.05	423	505
Layer 3	51.02	24.51	33.11	51.02	37.25	43.06	49	102
Layer 4	0.00	0.00	0.00	0.00	10.00	0.00	0	10
Layer 5	0.00	0.00	0.00	0.00	0.00	0.00	0	1

Table 6: Results of layer evaluation on ACE2005 test dataset.

NER layer was different in terms of extended evaluation metrics. For the first two flat NER layers, performance of extended evaluation is better than the performance of standard evaluation. It indicates that gold entities correspond to some of the predictions on the specific flat NER layer are from other flat NER layers. This may lead to the zero performances for the last flat NER layer. In addition, performance on the second flat NER layer was higher than it was on the first flat NER layer in terms of extended F-score. This demonstrates that our model is able to obtain higher performance on top level of entities than innermost entities.

Table 6 lists the results of each flat NER layer on ACE2005 test dataset. Similar to GENIA, the first flat NER layer achieved better performance than the rest flat NER layers. Performances decreased in a bottom-to-up manner regarding model architecture. This phenomena was the same with the extended evaluation performances, which reflects that some of the predictions in a specific flat NER layer were detected in other flat NER layers. Unlike rising tendency (except last flat NER layer) regarding extend F-score in GENIA, performance in ACE2005 was in downtrend. This accounts for the fact that F-score on top level was lower than it on the fist flat NER layer. Even though the decline trend in extended F-score, the first flat NER layer contained the largest proportion of predictions for the gold entities, the overall performance on all nested entities showed in Table 1 was still high. Unlike GENIA, our model in ACE2005 stopped before reaching the maximum nested level of entities. It indicates our model failed to model the appropriate nested levels. This is one of the reasons that account for the zero predictions on the last flat NER layer. One reason is that our model The sparse instances on the high nested levels could be another reason that resulted in the zero performances on the last flat NER layer.

4.2 Flat NER

Compared with the state-of-the-art work on JNLPBA (Gridach, 2017) which achieved 75.87% in terms of F-score, our model obtained 75.55% in F-score. Since both the model by Gridach (2017) and our flat model are based on Lample et al. (2016), so it is reasonable that both models were able to get comparable performance.

4.3 Error analysis

We showed the error types and their statistics both for all nested entities and each flat NER layer on GENIA and ACE2005 test datasets. From ACE2005 test dataset, 28% of predictions were incorrect in 200 sentences which were selected at random. Among these errors, 39% of them were because their text spans were assigned with other entity types. We call this type of errors *type error*. The main reason is that most of them are pronouns and co-refer to other entities which are absent in the sentence. Taking this sentence “whether that is true now, we can not say” as an example, “we” is annotated as *ORG* while our model labeled it as *PER*. Lack of context information such as the absence of co-referent entities leads our model to make the wrong decisions. In addition, 30% of the errors were caused by that incorrect predictions were predicted as only parts of gold entities with correct entity types. This error type is referred to as *partial prediction error*. This might be due to these gold entities tend to clauses or independent sentences, thus possibly containing many modifiers. For example, in this sentence "A man who has been to Baghdad many times and can tell us with great knowledge exactly what it's going to be like to fight on those avenues in that sprawling city of Baghdad - Judy .", "A man who has been to Baghdad many times and can tell us with great knowledge exactly what it's going to be like to fight on those avenues in that sprawling city of Baghdad" is annotated as *PER* while our model could only extract "A man who has been to Baghdad many times" and predicted it as *PER*.

Errors on the first flat NER layer, we got 41% in type error and 11% of partial prediction error, respectively. Apart from this, our model recognized predictions from other flat NER layers, leading to 5% errors. We define this error type as *layer error*. Unlike the first flat NER layer, 26% of errors were caused by layer error. Additionally, 17% of the errors belong to type error. In particular, 22% errors were due to the type error. As for the last flat NER layer, 40% errors were caused by partial prediction error. The rest errors were different from the mentioned error types. One possible reason is that we have less gold entities to train this flat NER layer compared with previous flat NER layers. Another reason might be the error propagation.

