A Neural Pipeline Approach for the PharmaCoNER Shared Task using Contextual Exhaustive Models

Mohammad Golam Sohrab[†], **Pham Minh Thang**[†],
Makoto Miwa[†,‡], and **Hiroya Takamura**[†]
[†]Artificial Intelligence Research Center (AIRC)
National Institute of Advanced Industrial Science and Technology (AIST),
2-4-7 Aomi, Koto-ku, Tokyo, 135-0064, Japan
[‡]Toyota Technological Institute, Japan
{sohrab.mohammad, pham.thang, takamura.hiroya}@aist.go.jp,
makoto-miwa@toyota-ti.ac.jp

Abstract

We present a neural pipeline approach that performs named entity recognition (NER) and concept indexing (CI), which links them to concept unique identifiers (CUIs) in a knowledge base, for the PharmaCoNER shared task on pharmaceutical drugs and chemical entities. We proposed a neural NER model that captures the surrounding semantic information of a given sequence by capturing the forward- and backward-context of bidirectional LSTM (Bi-LSTM) output of a target span using contextual span representation-based exhaustive approach. The NER model enumerates all possible spans as potential entity mentions and classify them into entity types or no entity with deep neural networks. For representing span, we compare several different neural network architectures and their ensembling for the NER model. We then perform dictionary matching for CI and, if there is no matching, we further compute similarity scores between a mention and CUIs using entity embeddings to assign the CUI with the highest score to the mention. We evaluate our approach on the two sub-tasks in the shared task. Among the five submitted runs, the best run for each sub-task achieved the F-score of 86.76% on Sub-task 1 (NER) and the F-score of 79.97% (strict) on Sub-task 2 (CI).

1 Introduction

The PharmaCoNER (Gonzalez-Agirre et al., 2019) shared task[1] is an open challenge that allows participants to use any methodology and knowledge sources for the clinical records with protected health information. The task aims at two sub-tasks in pharmaceuticals drug and clinical domain: named entity recognition (NER), which is officially called NER offset and entity classification, and concept indexing (CI). Among these sub-tasks, we focus on NER since NER has drawn considerable attentions as the first step towards many natural language processing (NLP) applications including relation extraction (Miwa and Bansal, 2016), event extraction (Feng et al., 2016), and co-reference resolution (Fragkou, 2017). Recently, deep neural networks have shown impressive performance on named entity recognition in several domains (e.g., Lample et al. (2016)). Such models achieved state-of-the-art results without requiring any hand-crafted features or external knowledge resources.

In this paper, we present a pipeline approach that addresses both NER and CI. We mainly focus on NER and employ a neural exhaustive model (Sohrab and Miwa, 2018; Sohrab et al., 2019) for NER. The model detects flat and nested entities by reasoning over all the spans within a specified maximum size. Unlike the existing models that rely on token-level labels, our model directly employs an entity type as the label of a span. Each span is represented as the combination of the boundary and inside representations by using the outputs of bidirectional long short-term memory (Bi-LSTM). We employ and compare different span representations following (Sohrab and Miwa, 2018; Sohrab et al., 2019) that leads to propose a new contextual exhaustive models. The original model (Sohrab and Miwa, 2018) simply treated all the tokens in a span equally by taking the average of LSTM outputs corresponding to tokens inside the span for inside representation and concatenated them with boundary representation where context of each span is totally ignored. Sohrab et al. (2019) proposed several extensions for the representation including contextual span representations and several different inside representations. In this approach, the contextual span representations are considered to capture only the previous and next time steps of LSTM output of

[1] https://2019.bionlp-ost.org/tasks

Proceedings of the 5th Workshop on BioNLP Open Shared Tasks, pages 47–55
Hong Kong, China, November 4, 2019.

