ISSN:1738-6764, eISSN:2093-7504

# Encoding Dictionary Feature for Deep Learning-based Named Entity Recognition

**Chirawan Ronran [1], Sayan Unankard [2] and Seungwoo Lee [3,*]**

[1] Major of Big Data Science, University of Science and Technology (UST) and Korea Institute of Science and Technology Information (KISTI), Korea; chirawan@kisti.re.kr
[2] Information Technology Division, Faculty of Science, Maejo University; sayan@gmaejo.mju.ac.th
[3] Major of Big Data Science, University of Science and Technology (UST) and Korea Institute of Science and Technology Information (KISTI), Korea; swlee@kisti.re.kr
* Correspondence

https://doi.org/10.5392/IJoC.2021.17.4.001

Manuscript Received 21 June 2021; Received 06 October 2021; Accepted 02 November 2021

This is an excellent paper selected from the papers presented at ICCC 2020

***Abstract:*** *Named entity recognition (NER) is a crucial task for NLP, which aims to extract information from texts. To build NER systems, deep learning (DL) models are learned with dictionary features by mapping each word in the dataset to dictionary features and generating a unique index. However, this technique might generate noisy labels, which pose significant challenges for the NER task. In this paper, we proposed DL-dictionary features, and evaluated them on two datasets, including the OntoNotes 5.0 dataset and our new infectious disease outbreak dataset named GFID. We used (1) a Bidirectional Long Short-Term Memory (BiLSTM) character and (2) pre-trained embedding to concatenate with (3) our proposed features, named the Convolutional Neural Network (CNN), BiLSTM, and self-attention dictionaries, respectively. The combined features (1-3) were fed through BiLSTM - Conditional Random Field (CRF) to predict named entity classes as outputs. We compared these outputs with other predictions of the BiLSTM character, pre-trained embedding, and dictionary features from previous research, which used the exact matching and partial matching dictionary technique. The findings showed that the model employing our dictionary features outperformed other models that used existing dictionary features. We also computed the F1 score with the GFID dataset to apply this technique to extract medical or healthcare information.*

**Keywords:** BiLSTM Dictionary; CNN Dictionary; Self-attention Dictionary; GFID Dataset; Biomedical NER.

## 1. Introduction

In Natural Language Processing (NLP), a use of deep learning models has allowed Artificial Intelligence (AI) to surpass human levels on several important tasks, for example, Question & Answering (QA), Machine Translation (MT), and Information Extraction (IE). Named Entity Recognition (NER) is a sub-task of IE that extracts entities mentioned in an unstructured text into a category such as organization, person, and location. There are four different types of NER techniques: a rule-based approach that relies on hand-crafted rules, an unsupervised learning approach that is based on an algorithm without any label data, a feature-based supervised learning approach that focuses on a supervised learning algorithm, and DL approach that detects results from raw input [1,2].

Recently, along with the development of DL models, neural network models have been successfully used for the NER task. It has been shown that the injection dictionary as features in a sequence tagger was able to improve the performance in NER (e.g., Ratinov and Roth, 2009, Cohenand Sarawagi, 2004, Chui and Nichols, 2015, and Tianyu et al., 2019) [5-8]. It also provides valuable information for handling unseen cases and rare words because the dictionary feature demonstrates human knowledge [1,9]. Therefore, the previous research commonly concatenated the dictionary with other features before passing it to the model.

We hypothesized that if DL-based dictionary properly classified the named entities, the accuracy of the model could be improved. To investigate the performance of the dictionary on DL, we studied the effect of a

dictionary representation being fed into a model (CNN/BiLSTM/Self-Attention). The model was concatenated with other representations (FastText, ELMo, and Character Embedding). In this experiment, we used a DL model designed by DeLFT [10] using BiLSTM for learning the model and the Conditional Random Field (CRF) for decoding the results. This research was conducted with a purpose to:

1. Learn and evaluate the Bidirectional Long Short-Term Memory (BiLSTM) CRF model for BiLSTM character-level and FastText word representation [11] with/without context embedding (BERT, ELMo) [12,13]. The models were used as baselines for our experiments.

2. Concatenate the BiLSTM character-level, FastText, with/without context embedding (called previous representations) with a dictionary representation. In this task, we divided the dictionary representation into two categories:

2.1. Baseline representation: The previous representations were combined with partial matching [7,14], exact matching [5,15,16], and token matching [2].

2.2. Dictionary feature representation: The previous representations were concatenated with (1) Found - not Found, (2) SoftMax, and (3) Hyperbolic tangent (Tanh) dictionary feature. In this study, the Found- not Found dictionary feature was applied from Ronran et al., 2020 [17], while the SoftMax and Tanh dictionary features are presented as new experiments.

2.3. Then, they were passed through CNN, BiLSTM, and Self-Attention models.

The combinations (BiLSTM character-level, FastText, with/without context-embedding, and dictionary representation) were injected into the BiLSTM CRF, and we verified the performance of each model by using F1-score.

For evaluation, we conducted experiments with the OntoNotes 5.0 dataset, and GFID dataset. The GFID dataset is generated from electronic literature and biomedical news, which has annotation of entities on infectious disease outbreak. In this task, we rely on handcrafted to identify 20 classes including date, disease, duration, died, GPE, infected, location, money, nationality, number, ordinal, organization, outbreak, pathogen, percent, person, probable, suspected, time and title.

The rest of the paper is structured as follows. Related works are reviewed in section 2. We present our model architecture and hyperparameters in section 3. The experimental results are shown in section 4. A discussion is given in section 5. Section 6 gives the conclusion.

## 2. Related Works

### 2.1 Dictionary Features

The dictionary features are used to map an entity of word or token in a dictionary with a word in text and generate the unique index. There are three common matching methods. The first is an exact matching, the second is a partial matching, and the third is a token matching.

2.1.1 Exact matching: this method injects a dictionary as features into a model [5,6] and labels the object with a CRF. Given a dictionary of named entities, a dictionary feature is set to a word or a phrase in the input sentence. The dictionary feature of the first word in the named entity is assigned to Begin tag (B), while the dictionary feature of the remaining words in the named entity is assigned to Inside tag (I). The other words in the input sentence which are not part of any named entity in the dictionary are set to Outside tag (O). If multiple matches are found in an entity, the longest word or phrase is preferred [18]. With the exact matching technique, the correct type of word is selected as much as the n-gram covers the ground truth [1]. However, a longer match requires additional bits to categorize each word type [19].

2.1.2 Partial matching: n-gram is used to find a partial match in text with a dictionary entry. The standard model of this technique was proposed by Chiu and Nichols, 2015 [7].