Similarly, 200 sentences were randomly selected from GENIA test dataset. We got 20% errors of predictions in the subset. Among these errors, 17% and 24% of errors were separately due to type error and partial prediction error. In addition, 24% of the predictions on the first flat NER layer were incorrect. Among them, the top error types were layer error, partial prediction error and type error, accounting for 21%, 18% and 13%, respectively. Errors on the second flat NER layer were mainly caused by type error and the and partial prediction error.

5 Related Work

The success of neural networks has boosted the performance of flat named NER in different domains (Lample et al., 2016; Ma and Hovy, 2016; Gridach, 2017; Strubell et al., 2017). Such models achieved the state of the art without any hand-crafted features and external knowledge resources.

Contrary to flat NER, much fewer attempts have emphasized the nested entity recognition. Existing approaches to nested NER (Shen et al., 2003; Alex et al., 2007; Finkel and Manning, 2009; Lu and Roth, 2015; Xu and Jiang, 2016; Muis and Lu, 2017) mainly rely on hand-crafted features. They also failed to take advantage of the dependencies among nested entities. Our model enables capturing dependencies and automatic learning high-level abstract features from texts.

Early work regarding nested NER involve mainly hybrid systems that combined rules with supervised learning algorithms. For example, Shen et al. (2003), Zhou et al. (2004) and Zhang et al. (2004) employed a Hidden Markov Model to GENIA to extract inner entities and then used rule-based methods to obtain the outer entities. Furthermore, Gu (2006) extracted nested entities based on SVM which were trained separately on both inner entities and outermost entities without putting the hidden relations between nested entities into consideration. All these methods failed to capture the dependencies between nested entities. One trial work is that Alex et al. (2007) separately built a inside-out and outside-in layered CRFs which were able to use the current guesses as the input for next layer. They also cascaded separate CRFs of each entity type by using output from previous CRFs as features of current CRFs, yielding best performance in their work. One of the main drawbacks in the cascading approach was that it failed to handle nested entities sharing the same entity type, which were quite common in natural languages.

Finkel and Manning (2009) proposed a discriminative constituency tree to represent each sentence where the root node was used for connection. All entities were treated as phrases and represented as subtrees following the whole tree structure. Unlike our linguistic features independent model, Finkel and Manning (2009) used a CRF-based approach driven by entity-level features to detect nested entities

Later on, Lu and Roth (2015) built hyper-graphs that allow edges to connect multiple nodes to represent both the nested entities and their references (a.k.a. mentions). One issue in their approach is the spurious structures of hyper-graphs as they enumerate combinations of nodes, types and boundaries to represent entities. In addition, they fail to encode the dependencies among embedded entities using hyper-graphs. In contrast, our model enables nested entity representation by merging representation of multiple tokens composed in the entity and considers it as the longer entity representation. This allows us to represent outer entities based on inner entity representation, thus managing to capture the relations between inner and outer entities, and hence overcoming the spurious entity structure problem.

As an improvement in overcoming spurious structure issue in Lu and Roth (2015), Muis and Lu (2017) further incorporated mention separators along with features to yield better performance on nested entities. Both Lu and Roth (2015) and Muis and Lu (2017) rely on hand-crafted features to extract nested entities without incorporating hidden dependencies in nested entities. In contrast, we make the most of dependencies of nested entities in our model to encourage outer entity recognition by automatic learning of high-level and abstract features from sequences.