a target span, where the surrounding context of a sequence from beginning to target span and end to target span as forward- and backward-context are ignored. Unlike the previous methods (Sohrab and Miwa, 2018; Sohrab et al., 2019), the proposed contextual exhaustive approach captures the surrounding context representation of a given sequence by capturing the forward- and backward-context of Bi-LSTM output of a target span; we describe the details in Section 3.1.3. Besides, the contextual exhaustive approach is extended to leverage the output of a morphological analyser. The spans with the representations are classified into their entity types or non-entity. With the mentions predicted by the NER module, we map them to a knowledge base (KB) (i.e., SNOMED-CT) by direct dictionary matching and similarity scores between mentions and the names of their candidate CUI terms. The best run for each sub-task achieved the F-score of 86.76% on sub-task 1 (NER) and the F-scores of 79.97% on sub-task 2 (CI).

2 Related Work

Most NER work focus on flat entities. Lample et al. (2016) proposed a LSTM-CRF (conditional random fields) model and this has been widely used and extended for the flat NER, e.g., Akbik et al. (2018). In recent studies of neural network based flat NER, Gungor et al. (2018, 2019) have shown that morphological analysis using additional word representations based on linguistic properties of the words, especially for morphologically rich languages such as Turkish and Finnish, improves the NER performances further compared with using only representations based on the surface forms of words.

Recently, nested NER has been widely interested in NLP. Zhou et al. (2004) detected nested entities in a bottom-up way. They detected the innermost flat entities and then found other NEs containing the flat entities as sub-strings using rules on the detected entities. The authors reported an improvement of around 3% in the F-score under certain conditions on the GENIA data set (Collier et al., 1999). Recent studies show that the conditional random fields (CRFs) can produce significantly higher tagging accuracy in flat or nested (stacking flat NER to nested representation) NERs (Son and Minh, 2017). Ju et al. (2018) proposed a novel neural model to address nested entities by dynamically stacking flat NER layers until no outer entities are extracted. A cascaded CRF layer is used after the LSTM output in each flat layer. The authors reported that the model outperforms state-of-the-art results by achieving 74.5% in F-score on the GENIA data set.

Sohrab and Miwa (2018) proposed a neural model that detects nested entities using exhaustive approach and outperforms the state-of-the-art results by achieving 77.1% in terms of F-score on the GENIA data set. Sohrab et al. (2019) further extended the span representations for entity recognition and addressed sensitive span detection tasks in the MEDDOCAN (MEDical DOCument ANonymization) shared task[2], and the system achieved 93.12% and 93.52% in terms of F-score for NER and sensitive span detection, respectively.

3 Pipeline Approach for NER and Concept Indexing

The pipeline approach consists of two modules:

- Named entity recognition that uses a contextual neural exhaustive approach
- Concept indexing (CI) that generates the list of unique SNOMED concept identifiers of the mentions that are detected by the NER module for each document.

3.1 Neural Named Entity Recognition

We solve the NER task, first by employing a neural exhaustive model (Sohrab and Miwa, 2018; Sohrab et al., 2019) that leads to implement a new contextual exhaustive approach, exhaustively considers all possible contextual spans in a sentence using a single neural network. The model detects nested entities by enumerating all possible contextual spans. The model is built upon a shared bidirectional LSTM (Bi-LSTM) layer, and we consider several different representations for the contextual span using the outputs of Bi-LSTM. Figure 1 shows the contextual exhaustive model to detect the possible mentions. The proposed neural contextual exhaustive model consists of embedding, bidirectional LSTM and exhaustive layers. we will explain each layer in the following subsections.