**Table 1.** Example of how partial matching is applied. B, I, E, and S tags indicate that the token matches an entry in the dictionary

| Entity | John | studies | at | Washington | And | Lee | University |
|---|---|---|---|---|---|---|---|
| LOC | - | - | - | B | I | I | E |
| GPE | - | - | - | S | - | - | - |
| PER | S | - | - | - | - | - | - |
| ORG | - | - | - | - | - | - | - |

To generate dictionary features, the Begin tag (B), Inside tag (I), End tag (E), and Single tag (S) indicate a token of an entry in the dictionary, as shown in Table 1. The overall model's performance could be improved by a partial matching dictionary. In contrast, some researchers forgo the dictionary because it produces many wrong matches [5].

2.1.3 Token matching: Ronran et al., 2020 [2] proposed the token matching strategy. To avoid the problem of tag mismatch, especially for a person that commonly assigned the first name as B, the last name is typically classified with an I or E. However, text is possibly assigned the last name as B, I, E, or S. They omitted the tag position and used 0 and 1 to indicate the presence or absence of a word in the dictionary as illustrated in Table 2.

**Table 2.** Example of how token matching is applied. The 1 indicates that the token is found and 0 indicates that the token is not found in the dictionary.

| Entity | John | studies | at | Washington | And | Lee | University |
|---|---|---|---|---|---|---|---|
| LOC | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| GPE | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| PER | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ORG | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

2.2 Architecture for NER extraction on BiLSTM model

A DL-based NER architecture was introduced by Collobert et al., 2011 [20]. The authors proposed vanilla neural networks over word embedding. The bidirectional model was used and extended by concatenating pre-trained word embedding with the CNN Character-BiLSTM-CRF (Ma and Hovy, 2016, Peters et al., 2018) [13,21], the BiLSTM Character-BiLSTM-CRF (Lample et al., 2016, Liu et al., 2018) [22,23], the CNN Character-BidLSTM-CNN (Chiu & Nichols, 2016) [7], and the BiGRU-CRF (Peters et al., 2017) [24].

The BiLSTM for NER was used to extract information from various unstructured texts. For example, Hamada et al., 2019 utilized pre-trained word representation, character representation, disease-dictionary information, and BiLSTM CRF for disease recognition [25]. Wu et al., 2020 proposed a DL based on BiLSTM-CRF and domain dictionary-matching correction to extract cyber threat information [26]; Cui et al., 2021 and Wu et al.,2021 applied BiLSTM with Attention-CRF to extract the named entities of job-skill and Chinese-clinic, respectively [27,28].

In DeLFT NER (2020), the authors re-implemented several state-of-the-art algorithms for NER, including the usage of ELMo and BERT architecture. DeLFT offered validate-reported results for benchmarking several methods under the same criteria and conditions. From the F1-score obtained by DeLFT, we found that the model of Lample et al., 2016 achieved the best score in this task.

## 3. Materials and Methods

3.1 Features

3.1.1 Embedding Features.

We used three kinds of pretrained embedding: (1) FastText (cc300.vec) is a distributed representation of words in a vector space that captures syntactic and semantic information [11]. (2) ELMo is a contextualized embedding and character-level embedding method. ELMo analyzes entire sentences before assigning embedding for each sentence [13]. (3) BERT is a representation that takes sub-words as inputs and learns embeddings from them, returning sentence embedding [12].

3.1.2 Dictionary Features.

In this section, we discuss how dictionaries employed by our system were constructed. First, we classified the dictionary into 11 entity types for OntoNotes 5.0 and 7 entity types for GFID. Then, we crawled data from Wikipedia, Geoname, and Kaggle to extract all possible vocabularies. All vocabularies were grouped for each entity type. Each vocabulary was searched on the Google search engine, and the number of pages found was counted. Table 3 and Table 4 shows the number of vocabularies from three sources mentioned above found in both datasets respectively.

**Table 3.** The coverage of the dictionary on the OntoNotes5.0 dataset

| Entity Type | Found | Found (%) | Not Found | Not Found (%) |
|---|---|---|---|---|
| GPE | 1,411 | 73.30 | 514 | 26.70 |
| Event | 197 | 33.68 | 388 | 66.32 |
| Law | 146 | 34.11 | 282 | 65.89 |
| Faculty | 246 | 29.71 | 582 | 70.29 |
| Language | 22 | 47.83 | 24 | 52.17 |
| Location | 134 | 21.04 | 503 | 78.96 |
| NORP | 302 | 45.76 | 358 | 54.24 |
| Organization | 3,437 | 72.69 | 1,291 | 27.31 |
| Product | 156 | 32.70 | 321 | 67.30 |
| Person | 3,481 | 49.76 | 3,514 | 50.24 |
| Work of art | 791 | 53.05 | 700 | 46.95 |
| **Total** | **10,323** | **54.91** | **8,477** | **45.09** |

**Table 4.** The coverage of the dictionary on the GFID dataset

| Entity Type | Found | Found (%) | Not Found | Not Found (%) |
|---|---|---|---|---|
| Disease | 375 | 1.82 | 1,086 | 5.26 |
| GPE | 3,815 | 18.47 | 2,078 | 10.06 |
| Location | 236 | 1.14 | 3,041 | 14.73 |
| National | 144 | 0.70 | 83 | 0.40 |
| Organization | 2,363 | 11.44 | 2,140 | 10.36 |
| Pathogen | 271 | 1.31 | 999 | 4.84 |
| Person | 2,071 | 10.03 | 1,948 | 9.43 |
| **Total** | **9,275** | **44.92** | **11,375** | **55.08** |

After that, we assigned a page count into a token using Algorithm 1.

**Algorithm 1: The pseudo-code for finding a page count of each token**

**Input:** Entity types and the page count of each vocabulary from Google Search
**Output:** A page count of each token categorized composing entity types
**Begin**
dic ← dictionary
**for** entity in entity types:
**for** vocabulary in entity:
number ← vocabulary's count of each entity type

```
            tokens ← tokenize(vocabulary)
            for token in tokens:
                if token was found in dic
                    dic[token] ← dic[token] + number
                else:
                    dic[token] ← number
        return (dic)
End
```

A token and page count of each token (from Algorithm 1) is used to find a possible value by using three techniques: Found and Not Found [2,17], (2) Softmax, and (3) Tanh function.