Shared tasks dealing with nested entities like SemEval-2007 Task 9[9] and GermEval-2014[10] were held in order to advance the state-of-the-art on this issue. Additionally, as subtasks in KBP 2015[11] and KBP 2016[12], one of the aims in tri-lingual Entity Discovery and Linking Track (EDL) track was extracting nested entities from textual documents varying from English, Chinese and Spanish. Following this task, Xu and Jiang (2016) firstly developed a new tagging scheme which is based on fixed-size ordinally-forgetting encoding (FOFE) method for text fragment representation. All the entities along their contexts were represented using this novel tagging scheme. Different from the extensively used LSTM-RNNs in sequence labeling task, a feed-forward neural network was used to predict labels on entity level for each fragment in any of given sequences. Additionally, Li et al. (2017) used the model proposed in Lample et al. (2016) to the extract both flat entities and components composed in nested and discontinuous entities. Another BiLSTM was applied to combine the components to get nested and discontinuous entities. However, these methods failed to capture and utilize the inner entity representation to facilitate outer entity detection.

[9] http://nlp.cs.swarthmore.edu/semeval/tasks/index.php

[10] https://sites.google.com/site/germeval2014ner/

[11] https://tac.nist.gov//2015/KBP/

[12] https://tac.nist.gov//2016/KBP/

6 Conclusion

This paper presented a dynamic layered model which takes full advantage of inner entity information to encourage outer entity recognition in an end-to-end manner. Our model is based on a flat NER layer consisting of LSTM and CRF, so our model is able to capture context representation of input sequences and globally decode predicted labels at a flat NER layer without relying on feature-engineering. Our model automatically stacks the flat NER layers with sharing the parameters of LSTM and CRF in the layers. The stacking continues until the current flat NER layer predicts sequences as all outside of entities, which enables stopping dynamically stacked flat NER layers. Each flat NER layer receives the merged context representation as input for outer entity recognition, based on the predicted entities from the previous flat NER layer. With this dynamic end-to-end model, our model is able to outperform existing models, achieving the-state-of-art on two nested NER tasks. In addition, the model can be flexibly simplified as a flat NER model by removing components cascaded after the first NER layer.

Extensive evaluation shows that utilization of inner entities significantly encourages outer entities detection with improvements of 3.9 and 9.1 percentage points in F-score on GENIA and ACE2005, respectively. Additionally, utilization of only current context representation contributes to the performance improvement than use of context representation from multi-layers.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. The first author is financially supported by the University of Manchesters 2016 Presidents Doctoral Scholar Award. Sophia Ananiadou acknowledges BBSRC BB/P025684/1 Japan Partnering Award and BB/M006891/1 Empathy. This research has also been carried out with funding from AIRC/AIST and results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *Proceedings of the Workshop on*

BioNLP 2007: Biological, Translational, and Clinical Language Processing. Association for Computational Linguistics, Association for Computational Linguistics, Stroudsburg, PA, USA, pages 65–72. http://dl.acm.org/citation.cfm?id=1572392.1572404.

Balu Bhasuran, Gurusamy Murugesan, Sabenabanu Abdulkadhar, and Jeyakumar Natarajan. 2016. Stacked ensemble combined with fuzzy matching for biomedical named entity recognition of diseases. *Journal of biomedical informatics* 64(Supplement C):1–9. https://doi.org/https://doi.org/10.1016/j.jbi.2016.09.009.

Kate Byrne. 2007. Nested named entity recognition in historical archive text. In *ICSC*. IEEE Computer Society, pages 589–596. http://dblp.uni-trier.de/db/conf/semco/icsc2007.html#Byrne07.

Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. 2016. How to train good word embeddings for biomedical NLP. *Proceedings of the 15th Workshop on Biomedical Natural Language Processing* pages 166–174. http://www.aclweb.org/anthology/W16-2922.

HC Cho, N Okazaki, M Miwa, and J Tsujii. 2010. Nersuite: a named entity recognition toolkit. *Tsujii Laboratory, Department of Information Science, University of Tokyo, Tokyo, Japan* http://nersuite.nlplab.org/.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, volume 2, pages 66–71. http://anthology.aclweb.org/P16-2011.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, Association for Computational Linguistics, Singapore, pages 141–150. http://www.aclweb.org/anthology/D/D09/D09-1015.