[2] http://temu.bsc.es/meddocan/

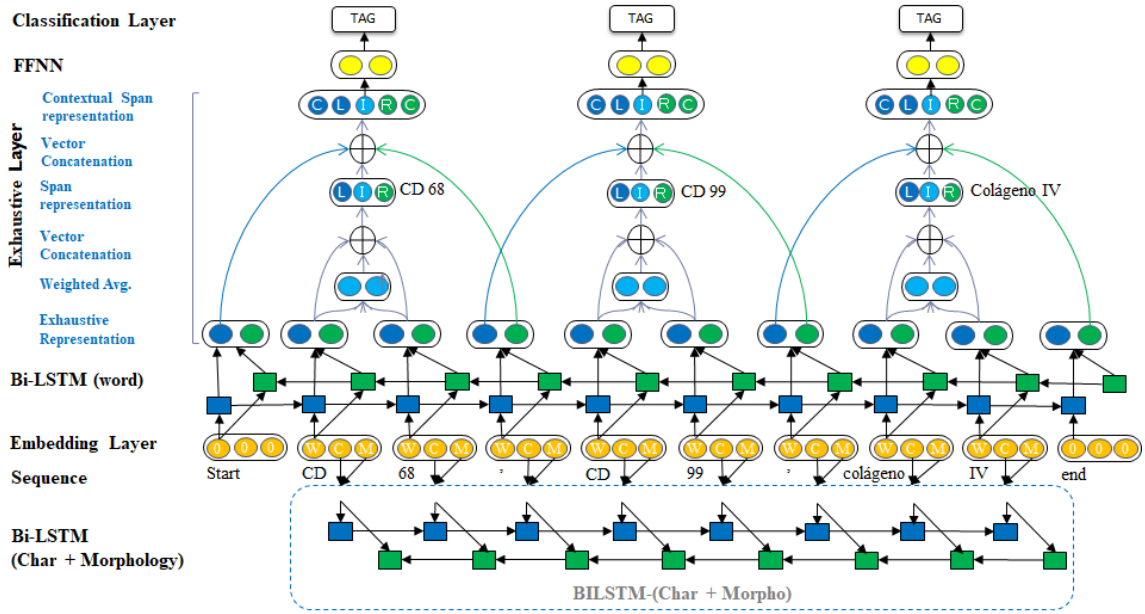

Figure 1: An overview of the exhaustive contextual span representations model. To compute the contextual span representations of 'CD 99', the model concatenates the left-, right-, and inside-representations of Bi-LSTM output vector and further concatenates the contextual information that are represented with the forward LSTM output vector of 'CD' in the previous time step and the backward LSTM output vector of '99' in the previous time step.

3.1.1 Embedding Layer

In the embedding layer, each word is represented by concatenating the pre-trained word embedding and character-based word representation, where we encode the character-level information of the word. The character-based word representation is obtained by feeding the sequence of character embeddings comprising a word to Bi-LSTM and concatenate the forward and backward output representations. Besides, we leverage the morphological analyzer[3] to generate morphological tags, where the tag for each input word is generated by merging the lemma and part-of-speech tag of the word. Then each tag produced by the morphological analyzer is treated as a sequence of characters and encoded using the character-level information using randomly initialized character embeddings. Specifically, we fed the sequence to a separate Bi-LSTM and concatenate the forward and backward outputs to obtain the morphological representation of a word.

3.1.2 Bidirectional LSTM Layer

Given an input sentence sequence $X = \{x_1, ..., x_n\}$ where x_i denotes the i-th word and n denotes the number of words in the sentence sequence, the distributed embeddings of the words in the sequence from the embedding layer are fed into a Bi-LSTM layer. The Bi-LSTM layer computes the hidden vector sequence in forward $\overrightarrow{\mathbf{h}} = \left[\overrightarrow{\mathbf{h}_1}, \overrightarrow{\mathbf{h}_2}, \ldots, \overrightarrow{\mathbf{h}_n}\right]$ and backward $\overleftarrow{\mathbf{h}} = \left[\overleftarrow{\mathbf{h}_1}, \overleftarrow{\mathbf{h}_2}, \ldots, \overleftarrow{\mathbf{h}_n}\right]$ manners. We concatenate the forward and backward outputs as $\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}\right]$, where $[;]$ denotes concatenation.