- **Found and Not Found**: It finds the existence of a token in each entity. In equation 1, T is a token, and i is the index of entity type.

$$T_i = \begin{cases} 1 & Page\ count\ of\ Token>0 \\ 0 & Otherwise \end{cases} \quad (1)$$

- **Softmax:** The input values for Softmax can be positive, negative, or zero. To interpret the value as probability, the values should be between 0 and 1. This SoftMax formula is shown in equation 2: [29]

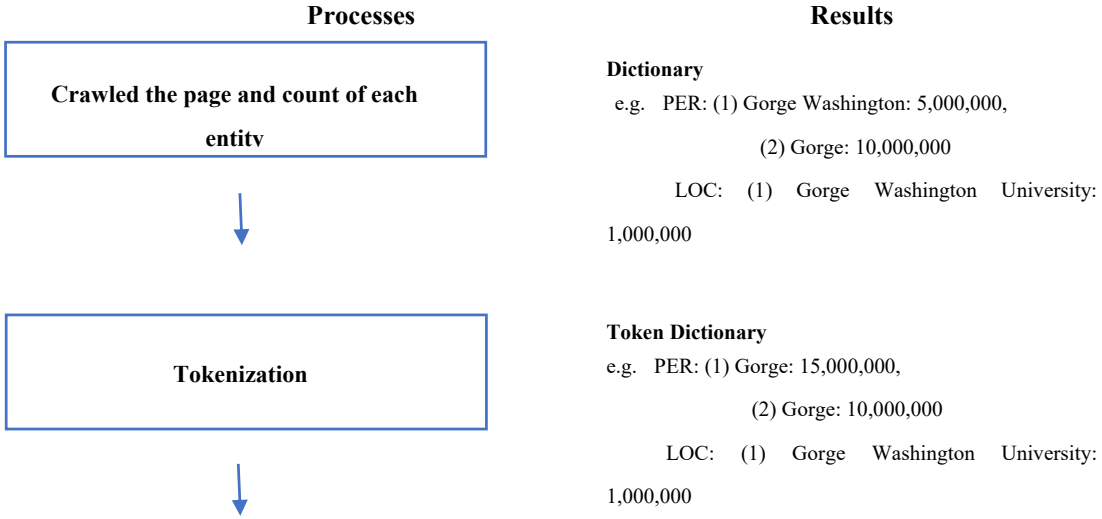
$$\sigma(\vec{t})_i = \frac{e^{t_i}}{\sum_{j=1}^{k} e^{t_j}} \quad (2)$$

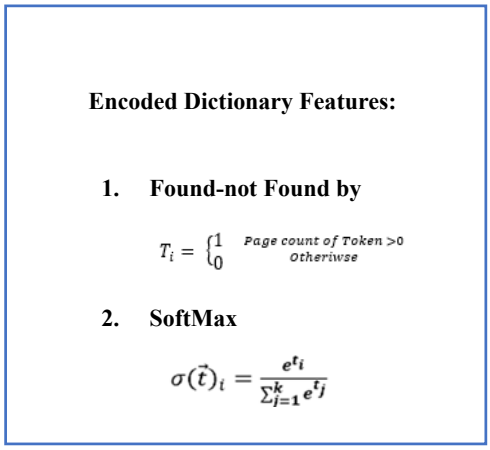
where all the $t_i$ values are the page count of a token in each entity type, K is the number of classes in the multi-class classifier. The term $\sum_{j=1}^{k} e^{t_j}$ is the normalization term to ensures that all the output values of the Softmax function will be equal to 1.

- **Tanh:** Equation 3 uses the Tanh function to generate an output range from -1 to 1. The Tanh advantage is that inputs will be mapped strongly positive and negative inputs nearly 1 and -1 respectively in the Tanh graph. In this experiment, we calculated a Tanh of a token using the following equation.

$$T_i = \text{Tanh}\left(\frac{e_i}{e_{max}}\right) \quad (3)$$

where the $e_i$ value is the page count of an entity token, $e_{max}$ is the maximum value of a page count in an entity-type. Figure 1 summarizes the process of dictionary representation creation.






**Three Dictionary Features**

**1 Found not Found**

| *Vocabulary* | | PER | LOC | |
|---|---|---|---|---|
| Gorge | ( | 1 | 1 | ) |
| Washington | ( | 1 | 1 | ) |
| University | ( | 0 | 1 | ) |

**2 SoftMax**

| *Vocabulary* | | PER | LOC | |
|---|---|---|---|---|
| Gorge | ( | 0.5 | 0.33 | ) |
| Washington | ( | 0.5 | 0.33 | ) |
| University | ( | 0 | 0.33 | ) |

**3 Tanh**

| *Vocabulary* | | PER | LOC | |
|---|---|---|---|---|
| Gorge | ( | 0.67 | 0.33 | ) |
| Washington | ( | 0.67 | 0.33 | ) |
| University | ( | 0 | 1 | ) |

**Figure 1** The process of dictionary representation**.**

The experiments are conducted using three dictionary features with three models as follows:

- **CNN:** CNN is one of the models that is used for the dictionary [2], which attempts to transform a sentence from a sequence of entity dictionaries into a sequence of dense vectors. In the CNN layer, an embedded matrix $E \in R^{vxd}$ is used to map a token in the dictionary into a vector, where v is the vocabulary size, and d is the embedded dimension.

We denoted an input sentence as $s = [w_1, w2, ...,w_N]$, where N is the sentence length and $w_i \in R^V$ is the one-hot vector of the $d_i$ dictionary. The output of this layer is a sequence of vectors $[x_1, x_2, ..., x_N]$, where $x_i = E_{wi} \in R^d$. We used the CNN to capture the local context information for NER. $w \in R^{kd}$ is a filter in the CNN where k is the window, then the contextual representation of the $d_i$ dictionary learned by this filter is: $d_i = f(w^T \times x[i - \frac{k-1}{2}]: [i + \frac{k-1}{2}])$, where $x[i - \frac{k-1}{2}]:[i + \frac{k-1}{2}]$ represents the concatenation of the embeddings in the dictionary from $[i - \frac{k-1}{2}]$ to $[i + \frac{k-1}{2}]$, and f is the Tanh activation function.

In this layer, the embeddings are passed through the CNN. We used embedding size = 25 with three filter lengths and followed by global max pooling to learn contextual dictionary representations. Let m be the filter number. The dictionary representation of the i (denoted as $d_i$) is the combination of the outputs of all filters at this position. The output of the CNN layer is $c = [d_1, d_2, ..., d_N]$, where $d_i \in R^m$ [30].