Pavlina Fragkou. 2017. Applying named entity recognition and co-reference resolution for segmenting english texts. *Progress in Artificial Intelligence* 6(4):325–346. https://doi.org/10.1007/s13748-017-0127-3.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by back-propagation through structure. In *Neural Networks, 1996., IEEE International Conference on*. IEEE, volume 1, pages 347–352. https://doi.org/10.1109/ICNN.1996.548916.

Mourad Gridach. 2017. Character-level neural network for biomedical named entity recognition. *Journal of biomedical informatics* 70:85–91. https://doi.org/10.1016/j.jbi.2017.05.002.

Baohua Gu. 2006. Recognizing nested named entities in GENIA corpus. In *Proceedings of the HLT-NAACL BioNLP Workshop on Linking Natural Language and Biology*. Association for Computational Linguistics, New York, New York, LNLBioNLP '06, pages 112–113. http://www.aclweb.org/anthology/W/W06/W06-3318.

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*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2671–2680. https://www.aclweb.org/anthology/D17-1284.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580. http://arxiv.org/abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. GENIA corpus a semantically annotated corpus for bio-textmining. *Bioinformatics* 19(suppl. 1):i180–i182.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 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*. Association for Computational Linguistics, pages 70–75.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. https://arxiv.org/abs/1412.6980.

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, pages 260–270. http://www.aclweb.org/anthology/N16-1030.

Robert Leaman and Zhiyong Lu. 2016. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics* 32(18):2839–2846. https://doi.org/10.1093/bioinformatics/btw343.

Robert Leaman, Chih-Hsuan Wei, and Zhiyong Lu. 2015. tmChem: a high performance approach for chemical named entity recognition and normalization. *Journal of cheminformatics* 7(1):S3. https://doi.org/10.1186/1758-2946-7-S1-S3.

Fei Li, Meishan Zhang, Bo Tian, Bo Chen, Guohong Fu, and Donghong Ji. 2017. Recognizing irregular entities in biomedical text via deep neural networks. *Pattern Recognition Letters* 105:105–113. https://doi.org/10.1016/j.patrec.2017.06.009.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'12, pages 94–100. http://dl.acm.org/citation.cfm?id=2900728.2900742.

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*. Association for Computational Linguistics, Lisbon, Portugal, pages 857–867. http://aclweb.org/anthology/D15-1102.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, pages 55–60. http://www.aclweb.org/anthology/P14-5010.

Michał Marcińczuk. 2015. Automatic construction of complex features in conditional random fields for named entities recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*. INCOMA Ltd. Shoumen, BULGARIA, Hissar, Bulgaria, pages 413–419. http://www.aclweb.org/anthology/R15-1054.

Lluís Màrquez, Luis Villarejo, M. A. Martí, and Mariona Taulé. 2007. Semeval-2007 task 09: Multilevel semantic annotation of catalan and spanish. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval '07, pages 42–47. http://dl.acm.org/citation.cfm?id=1621474.1621482.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures pages 1105–1116. http://www.aclweb.org/anthology/P16-1105.

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, pages 2598–2608. https://www.aclweb.org/anthology/D17-1276.

Vardaan Pahuja, Anirban Laha, Shachar Mirkin, Vikas Raykar, Lili Kotlerman, and Guy Lev. 2017. Joint learning of correlated sequence labeling tasks using bidirectional recurrent neural networks pages 548–552. https://doi.org/10.21437/Interspeech.2017-1247.

Tim Rocktäschel, Michael Weidlich, and Ulf Leser. 2012. ChemSpot: a hybrid system for chemical named entity recognition. *Bioinformatics* 28(12):1633–1640.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2003. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*. Association for Computational Linguistics, Association for Computational Linguistics, Sapporo, Japan, pages 49–56. https://doi.org/10.3115/1118958.1118965.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. Curran Associates Inc., USA, pages 2951–2959. http://dl.acm.org/citation.cfm?id=2999325.2999464.

M. Stone and R. Arora. 2017. Identifying nominals with no head match co-references using deep learning. *CoRR* abs/1710.00936. https://arxiv.org/abs/1710.00936.