3.1.3 Exhaustive Layer

The exhaustive layer enumerates all possible spans by exhaustive combination. We generate all possible spans with the sizes less than or equal to the maximum span size L, which is a predefined hyper-parameter. We use (i, k) to represent the span from i to k inclusive, where $1 \leq i < k \leq n$ and $k - i < L$. We represent each span using the outputs of the shared underlying LSTM layer and represent span with different ways as in explained later. We then feed the representation of each segmented span to a rectified linear unit (ReLU) as an activation function. Finally, the output of the activation layer is passed to a softmax output layer to classify the span into a specific entity type.

In the latter part of this section, we introduce the span representations and its several enhancements.

Contextual Span Representations with Averaging For contextual span representations (Sohrab

[3] https://github.com/PlanTL-SANIDAD/SPACCC_POS-TAGGER

et al., 2019), we represent the span with three separate representations: the surrounding context representation, the boundary representation for span detection and the inside representation for semantic type classification. We capture the context representation of a given sequence from Bi-LSTM output $\mathbf{h_i}$. Specifically, we obtain the contextual span representation by capturing the forward- and backward-context of Bi-LSTM output of a target span (i, k) by concatenating vector output of previous $\overrightarrow{\mathbf{h}}_{i-1}$ in forward manner, and output of previous $\overleftarrow{\mathbf{h}}_{i-1}$ in backward manner. The boundary representation is prepared to capture both ends of the span. For this, we rely on the outputs of the Bi-LSTM layer corresponding to the boundary words of a target span. The inside representation is prepared to capture its semantic type by encoding the whole semantic information of the span. We use the average of all the outputs corresponding to the words in the span for the inside representation. Following the above contextual, boundary, and inside representations, we represent the representation $\mathbf{R}(i,k)^{[F,L,A,R,B]}$ (Forward-context, Left-boundary, inside with Average, Right-boundary, and Backward-context) of the span (i, k) as follows:

$$\mathbf{R}(i,k)^{[F,L,A,R,B]} = \left[\overrightarrow{\mathbf{h}}_{i-1}; \mathbf{h}_i; \frac{1}{k-i+1}\sum_{j=i}^{k}\mathbf{h}_j; \mathbf{h}_k; \overleftarrow{\mathbf{h}}_{i-1}\right]. \quad (1)$$

Contextual Span Representations using Attention We also try an attention mechanism (Bahdanau et al., 2015) instead of the average over words in each span. Specifically, we replace the inside representations using attention mechanism as follows:

$$\alpha_t = \mathbf{w}_\alpha FFNN_\alpha\left(\overleftrightarrow{\mathbf{x}}_t\right), \quad (2)$$

$$\alpha_{i,t} = \frac{\exp(\alpha_t)}{\sum_{k=start(i)}^{end(i)}\exp(\alpha_k)}, \quad (3)$$

$$\overline{\mathbf{x}}_i = \sum_{k=start(i)}^{end(i)}\alpha_{i,t}\overleftrightarrow{\mathbf{x}}_t, \quad (4)$$

where $\overleftrightarrow{\mathbf{x}}_t$ is the concatenated output of the Bi-LSTM layer over a span. $\overline{\mathbf{x}}_i$ is a weighted sum of word vectors in span (i, k). Instead of Equation (1), we obtain the representation $\mathbf{R}(i,k)^{[F,L,\overline{A},R,B]}$ ($\overline{A}$ for inside with Attention-based representation) of the span (i, k) as follows:

$$\mathbf{R}(i,k)^{[F,L,\overline{A},R,B]} = \left[\overrightarrow{\mathbf{h}}_{i-1}; \mathbf{h}_i; \overline{\mathbf{x}}_i; \mathbf{h}_k; \overleftarrow{\mathbf{h}}_{i-1}\right]. \quad (5)$$