- **BiLSTM:** As shown in equation 4, BiLSTM is used to capture future and past information for each time step.

$$H = \overrightarrow{LSTM}_1(X) \parallel \overleftarrow{LSTM}_2(X) \quad (4)$$

where

- $\overrightarrow{LSTM}_1$ and $\overleftarrow{LSTM}_2$ denote the cells in forward and backward order.
- $X = [x_1, x2, ...,x_N]$ denote the sequence of vector where N is the sentence length.
- H denotes the resulting feature matrix for BiLSTM, and || denotes row-wise combination.

For BiLSTM we assign dictionary types as an input dimension, where the number of units in the hidden layer is 128 [28].

- **Self-Attention**: It is a process relating various positions in a single sequence intent to compute a representation from the sequence [31-33]. Given the hidden feature H of a sequence, each hidden state is projected to different sub-spaces as (1) the query to communicate other hidden states for a token of each word, (2) the key vector for calculating 'dot similarities' with incoming queries, or (3) the value vector for weighted and conveyed information to the query token. Formally, let $d_c$ be the sub-space dimension and m be the number of attention heads. For each head $i \in \{1..m\}$, the attention weight and context matrix are computed as the following equations (5-6).

$$\alpha^i = \sigma\left(\frac{HW^{qi}(HW^{ki})^T}{\sqrt{d_c}}\right) \tag{5}$$

$$C^i = \alpha H W^{vi} \tag{6}$$

where $W^{qi}, W^{ki}, W^{vi} \in R^{d_h x d_c}$ are three (query, key, and value) projections matrices and $\sigma$ performs the sigmoid function. Each row of the $\alpha^1, \alpha^2, \ldots, \alpha^m \in R^{nxn}$ is the result. It contains a token's attention weights to its context. For each row of $C^1, C^2, \ldots, C^m \in R^{nxd_c}$ is the context vector. We used the default hyperparameter setup from Keras Self-Attention [34] for processing sequential data that considers the context for each timestamp.

### 3.1.3 BiLSTM Character Feature

This character word representation has been introduced to represent morphological and orthographical elements [25]. We followed Ma and Hovy and DeLFT and used a BiLSTM to extract features from 25 dimensions. The BiLSTM Character Feature is combined with the dictionary representation and pre-trained embedding before feeding it into BiLSTM CRF model as introduced in section 3.2.

## 3.2 BiLSTM CRF Model

We concatenated the character BiLSTM with pre-trained word embedding with and without context embedding before passing through the BiLSTM CRF model. This model was used to assess the model performance without the dictionary. After that, we evaluated the performance of the previous research dictionary representation by combining all the features in subsection 3.1 and feeding them into a BiLSTM CRF as shown in Figure 2. The previous dictionary results are compared with the results of our dictionary model that the dictionary information is passed through a deep learning model (CNN/BiLSTM/Self Attention) before using BiLSTM CRF to predict the last results.

The CRF layer is used after the output BiLSTM layer, which denotes H. Given the prediction of the label, the sequence is computed using equation 7 [35].

$$P(Y|X) = \frac{exp(\sum_i (W_{CRF}^{y_i} H_i + b_{CRF}^{(y_{i-1}, y_i)}))}{\sum_{y'} (exp(\sum_i (W_{CRF}^{y_i} H_i + b_{CRF}^{(y'_{i-1}, y'_i)})))} \tag{7}$$

where y' denotes an arbitrary label sequence. $W_{CRF}^{y_i}$ and $b_{CRF}^{(y_{i-1}, y_i)}$ are trainable parameters.

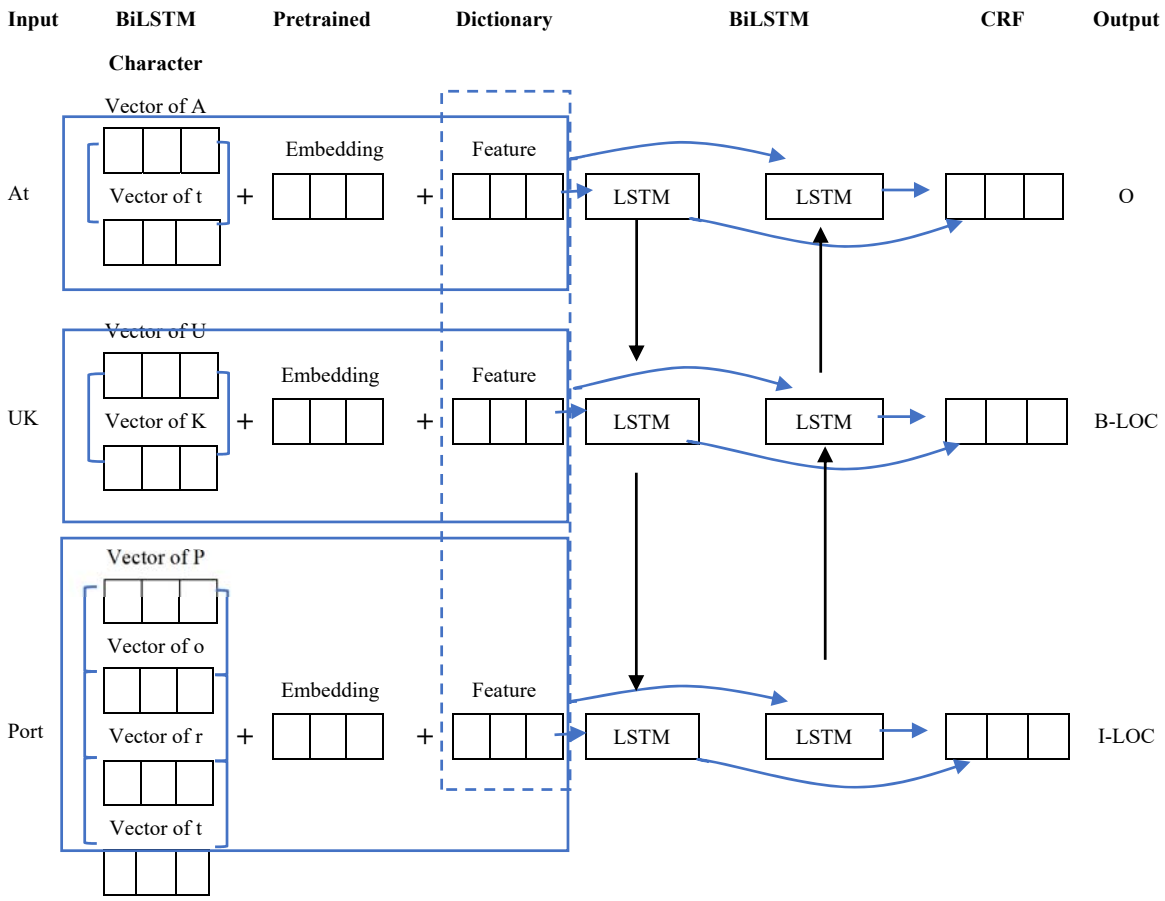



**Figure 2** The model with dictionary

## 3.3 Experimental Setup

### 3.3.1. Dataset

OntoNotes 5.0 dataset consists of a million-token corpus from a variety source of newswires, broadcast news, broadcast conversation, etc. This dataset has seven values (date, time, percent, money, quantity, ordinal, and cardinal) and 11 types of entities: Organization, Work_of_Art, Product, Person, Norp, Location, Law, Language, GPE, Facility, and Event.