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, pages 2670–2680. https://www.aclweb.org/anthology/D17-1283.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*. volume 5.

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

Yefeng Wang. 2009. Annotating and recognising named entities in clinical notes. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACLstudent '09, pages 18–26. http://dl.acm.org/citation.cfm?id=1667884.1667888.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78(10):1550–1560. https://doi.org/10.1109/5.58337.

Mingbin Xu and Hui Jiang. 2016. A FOFE-based local detection approach for named entity recognition and mention detection. *arXiv preprint arXiv:1611.00801* https://arxiv.org/abs/1611.00801.

Jie Zhang, Dan Shen, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2004. Enhancing HMM-based biomedical named entity recognition by studying special phenomena. *Journal of biomedical informatics* 37(6):411–422. https://doi.org/https://doi.org/10.1016/j.jbi.2004.08.005.

Guodong Zhou, Jie Zhang, Jian Su, Dan Shen, and Chewlim Tan. 2004. Recognizing names in biomedical texts: a machine learning approach. *Bioinformatics* 20(7):1178–1190. https://doi.org/10.1093/bioinformatics/bth060.

A Data Statistics and Preprocessing

Statistics of GENIA, ACE2005 and JNLPBA are described in Tables 7, 8 and 9, respectively.

We used NERSuite (Cho et al., 2010) for GENIA to perform tokenization while Stanford CoreNLP (Manning et al., 2014) was used for ACE2005. The JNLPBA dataset has already been went through tokenization and sentence splitting, so we did not apply any preprocessing.

For GENIA, we had to manually revolve the following two issues, in addition to the above preprocessing.

One of the issues we had in this corpus is the removal of discontinuous entities during parsing. In provided GENIA XML file, each flat entity is annotated with 'lex' (lexical) and 'sem' (semantics) attributes while discontinuous and nested entities may have none, one or two attributes when these entities embed with each other, making it difficult to extract the strictly nested ones. Taken the text "recombinant human nm23-H1, -H2, mouse nm23-M1, and -M2 proteins" as an example, there are six discontinuous entities, "recombinant human nm23-H1 protein", "recombinant human H2 protein", "recombinant mouse nm23-M1 protein", "recombinant mouse nm23-M2 protein", "mouse nm23-M2" and "human nm23-H2", and two nested entities, "mouse nm23-M1" and "human nm23-H1". We extract these nested entities based on symbol * appeared 'lex' attribute which

Item	Train	Dev.	Test
Documents	1,599	189	212
Sentences	15,022	1,669	1,855
Split percentage	81%	9%	10%
DNA	7,921	1061	1,283
RNA	730	140	117
Protein	29,032	2,338	3,098
Cell Line	3,149	340	460
Cell Type	6,021	563	617
Outermost entity	42,462	4,020	4,942
Nested level	4	3	3
Entities in level 1	42,846	4,060	4,991
Entities in level 2	3,910	381	569
Entities in level 3	91	1	15
Entities in level 4	1	0	0
Entity avg. length	2.87	3.13	2.93
Multi-token entity	33951	3554	4203
Overall entities	46,853	4,442	5,575

Table 7: Statistics of GENIA.

Item	Train	Dev.	Test
Documents	370	43	51
Sentences	9,849	1,221	1,478
FAC	924	83	173
GPE	4,725	486	671
LOC	763	81	69
ORG	3,702	479	559
PER	13,050	1,668	1,949
VEH	624	81	66
WEA	652	94	67
Outermost entity	18,455	2,285	2,724
Nested level	6	4	5
Entities in level 1	19,676	2,429	2,936
Entities in level 2	3,934	448	505
Entities in level 3	731	85	102
Entities in level 4	90	10	10
Entities in level 5	7	0	1
Entities in level 6	2	0	0
Entity avg. length	2.28	2.33	2.28
Multi-token entity	10,577	1,323	1,486
Overall entities	24,440	2,972	3,554

Table 8: Statistics of ACE2005.