Contextual LSTM-Minus-based Span Representations We also try LSTM-Minus (Wang and Chang, 2016) for the boundary representation[4]. The left boundary is computed as the representation of the previous word of the span subtracted from the representation of the last word of the current span. Similarly, the right boundary is computed as the representation of the next word of the span subtracted from the representation of the first word of the current span. In contextual LSTM-Minus-based span representations of an input sequence, we compute the forward- and backward-context of a target span as the same manner that stated to represent the forward- and backward-context representations of $\mathbf{R}(i,k)^{[F,L,A,R,B]}$. We obtain the representation $\mathbf{R}(i,k)^{[F,\overline{L},A,\overline{R},B]}$ ($\overline{L}$ and $\overline{R}$ for Left- and Right-boundary based on LSTM-Minus, respectively) of the span (i, k) as follows:

$$\mathbf{R}(i,k)^{[F,\overline{L},A,\overline{R},B]} = [\overrightarrow{\mathbf{h}}_{i-1}; \mathbf{h}_k - \mathbf{h}_{i-1}; \frac{1}{k-i+1}\sum_{j=i}^{k}\mathbf{h}_j; \mathbf{h}_i - \mathbf{h}_{k+1}; \overleftarrow{\mathbf{h}}_{i-1}]. \quad (6)$$

Furthermore, the LSTM-Minus based representation using attention can be considered as:

$$\mathbf{R}(i,k)^{[F,\overline{L},\overline{A},\overline{R},B]} = [\overrightarrow{\mathbf{h}}_{i-1}; \mathbf{h}_k - \mathbf{h}_{i-1}; \overline{\mathbf{x}}_i; \mathbf{h}_i - \mathbf{h}_{k+1}; \overleftarrow{\mathbf{h}}_{i-1}]. \quad (7)$$

Base Span Representations We further consider representations without context representation (Sohrab and Miwa, 2018), which we denote base span representations. For the base span representations, we generate representations by eliminating forward- and backward-context from Equations (1), (5)–(7) and they can be rewritten respectively as:

$$\mathbf{R}(i,k)^{[L,A,R]} = \left[\mathbf{h}_i; \frac{1}{k-i+1}\sum_{j=i}^{k}\mathbf{h}_j; \mathbf{h}_k\right]. \quad (8)$$

[4]Note that we used the bi-directional representations to take the differences for LSTM-Minus unlike the original one (Wang and Chang, 2016). The investigation of different formulations is left for future work.

$$\mathbf{R}(i,k)^{[L,\overline{A},R]} = [\mathbf{h}_i; \overline{\mathbf{x}}_i; \mathbf{h}_k]\,. \tag{9}$$

$$\mathbf{R}(i,k)^{[\overline{L},A,\overline{R}]} = \left[\mathbf{h}_k - \mathbf{h}_{i-1}; \frac{1}{k-i+1}\sum_{j=i}^{k}\mathbf{h}_j; \mathbf{h}_i - \mathbf{h}_{k+1}\right]. \tag{10}$$

$$\mathbf{R}(i,k)^{[\overline{L},\overline{A},\overline{R}]} = [\mathbf{h}_k - \mathbf{h}_{i-1}; \overline{\mathbf{x}}_i; \mathbf{h}_i - \mathbf{h}_{k+1}]\,. \tag{11}$$

3.2 Concept Indexing

The concept indexing (CI) requires to identify a concept unique identifier (CUI) for every mention span of a concept in a document. SNOMED-CT knowledge-base is used to extract all candidates CUI and its term names. For CI, the input is all predicted mention span $M = \{m_1, m_2, \ldots, m_n\}$, where m_i denotes the i-th mention and n denotes the total number of predicted mentions. Each mention is represented as a word sequence $m_i = \{w_1, ..., w_k\}$. Each CUI c is an entry in a knowledge base (KB) (i.e., SNOMED-CT). For the CI task, the list of entity mention $\{m_i\}_{i=1,...,T}$ needs to be mapped to a list of corresponding CUIs $\{c_i\}_{i=1,...,T}$.

Using the SNOMED-CT database, we first conduct dictionary look-up matching for each mention m_i with CUIs' term names to retrieve an optimal CUI. If the CUI is not found for a mention, we then compute a similarity score using the dot-product with entity embeddings that supposedly should capture possible related CUIs and select the maximum score to predict the optimal CUI for a mention.