In this task, we followed Chiu and Nichols, 2016, that utilized the dataset with the gold standard of annotations [36]. Statistics of this dataset are shown in Table 5.

**Table 5.** The statistics of the OntoNotes5.0 dataset

| | Train | Dev | Test |
|---|---|---|---|
| Sentence | 59,924 | 8,528 | 8,262 |
| Token | 1,088,503 | 147,724 | 152,728 |
| Entity | 81,828 | 11,066 | 11,257 |

GFID dataset, we focused on distinguishing disease and pathogen for extracting cause of sickness in each location. In this task, we build it manually. GFID dataset is for detecting infectious disease outbreak events. So, it has annotation of named entities like disease, GPE, location, nationality, organization, outbreak, pathogen, person, and title; values like date, time number, ordinal, money, duration, and percent; as well as event triggers like probable, suspected, infected, and died - by using Stanford CoreNLP (NER) and dictionary to list of infectious disease and pathogen names. Additionally, the datasets are isolated within train, development, and test set as shown in Table 6.

**Table 6.** The statistics of the GFID dataset

| | Train | Dev | Test |
|---|---|---|---|
| Sentence | 162,664 | 19,631 | 18,234 |
| Token | 2,006,518 | 243,440 | 232,333 |
| Entity | 268,955 | 32,795 | 29,302 |

3.3.2. Hyperparameter Setup

All our experiments used the BiLSTM CRF model given in section 3.2. To avoid bias from hyperparameter tuning, we borrowed the parameters from DeLFT NER as shown in Table 7.

**Table 7.** DeLFT's hyperparameter setup

| **Hyper-parameter** | |
|---|---|
| Char_emb_size | 25 |
| Char_lstm_units | 25 |
| Max_char_length | 30 |
| Char_cnn_filler size | 3 |
| FastText embeddings_name | cc300.vec |
| FastText embedding_dimension | 300 |
| ELMo embeddings_name | 5.5B word corpus |
| ELMo embedding_dimension | 1024 |
| BERT embeddings_name | Cased |
| BERT Base embedding_dimension | 768 |
| BERT Large embedding_dimension | 1024 |
| Hidden unit | 128 |
| Max_sequence_length | 300 |
| Learning_rate | 0.001 |
| Lr_decay | 0.9 |
| Clip_gradients | 5.0 |
| Patience | 5 |
| Optimizer | Adam |
| Epoch | 200 |

For the dictionary, we applied the hyperparameter from the previous model except for emb_size and dic_length. We changed these values to the number of entity types in the dictionary as shown in Table 8.

**Table 8.** Hyperparameter setup of the dictionary model.

| **Hyper-parameter** | |
|---|---|
| Dictionary OntoNotes_emb_size | 11 |
| Dictionary GFID_emb_size | 7 |
| Dictionary _lstm_units | 25 |
| Max_ Dictionary _length | 11 |
| Dictionary _cnn_filler size | 3 |
| Attention_activation | Sigmoid |

In the experiment, we used the last two layers of BERT-BASE and the last one layer of BERT-LARGE without finetuning due to the limitation of our hardware. We also increased the epochs to 200 for all models. The model performances were evaluated with F1- Score. We trained each model to obtain the precision, recall, F1-Score.

## 4. Results

### 4.1. Baseline

We divided our baseline into two parts: (1) the Lample model with and without context embedding (BERT or ELMo), and (2) the Lample model with partial [7,14] and exact matching dictionary representation [5,15,16]

For reference, we executed the baseline method with FastText, ELMo+FastText, BERT+FastText BiLSTM combined with Character BiLSTM before passing through the CRF to predict the last result. F1-scores of the baseline are shown in Table 9.

**Table 9** The results of the baseline model (Lample) with/without contextual embedding (BERT and ELMo)

| Model / Dataset | | Dimension | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| BiLSTM Char + FastText | OntoNotes | 25+300 | 86.150 | 85.296 | 85.722 (±0.198) |
| BiLSTM Char+ ELMo+ Fasttext | OntoNotes | 25+1,024+300 | 88.504 | 88.928 | 88.713 (±0.106) |
| BiLSTM Char +BERT LARGE 1Layer+ Fasttext | OntoNotes | 25+1,024+300 | 87.585 | 88.105 | 87.845 (±0.035) |
| BiLSTM Char + BERT BASE 2 Layers+ Fasttext | OntoNotes | 25+1,536+300 | 88.458 | 88.948 | 88.700 (±0.083) |
| BiLSTM Char + FastText | GFID | 25+300 | 87.557 | 90.070 | 88.793 (±0.191) |

The results show that the re-implemented Lample FastText model from DELFT provided the F1 score at 85.722%, whereas ELMo, BERT-LARGE-cased 1 Layer and BERT-BASE-cased 2 Layers performed better in this task at 88.713%, 88.700%, and 88.793%, respectively. (*We used frozen ELMo and BERT without fine-tuning due to hardware limitation.)

We re-implemented the second baseline using (1) the partial matching dictionary and (2) the exact matching dictionary. Each dictionary was combined with pre-trained embedding and BiLSTM-Character before feeding them into BiLSTM-CRF. Table 10 provides the F1-score of these models. The results of the partial and exact matching dictionary with BiLSTM Character – FastText showed 0.118% and 0.568% improvement respectively.

**Table 10** The results of baseline model (Lample) with partial or exact matching dictionary

| Model | Dataset | Matching Dictionary | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| BiLSTM Char + FastText | OntoNotes | Partial | 86.208 | 85.470 | 85.840 (±0.550) |
| BiLSTM Char + FastText | OntoNotes | Exact | 86.630 | 85.950 | 86.290 (±0.344) |
| BiLSTM Char + FastText | GFID | Partial | 87.463 | 90.100 | 88.727 (±0.087) |
| BiLSTM Char + FastText | GFID | Exact | 87.737 | 90.503 | 88.953 (±0.246) |

We believe that a dictionary with the DL model may influence the performance of the NER. According to this, we further experimented on dictionary models and compared the results with the baseline as discussed in the next sub-section.