Item	Train	Dev.	Test
Sentences	16,691	1,855	3,856
Split percentage	90%	10%	-
DNA	8,649	884	1,056
RNA	863	88	118
Protein	27,263	3,006	5,067
Cell Line	3,459	371	500
Cell Type	6,045	673	1,921
Overall entities	46,279	5,022	8,662

Table 9: Statistics of JNLPBA.

Hyper params	Range	Best
Batch size	[16 – 256]	67
No. of hidden units	200, 250, 300	200
Dim. of char. emb.	[15 – 50]	35
Dropout rate	[0.1 – 0.5]	0.2144
Learning rate	[0.001 – 0.02]	0.00754
Gradient clipping	[5 – 50]	27
Weight decay (L2)	[10^{-8} – 10^{-3}]	4.54^{-5}

Table 10: Value range and best value of tuned hyper parameters in GENIA.

Hyper params	Range	Best
Batch size	[16 – 256]	91
No. of hidden units	200, 250, 300	200
Dim. of char. emb.	[15 – 50]	28
Dropout rate	[0.1 – 0.5]	0.1708
Learning rate	[0.001 – 0.02]	0.00426
Gradient clipping	[5 – 50]	11
Weight decay (L2)	[10^{-8} – 10^{-3}]	9.43^{-5}

Table 11: Value range and best value of tuned hyper parameters in ACE2005.

Hyper Parameters	Initialized Value
Acquisition Function	gp_hedge
n-calls	10
n_random_state	None
n_random_starts	10
Acquisition Optimizer	lbfgs
n_restarts_optimizer	100
noise	gaussian
n_points	50000
xi	0.1
n_jobs	1

Table 12: Hyper parameters used of Bayesian Optimization.

is an connection indicator of the separated texts in discontinuous entities. Meanwhile, each of the separated texts has no 'sem' attribute unless itself is an innermost entity. Unfortunately, there are some inconsistent cases such as "c-fos and c-jun transcripts" where symbol * should be in the 'lex' attribute as the discontinuous entity "c-fos transcript" is connected by "c-fos" and "transcript" while "c-jun transcript" is connected by "c-jun" and "transcript". These two entities share the same text "transcript". However, each of them is annotated with two attributes: 'lex' and 'sem', following the same annotation for flat entities. Although it is possible to ignore the latter entity based on 'lex' attribute and its belonging sentence, this rule fails to deal with entity "c-jun gene" in the example of "c-fos and c-jun genes" as the 'lex' of "c-jun gene" is mistaken as "c-jun genes". Therefore, in this case, we ignored "c-fos transcript" and instead kept the "c-jun transcripts" as a flat entity.

Another issue is the incomplete tokenization. The label assignment to one word was conducted on the word-level instead of character level, but there are entities that correspond to parts of words. An example is "NF-YA subunit", which contains two protein entities: "NF-Y" and "A subunit". To cope with this problem, we treat both two entities as false negative entities in training dataset as there are only 13 such entities in the training data set.

B Bayesian Optimization Setting

The hyper-parameters which were tuned for our model are listed in Table 10 and Table 11. These hyper-parameters are tuned by Bayesian optimization with the hyper parameters listed in Table 12.

C Model Structure

Figure 4 shows the model architecture when we skip all intermediate LSTM layers and only word embeddings are used to produce the input for the next flat NER layer.

Figure 5 describes the model architecture when we skip the representation of non-entity words to prepare the input for the next flat NER layer. Concretely, we merge and average representation following Equation 1. For the predicted non-entity words, however, we skip the LSTM layer and directly use their corresponding representation from the input rather than the output context representation.