We use fixed, continuous, task-specific entity embeddings, namely the pre-trained entity embeddings of Spanish SNOMED-CT KB by extracting all CUIs term name using GloVe (Pennington et al., 2014). For the multi-token term name of a CUI, we simply compute the average embeddings.

4 Experimental Settings

We provide empirical evidence on the effectiveness of the pipeline architecture in both NER and concept indexing on the PharmaCoNER[5] task of the BioNLP-OST 2019[6]. The PharmaCoNER corpus with four entity types[7] is randomly split into three subsets: train, development and test sets, which contain 500, 250 and 250 clinical cases, respectively.

Our model is implemented in the Chainer[8] deep learning framework. We employed the official PharmCoNER evaluation script[9] to evaluate our system's performances on both tasks.

4.1 Data Pre-processing

Each text and the corresponding annotation file were processed by several simple rules only for tokenization. [10] After tokenization, each text with mapping annotation files were directly passed to the deep neural approach for mention detection and classification. Note that the offsets were restored to the original offsets in evaluation.

4.2 Hyper-parameters

Word representations We generated task specific word embeddings of Spanish PharmaCoNER corpus by merging the raw text of training, development, and test (including background set) sets using GloVe (Pennington et al., 2014). We set the dimension of word embeddings to 200, the dimension of character embeddings for character encoding to 25, and character embeddings for morphological analysis to 25.

Hidden dimensions The hidden states in the LSTMs had 200 dimensions. Each feed forward neural network consisted of two hidden layers with 150 dimensions.

Learning We chose Adam (Kingma and Ba., 2015) as the optimization algorithm with a mini-batch size of 10. We used the same hyper-parameters in all the experiments; we set the gra-

[5] http://temu.bsc.es/pharmaconer/

[6] https://2019.bionlp-ost.org/

[7] (NORMALIZABLES: mentions of chemicals that can be manually normalized to a unique concept identifier, NO_NORMALIZABLES: mentions of chemicals that could not be normalized manually to a unique concept identifier, PROTEINAS: mentions of proteins, genes, peptides, peptide hormones and antibodies, and UNCLEAR: cases of general substance class mentions of clinical and biomedical relevance)

[8] https://chainer.org/

[9] https://github.com/PlanTL-SANIDAD/PharmaCoNER-Evaluation-Script

[10] Unlike the traditional NER models, our model is independent from traditional 'BIO' tagging scheme, where 'B', 'I', and 'O' stand for 'Begin', 'Inside', and 'Outside' of named entities respectively, so we do not need to assign such tags to the tokens.

	Sub-task 1: NER			Sub-task 2: CI		
Span representation	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Ensemble	86.88	86.65	**86.76**	87.53	73.61	**79.97**
CSR-Attn	87.08	85.61	86.34	88.01	73.25	79.95
CLM-Attn	85.32	83.93	84.62	87.98	72.54	79.52
CSR-Avg	84.53	83.67	84.09	86.81	71.83	78.61
BLM-Attn	77.58	88.48	82.67	88.41	68.54	77.22

Table 1: Performance of NER and CI on the test set

Label	P(%)	R(%)	F(%)	Prediction	Annotation	Correct
NORMALIZABLES	89.64	88.08	88.85	956	973	857
UNCLEAR	92.00	67.65	77.97	25	34	23
PROTEINAS	84.19	83.70	83.95	854	859	719
NO_NORMALIZABLES	99.99	10.01	18.18	1	10	1
Overall (micro)	86.88	86.65	86.76	1,836	1,876	1,600

Table 2: Sub-task 1: Categorical performance on the test set

dient clipping to 5, the dropout rate to 0.0 and the Adam hyper-parameters to the default values (Kingma and Ba., 2015). The model was trained for up to 10 epochs, with early stopping based on the performance on the development set.