### 4.2 Dictionary Model

We conducted experiments by creating a dictionary representation with three different models (CNN, BiLSTM and Self-Attention) and combined them with BiLSTM Character+FastText without ELMo. The F1-Score results are shown in Table 11 and Table 12.

In Table 11 and 12, the results of CNN Found -not Found dictionary are the technique from Ronran et al., 2020. This CNN Dictionary was better than the partial model. The exact matching dictionary seemed to get a higher F1-Score.

We found that the F1-Scores of our model, except the Tanh CNN dictionary, were better than the partial, and the exact match dictionary.

**Table 11** The OntoNotes 5.0's results of our CNN/BiLSTM/Self-Attention dictionary with FastText BiLSTM-CRF compared with Partial and Exact matching (Different =F1-Score – Partial/Exact matching F1-Score of Table 10)

| Dictionary | | Precision | Recall | F1-Score | Different | |
|---|---|---|---|---|---|---|
| | | | | | Partial | Exact |
| CNN | Found-not Found | 86.635 | 85.855 | 86.245*(±0.021) | 0.405 | -0.045 |
| | SoftMax | 86.845 | 86.155 | 86.500 (±0.014) | 0.660 | 0.210 |
| | Tanh | 86.570 | 85.597 | 86.083 (±0.129) | 0.243 | -0.207 |
| BiLSTM | Found-not Found | 86.973 | 86.190 | 86.580 (±0.061) | 0.740 | 0.290 |
| | SoftMax | 86.997 | 86.273 | 86.630 (±0.052) | 0.790 | 0.340 |
| | Tanh | 86.893 | 86.180 | 86.533 (±0.135) | 0.693 | 0.243 |
| Self-Attention | Found-not Found | 86.755 | 86.050 | 86.400 (±0.198) | 0.560 | 0.110 |
| | SoftMax | 86.794 | 86.110 | 86.452 (±0.423) | 0.612 | 0.162 |
| | Tanh | 86.760 | 85.975 | 86.370 (±0.170) | 0.530 | 0.080 |

* Ronran et al., 2020 model

**Table 12** The GFID's results of our CNN/BiLSTM/Self-Attention dictionary with FastText BiLSTM-CRF compared with Partial and Exact matching (Different = F1-Score – Partial/Exact matching F1-Score of Table 10)

| Dictionary | | Precision | Recall | F1-Score | Different | |
|---|---|---|---|---|---|---|
| | | | | | Partial | Exact |
| CNN | Found-not Found | 87.627 | 90.277 | 88.933* (±0.404) | 0.173 | -0.020 |
| | SoftMax | 88.037 | 90.327 | 89.163 (±1.674) | 0.403 | 0.210 |
| | Tanh | 87.827 | 90.457 | 89.120 (±0.157) | 0.360 | 0.167 |
| BiLSTM | Found-not Found | 87.700 | 90.620 | 89.140 (±0.436) | 0.380 | 0.187 |
| | SoftMax | 87.703 | 90.707 | 89.180 (±0.964) | 0.420 | 0.227 |
| | Tanh | 87.667 | 90.557 | 89.087 (±0.379) | 0.327 | 0.134 |
| Self-Attention | Found-not Found | 87.740 | 90.540 | 89.120 (±0.436) | 0.393 | 0.167 |
| | SoftMax | 87.530 | 90.300 | 89.043 (±2.650) | 0.317 | 0.090 |
| | Tanh | 87.847 | 90.663 | 89.233 (±1.464) | 0.473 | 0.280 |

* Ronran et al., 2020 model

The results also showed that the performance of the Self-Attention dictionary was marginally improved over those of the Found-not Found CNN dictionary. The Tanh Self-Attention dictionary achieved the best score at 89.233% on GFID dataset. This F1-Score significantly improves 0.473% point when compared with partial matching and 0.280% point when compared with exact matching. The SoftMax BiLSTM dictionary provides the highest score at 86.630% on OntoNotes 5.0 dataset which also significantly improves 0.790% point and 0.340% point when compared with partial and exact matching, respectively (Wilcoxon rank sum test, $p < 0.05$).

Next, we further conducted experiments on the combination of BiLSTM dictionary+ ELMo+ FastText to improve our model accuracy on OntoNotes dataset. The results are shown in Table 13.

**Table 13** The OntoNotes 5.0's results of our BiLSTM dictionary with ELMo+ FastText BiLSTM-CRF

| Dictionary | Precision | Recall | F1-Score | Different | |
|---|---|---|---|---|---|
| | | | | Partial | Exact |
| Partial | 88.564 | 88.822 | 88.694 (±0.215) | - | -0.078 |
| Exact | 88.504 | 89.048 | 88.772 (±0.119) | 0.078 | - |

| Dictionary | | Precision | Recall | F1-Score | Different | |
|---|---|---|---|---|---|---|
| | | | | | Partial | Exact |
| BiLSTM | Found-not Found | 88.820 | 89.185 | 89.005 (±0.394) | 0.311 | 0.233 |
| | SoftMax | 88.820 | 89.200 | 89.013 (±0.130) | 0.319 | 0.241 |
| | Tanh | 88.787 | 89.137 | 88.963 (±0.076) | 0.269 | 0.191 |

In Table 13, the BiLSTM dictionary with SoftMax achieved the best F1-score at 89.013%. The Found-not Found is the second best and the Tanh gets the lowest score in these experiments. These F1-scores are higher than the partial and exact matching dictionary. We then used these results to analyze the performance of the models in the next section.

## 5. Discussion

Our experiments are performed against baseline approaches. The results show that our approach is effective in the NER task. Table 11 shows our experimental results on the OntoNotes 5.0 dataset. The results show that using our dictionary representation approach can improve the F1-Score of BiLSTM Character+FastText with exact matching from 86.290% (Table 10) to 86.630% (SoftMax). In addition, Table 12 shows the experiment results on the GFID dataset. The results show that using our dictionary representation approach can improve the F1-Score of BiLSTM Character+FastText with exact matching from 88.953% (Table 10) to 89.233% (Tanh). We estimated that the difference in the dictionary performances was caused by (1) the DL models, and (2) the functions to create the dictionary representation.