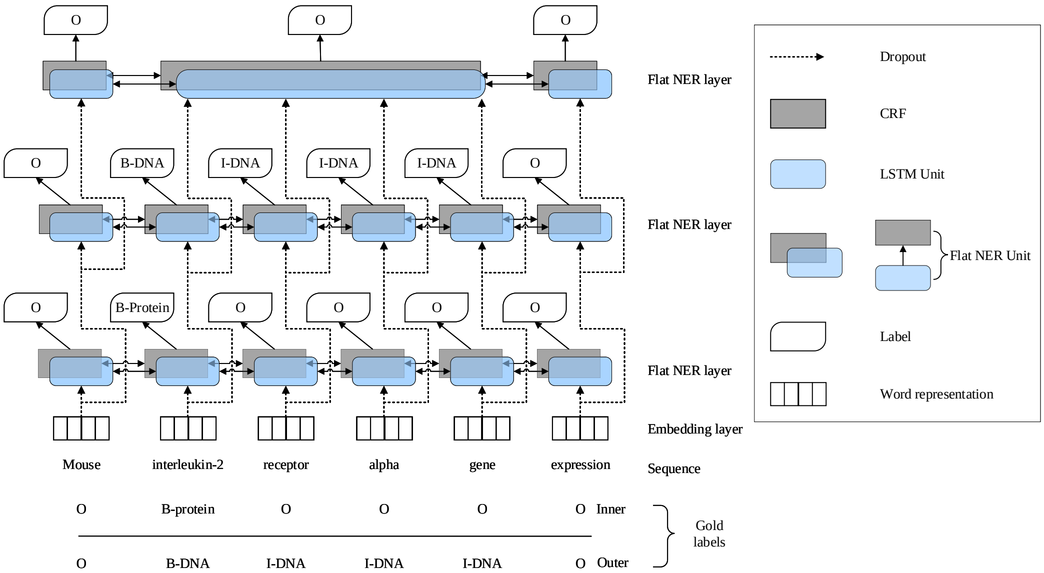

Figure 4: Overview of the model architecture with skipping representation for non-entity words. "interleukin-2" and "interleukin-2 receptor alpha gene" are nested entities.

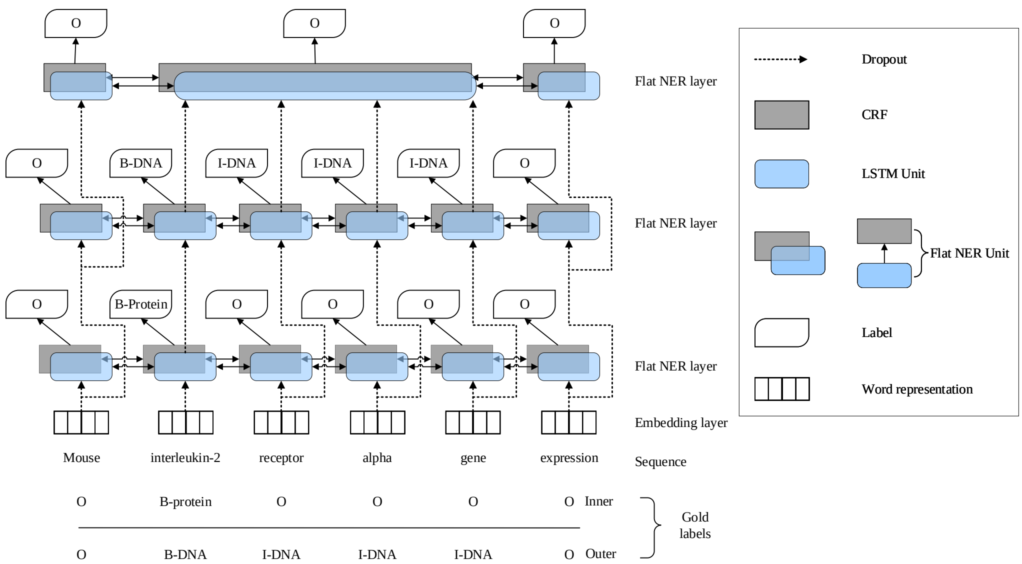

Figure 5: Overview of the model architecture with skipping representation for whole sequence. "interleukin-2" and "interleukin-2 receptor alpha gene" are nested entities.