5 Results and Discussions

In order to evaluate the performance of NER and concept indexing, we conducted experiments on different sets of span representations, including contextual span representation (CSR) with averaging (CSR-Avg), CSR using attention (CSR-Attn), contextual LSTM-Minus-based span representations (CLM) with averaging (CLM-Avg), CLM using attention (CLM-Attn). Besides for base span representations (BSR), BSR with averaging (BSR-Avg), BSR using attentions (BSR-Attn), base LSTM-Minus-based span representation (BLM) with averaging (BLM-Avg), BLM using attention (BLM-Attn) are also considered. We also report the result of ensemble learning that combines the predictions using different span representations to reduce the variance of predictions and reduce the generalization error.

Table 1 shows the five submitted results of NER and CI in terms of F-score on the test set. The top five span representations are chosen based on development score to submit the results. In this table, it is shown that the ensemble approach using maximum voting of all the approaches is effective to improve the system performance both in NER and CI tasks with achieving 86.67% in terms of F-score on NER. In contrast, the CSR-Attn shows the best performance as an individual span representation on NER with achieving 86.34% in terms of F-score.

In the CI task, the ensemble approach shows the best performance by achieving 79.97% in terms of F-score. CSR-Attn achieved 79.95% in terms of F-score as the best individual span representation. The pipeline approach may not be a perfect solution to solve the concept indexing task, where wrong predictions from the NER module will affect the results in the second step.

Table 2 shows the categorical performances using ensemble learning of NER on the test set. In this table, we also break down the number of predicted and correct mentions among the gold annotations. In this table, it can be observed that for the classes of NORMALIZABLES and PROTEINAS, the model shows high performance because there are a reasonable number of training instances for the classes and the mentions in these two classes appeared in the same documents. In contrast, for the rare classes UNCLEAR and NO_NORMALIZABLES, the performances are low. This may be partly due to their low frequency in the training set, making it hard to learn their representation in the network.

5.1 Ablation Study

We show the performances of different NER models for Sub-tasks 1 and 2 on the development set in Table 3 to compare the possible scenarios of the

	Sub-task 1: NER			Sub-task 2: CI		
Span representation	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Ensemble	90.82	87.80	**89.28**	88.91	68.46	**77.36**
CSR-Attn	91.54	86.50	88.95	88.33	68.19	76.96
CSR-Avg	87.89	85.93	86.90	86.96	67.25	75.84
CLM-Attn	89.87	84.27	86.98	88.73	68.33	77.20
CLM-Avg	86.80	85.36	86.07	87.51	67.85	76.44
BSM-Attn	86.43	85.31	85.86	87.22	68.39	76.67
BSM-Avg	84.99	86.45	85.71	87.33	68.12	76.54
BLM-Attn	87.15	85.20	86.16	88.91	67.38	76.66
BLM-Avg	87.32	84.37	85.82	87.57	67.25	76.07

Table 3: Performance of NER and CI on the development set

Label	P(%)	R(%)	F(%)	Prediction	Annotation	Correct
NORMALIZABLES	91.61	88.58	90.07	1,084	1,121	993
UNCLEAR	91.11	93.18	92.13	45	44	41
PROTEINAS	89.63	86.98	88.28	723	745	648
NO_NORMALIZABLES	90.00	56.25	69.23	10	16	9
Overall (micro)	90.82	87.80	89.28	1,862	1,926	1,691

Table 4: Sub-task 1: Categorical performances on the development set

given solutions and to report the best system submissions for NER and CI. The Sub-tasks 1 and 2 results in Table 3 shows that almost all the results in different approaches are close to each other to solve the Sub-tasks 1 and 2. The top four models (i.e., CSR-Attn, CLM-Attn, CSR-Avg, and BLM-Attn) and the ensemble of eight models are considered for test evaluation. As for the single NER model, the results on Sub-tasks 1 and 2 in Table 3 show that attention performs better than averaging when the other settings are same. LSTM-Minus helps when there is no contextual information, but it does not help when there is contextual information.

In the CI task on development set, the ensemble approach shows the best performance by achieving 77.36% in terms of F-score. CLM-Attn achieved 77.20% in terms of F-score as the best individual span representation.