Moreover, the CNN dictionary was designed for one-word sentences [2] that are not commonly found in the OntoNotes 5.0 and GFID dataset. Therefore, the CNN result was less effective than the other models. For the Self-Attention dictionary, we followed Keras Self-Attention [32], which has the BiLSTM cells in front of the Self-Attention layer. This layer uses the decoder to consider aspect-term information from the whole context. By using the Self-Attention mechanism, we achieved a better score than the CNN mechanism.

However, the Self-Attention achieved the score lower than the BiLSTM on OntoNotes 5.0 dataset. A possible reason is that the BiLSTM learns information only from both forward and backward directions. It might be easier to understand the structure of an entity that is in sequence [35]. Because of this, the BiLSTM obtained the highest F1-Score on OntoNotes 5.0 dataset. While the Self-Attention model may capture an important part of a sentence [37], it is useful to classify the boundary of disease and pathogen name which has the same pattern, such as ‘Hepatitis A virus, and ‘Hepatitis B virus’. Especially, the Self-Attention model is computed with Tanh function that is used to interpret the probability from an entity-type.

The Found - not Found function is computed from a type of entity in the dictionary as same as the Tanh function. The Tanh function also gives a better score than the SoftMax function when encoding with Self-Attention models on the GFID dataset. The SoftMax function calculates the probability of token values from all entities. With this SoftMax computation, the results in Tables 11 and 12 show that SoftMax BiLSTM provides the best and second-best result of experiments on the OntoNotes 5.0 and GFID dataset.

We further conducted experiments on the combination of BiLSTM dictionary+ ELMo+FastText to improve our performance on the OntoNotes 5.0 dataset. The results show that ELMo+FastText can improve the F1-Score of BiLSTM Character+FastText with exact matching from 86.290% (Table 10) to 88.772%. Then, the F1-Score is increased to 89.013% when using our SoftMax dictionary feature on the model. Therefore, the experimental results indicated that our dictionary approach can improve the performance of the NER task.

**Table 14.** The F1-score of Liu et al. models on OntoNotes 5.0 dataset

| Model | F1-Score |
|---|---|
| HSCRF | 89.38±0.11 |
| HSCRF+ softdict | 89.94±0.16 |

We also compared our model with the model of Liu et al., 2019 [38]. The results are shown in Table 14. The performance of our models is less than Liu et al models. A possible reason is that the technique of HSCRF generally outperforms conventional CRF [39]. The F1-score of Liu's HSCRF model without a dictionary is better than our model. Furthermore, HSCRF with softdict (the concatenation between gazetteer dictionary vector and BILOU scheme token-level label) achieves the improvement at 0.560%. Even the number of vocabularies in our dictionary is less than the gazetteer around 300,000 words, our model performance improves 0.908% and 0.300% when using SoftMax BiLSTM dictionary+FastText and SoftMax BiLSTM dictionary+FastText+ELMO, respectively. We believe our model with ELMo is able to improve by fine-tuning. Furthermore, the token-level label could increase model performance, so we plan to combine our model with Liu's techniques as our future work.

## 6. Conclusions

Many researchers have examined the effect of dictionaries on the NER model and have shown that the large scale of coverage of dictionaries was significant in improving the accuracy of the model. This was because the dictionary provided a kind of knowledge base for entity recognition. However, they did not demonstrate how to integrate a possible dictionary representation with DL rather than just adding another input feature to achieve better performance in NER. In this paper, we introduced three types of dictionary representation (Found-Not Found, Tanh, and SoftMax) and integrated them with three DL models consisting of (1) BiLSTM (2) Self-Attention, and (3) CNN.

We compared the proposed models with the previous dictionary representations. We combined these representations with (1) pretrained embedding and (2) BiLSTM character and passed them through the BLSTM-CRF model. Our extensive evaluation confirmed that our DL dictionary feature gives a better result than concatenating the representation directly to other embeddings.

The best-performing system used the SoftMax-BiLSTM dictionary and FastText + ELMo on OntoNotes 5.0 dataset and Tanh Self-Attention dictionary and FastText on GFID dataset. It achieved an F1-score of 89.013% and 89.233%, respectively. This performance of our model significantly improves that of the partial matching dictionary and the exact matching dictionary + FastText + ELMo by using the model with the SoftMax BiLSTM dictionary (Wilcoxon rank sum test, $p < 0.05$). We do not claim that our model outperforms the OntoNotes 5.0 state-of-the-art result. We are aware that our research may have two limitations: first, the insufficient number of vocabularies in dictionary. Second, the limitation of our hardware that cannot support large context models such as multilayer of BERT LARGE, GPT, ELETRA, RoBERTA, etc.

We intend to extend this model by using HSCRF and concatenating with (1) BILOU scheme token-level, and (2) fine-tuning pre-trained context embedding for our GFID biomedical dataset in the future.

**Acknowledgements:** This research was supported by Government-wide R&D Fund project for infectious disease research (GFID), Republic of Korea (grant number: HG18C0093).

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

[1] J.Li, A. Sun, J. Han, and C. Li, "A Survey on Deep Learning for Named Entity Recognition." IEEE Transactions on Knowledge and Data Engineering, pp. 1-1, 2020, doi: 10.1109/tkde.2020.2981314.

[2] C. Ronran, S. Lee, and H. J. Jang, "Delayed Combination of Feature Embedding in Bidirectional LSTM CRF for NER," Applied Sciences, vol. 10, no. 21, p. 7557, 2020, doi: 10.3390/app10217557.

[3] J. Shang, L. Liu, X. Gu, X. Ren, T. Ren, and J. Han, "Learning Named Entity Tagger using Domain-Specific Dictionary," Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, doi: 10.18653/v1/d18-1230.

[4] W. W. Cohen and S. Sarawagi, "Exploiting dictionaries in named entity extraction," Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD, 2004, doi: 10.1145/1014052.1014065.

[5] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition." Proceedings of the Thirteenth Conference on Computational Natural Language Learning CoNLL 2009, doi: 10.3115/1596374.1596399.

[6] S. Sarawagi and W. W. Cohen, “Semi-Markov conditional random fields for information extraction,” in Proc. Neural Inform. Process. Syst., pp. 1185–1192, 2004, doi: 10.1145/3447241.

[7] J. P. C. Chiu and E. Nichols, “Named entity recognition with bidirectional LSTM-CNNs,” 2015, arXiv:1511.08308. [Online]. Available: http://arxiv.org/abs/1511.08308, doi: 10.1162/tacl_a_00104.

[8] T. Liu, J.-G. Yao, and C.-Y. Lin, “Towards Improving Neural Named Entity Recognition with Gazetteers,” Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, doi: 10.18653/v1/p19-1524.

[9] Q. Wang, Y. Zhou, T. Ruan, D. Gao, Y. Xia, and P. He, “Incorporating dictionaries into deep neural networks for the chinese clinical named entity recognition,” J. Biomed. Inform., vol. 92, 2019, doi: 10.1016/j.jbi.2019.103133.

[10] “kermitt2/delft: a Deep Learning Framework for Text - GitHub,” https://github.com/kermitt2/delft accessed: Mar. 10, 2021.

[11] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” Trans. Assoc. Comput. Linguistics, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

[12] Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” arXiv: 1810.04805, 2018.

[13] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol., vol. 1, pp. 2227–2237, 2018, doi: 10.18653/V1/N18-1202.

[14] F. Dugas and E. Nichols, “Applying BLSTM-CNNs and extended lexicons to named entity recognition in tweets,” in Proc. W-NUT 2016, Osaka, Japan, pp. 178-187, 2016.

[15] A. Neelakantan and M. Collins. “Learning dictionaries for named entity recognition using minimal supervision,” Computer Science, Gothenburg, Sweden, pp.452-461, 2015, doi: 10.3115/v1/E14-1048.

[16] “spaCy · Industrial-strength Natural Language Processing in Python,” https://spacy.io/, accessed: Mar. 10, 2021.

[17] C. Ronran, K.Sarda, and S. Lee. “Lexical Feature Representation for Named Entity Recognition,” *ICCC* 2020, pp.375-376, 2020.

[18] Z. Jie and W. Lu, ‘‘Dependency-guided LSTM-CRF for named entity recognition,’’ in Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), pp. 1–3, 2019, doi: 10.18653/v1/D19-1399.

[19] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. C ernocky, “Strategies for training large scale neural network language models,” *in Proc*. 2011 IEEE Workshop Automatic Speech Recognition and Understanding, pp. 196–201, 2011, doi: 10.1109/ASRU.2011.6163930

[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. J. Mach. Learn. Res., 12:2493–2537, Nov. 2011.

[21] X. Ma and E. Hovy, “End-to-end sequence labeling via bi-directional LSTM_CNNs-CRF,” in Proc. 54th Annu. Meet. Assoc. Comput. Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1064–1074, doi 10.18653/v1/p16-1101

[22] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, “Neural architectures for named entity recognition,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol., San Diego, CA, USA, May 2016, pp. 260–270, doi: 10.18653/v1/N16-1030

[23] L. Liu, “Empower sequence labeling with task-aware neural language model,” in Proc. Assoc. Adv. Artif. Intell., pp. 5253–5260, 2018.

[24] M. Peters, W. Ammar, C. Bhagavatula, and R. Power, ‘‘Semi-supervised sequence tagging with bidirectional language models,’’ in Proc. 55th Annu. Meeting Assoc. Comput. Linguistics, vol. 1, 2017, pp. 1756–1765.

[25] H.A. Nayel and H. L. Shashirekha, "Integrating Dictionary Feature into A Deep Learning Model for Disease Named Entity Recognition," CoRR, abs/1911.01600, 2019.

[26] H. Wu, X. Li, and Y. Gao, "An Effective Approach of Named Entity Recognition for Cyber Threat Intelligence," *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2020, pp. 1370-1374, doi: 10.1109/ITNEC48623.2020.9085102.

[27] X. Cui., F. Dai, C. Sun., Z. Cheng, B. Li, B. Li,.Y.Zhang, Z. Ji, and D. Liu, “BiLSTM-Attention-CRF model for entity extraction in internet recruitment data,” Procedia Computer Science, vol. 183, pp.706-712, 2021, doi 10.1016/j.procs.2021.02.118.

[28] G. Wu, G. Tang, Z. Wang, Z. Zhang, and Z. Wang, "An Attention-Based BiLSTM-CRF Model for Chinese Clinic Named Entity Recognition," in *IEEE Access*, vol. 7, pp. 113942-113949, 2019, doi: 10.1109/ACCESS.2019.2935223.

[29] “Softmax Function Definition | DeepAI,” https://deepai.org/machine-learning-glossary-and-terms/softmax-layer accessed: Mar. 10, 2021.

[30] F. Wu, J. Liu, C. Wu, Y. Huang, and X. Xie, ‘‘Neural Chinese named entity recognition via CNN-LSTM-CRF and joint training with word segmentation,’’ *in Proc. World Wide Web Conf. (WWW),* May 2019, pp. 3342–3348, doi: 10.1145/3308558.3313743

[31] P. H. Li, T. J. Fu, and W. Y. Ma, “Why Attention? Analyze BiLSTM Deficiency and Its Remedies in the Case of Ner,” *in Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, doi 10.1609/aaai.v34i05.6338

[32] A. Vaswani, N. Shazeer, N. Parmar, and J. Uszkoreit, “Attention is all you need,” arXiv Preprint, arXiv:1706.03762, 2017.

[33] Y. Zhu and G. Wang, ‘‘CAN-NER: Convolutional attention network for Chinese named entity recognition,’’ in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol., Minneapolis, MN, USA, vol. 1, pp. 3384–3393, June, 2019.

[34] CyberZHG, “keras-self-attention: Attention mechanism for processing sequential data that considers the context for each timestamp,” PyPI. https://pypi.org/project/keras-self-attention accessed Mar. 12, 2021.

[35] H. Chen, Z. Lin, G. Ding, J. Lou, Y. Zhang, and B. Karlsson, ‘‘GRN: Gated relation network to enhance convolutional neural network for named entity recognition,’’ in Proc. AAAI Conf. Artif. Intell., (AAAI), New York, NY, USA, pp. 6236–6243, July, 2019.

[36] J.-H. Kim, H.-C. Kim, and Y.-S. Choi, “Feature Generation of Dictionary for Named-Entity Recognition based on Machine Learning.” Journal of Information Management, vol. 41, no. 2, pp. 31-46, 2010, doi: 10.1633/jim.2010.41.2.031.

[37] J. Xie, B. Chen, X. Gu, F. Liang, and X. Xu, Self-attention-based BiLSTM model for short text fine-grained sentiment classification, IEEE Access, vol. 7, pp. 180558–180570, 2019, doi: 10.1109/ACCESS.2019.2957510.

[38] T. Liu, J. Yao, and C. Lin, “Towards improving neural named entity recognition with gazetteers,” *in ACL*, pp. 5301–5307, 2019.

[39] Z.-X. Ye and Z.-H. Ling, “Hybrid semi-markov crf for neural sequence labeling,” *in ACL*, pp. 235–240, 2018.