Table 4 shows the categorical performances using ensemble learning of NER on the development set. In this table, it seems that the model is well generalized to detect the mentions of each classes including rare classes such as UNCLEAR and NO_NORMALIZABLES on development set. The categorical performances of NORMALIZABLES and PROTEINAS in terms of F-score are dropped marginally from development to test scores by 1.22% and 4.33%, respectively. But it is surprising that the categorical performances of the rare classes UNCLEAR and NO_NORMALIZABLES, where the performances in terms of F-score are significantly dropped by 14.16% and 51.05% respectively, that affect the overall F-score of test set. We remain this analysis for our future work.

6 Conclusion

This paper presented a pipeline approach that integrates the contextual that captures the surrounding context of a target span and non-contextual neural exhaustive models, which consider all possible spans exhaustively, for named entity recognition (NER) and dictionary and similarity score-based matching for concept indexing (CI), without depending on any external NLP tools. The proposed contextual exhaustive model is capable to detect flat and nested entities from the generated mention candidates of all possible spans. The model obtains the representation of each span using the outputs of the underlying shared bidirectional LSTM layer, and it represents the different spans by concatenating forward- and backward-context, boundary and inside representations of the span. Several enhancements, namely contextual span representation, average representation,

attention mechanism, LSTM-Minus, and ensembling are investigated for the representations. It then classifies the span into an entity type or non-entity. To predict the concept unique identifier (CUI) of a mention, the system performs dictionary matching and then computes a similarity score for a mention with no matching using entity embeddings. Among the five submitted runs, the best run for each Sub-task achieved the F-score of 86.76% on Sub-task 1 (NER) and the F-scores of 79.97% on Sub-task 2 (CI).

In the future direction, we will implement a joint modeling that directly recognize entity mentions and link them to a concept unique identifier in an end-to-end manner.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

N. Collier, H. S. Park, N. Ogata, Y. Tateisi, C. Nobata, T. Ohta, T. Sekimizu, H. Imai, K. Ibushi, and Jun'ichi Tsujii. 1999. The GENIA Project: Corpus-based Knowledge Acquisition and Information Extraction from Genome Research Papers. In *Proceedings of EACL*, pages 171–172. ACL.

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 ACL (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany.

Pavlina Fragkou. 2017. Applying named entity recognition and co-reference resolution for segmenting english texts. *Progress in Artificial Intelligence*, 6(4):325–346.

Aitor Gonzalez-Agirre, Montserrat Marimon, Ander Intxaurrondo, Obdulia Rabal, Marta Villegas, and Martin Krallinger. 2019. Pharmaconer: Pharmacological substances, compounds and proteins named entity recognition track. In *Proceedings of the BioNLP Open Shared Tasks (BioNLP-OST)*, pages 1–X, Hong Kong, China. Association for Computational Linguistics.

Onur Gungor, Tunga Gungor, and Suzan Uskudarli. 2019. The effect of morphology in named entity recognition with sequence tagging. *Natural Language Engineering*, 25(1):147–169.

Onur Gungor, Suzan Uskudarli, and Tunga Gungor. 2018. Improving named entity recognition by jointly learning to disambiguate morphological tags. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 2082–2092.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A Neural Layered Model for Nested Named Entity Recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana. ACL.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

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 ACL: Human Language Technologies. ACL*, volume 1, pages 260–270, San Diego, California. ACL.

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 1105–1116, Berlin, Germany. ACL.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*.

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

Mohammad Golam Sohrab, Pham Minh Thang, and Makoto Miwa. 2019. A generic neural exhaustive approach for entity recognition and sensitive span detect. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019)*, pages 735–743, Span. IberLEF 2019.

Nguyen Truong Son and Nguyen Le Minh. 2017. Nested Named Entity Recognition Using Multilayer Recurrent Neural Networks. In *Proceedings of PACLING 2017*, pages 16–18, Sedona Hotel, Yangon, Myanmar.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, Berlin, Germany. Association for Computational Linguistics.

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.