Article

Context-Aware Bidirectional Neural Model for Sindhi Named Entity Recognition

Wazir Ali [1], Jay Kumar [1], Zenglin Xu [1,2,*], Rajesh Kumar [1] and Yazhou Ren [1]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu 611713, China; aliwazirjam@gmail.com (W.A.); jay@std.uestc.edu.cn (J.K.); rajakumarlohano@gmail.com (R.K.); yazhou.ren@uestc.edu.cn (Y.R.)
[2] Harbin Institute of Technology, Shenzhen 510085, China
* Correspondence: zenglin@gmail.com

Abstract: Named entity recognition (NER) is a fundamental task in many natural language processing (NLP) applications, such as text summarization and semantic information retrieval. Recently, deep neural networks (NNs) with the attention mechanism yield excellent performance in NER by taking advantage of character-level and word-level representation learning. In this paper, we propose a deep context-aware bidirectional long short-term memory (CaBiLSTM) model for the Sindhi NER task. The model relies upon contextual representation learning (CRL), bidirectional encoder, self-attention, and sequential conditional random field (CRF). The CaBiLSTM model incorporates task-oriented CRL based on joint character-level and word-level representations. It takes character-level input to learn the character representations. Afterwards, the character representations are transformed into word features, and the bidirectional encoder learns the word representations. The output of the final encoder is fed into the self-attention through a hidden layer before decoding. Finally, we employ the CRF for the prediction of label sequences. The baselines and the proposed CaBiLSTM model are compared by exploiting pretrained Sindhi GloVe (SdGloVe), Sindhi fastText (SdfastText), task-oriented, and CRL-based word representations on the recently proposed SiNER dataset. Our proposed CaBiLSTM model achieved a high F1-score of 91.25% on the SiNER dataset with CRL without relying on additional handmade features, such as hand-crafted rules, gazetteers, or dictionaries.

Keywords: Sindhi named entity recognition; recurrent neural networks; CaBiLSTM; self-attention mechanism; contextual representation learning

check for updates

Citation: Ali, W.; Kumar, J.; Xu, Z.; Kumar, R.; Ren, Y. Context-Aware Bidirectional Neural Model for Sindhi Named Entity Recognition. *Appl. Sci.* **2021**, *11*, 9038. https://doi.org/10.3390/app11199038

Academic Editor: Javier Hernando

Received: 22 August 2021
Accepted: 9 September 2021
Published: 28 September 2021

1. Introduction

The Named Entity Recognition system recognizes named entities (NEs) and classifies them into predefined categories, such as a person, location, organization, and time [1]. It is used as the first step in question answering [2], information retrieval [1], text summarization [3], machine translation [4], and more [5]. A series of neural NER models have been proposed over the past decade for English [6–9], Chinese [10–12], Japanese [13], Urdu [4,14], and multilingual systems [6,15], which have achieved state-of-the-art performance. The NER task in Asian languages [16] has recently attracted many researchers due to its importance and widespread NLP applications. Still, Sindhi has not been studied well mainly because of the unavailability of labeled datasets [17,18].

Sindhi has a complex morphological and syntactical structure [19] that originates from Arabic, Hindi, Sanskrit, and Persian [18,20]. Context sensitivity in writing and comprehension are the its main characteristics [21]. Unlike other languages, Sindhi has spelling variations, ambiguities in suffixes, and different writing styles (inclusion and exclusion of space), which further contribute to increasing the difficulty in language processing and the NER [17] task. The NER in the English language has significantly benefited from its capitalization rule, part-of-speech tagging, and availability of other language resources [4]. On the contrary, Sindhi has no capitalization rule, which makes difference between plain text and NEs [17]. Conversely, digital processing also suffers from the scarcity of part-of-speech tagger and gazetteer resources [18]. Additionally, morphological richness in Sindhi leads to ambiguities, which make the NER task more challenging as compared to western languages [7]. We summarize the challenges related to Sindhi NER and characteristics in Table 1. Some of these ambiguities may also be found in other Asian languages, such as Urdu [4].

Table 1. A list of the ambiguities related to Sindhi NER, along with examples.

Ambiguity	**Description**
Lack of capitalization	The capitalization rule in the English language is an important characteristic to enhance the performance of the NER system. However, there is no difference between NEs and plain text in Sindhi language. Thus, NER becomes challenging task.
Context sensitivity	1. Due to change in the context, a token in a sentence that refers to an NE may change its type or may not remain an NE. Such as the name of a girl سنڌو (pronounced as *Sindhu*) is labeled as PERSON, whereas it also refers to the name of the river labeled as LOC (location) in the SiNER dataset. However, as a verb (pronounced as *Sandhou*), which means 'partition', it does not belong to NE. 2. Ambiguities in the person names are also common. An example of a person's name (*Wazeer*-وزير) is also tagged as the TITLE of a person because it also means minister. The name of a girl (*Suhnhi*- سهڻي) is also an adjective, which means beautiful. 3. The country name Syria (*Shaam*- شام) tagged as GPE (geopolitical entity) is also a name of person (*Shaam*- شام), and it also means evening.
Free word order	The sentence is normally written in subject-object-verb order, but all writers do not follow the same writing order. For example, the sentence "*Wazeer drives a car*" is translated as وزيرڪار هلائي ٿو (transliteration is *Wazeer car halai tho*), as well as ڪاروزير هلائي ٿو (transliteration is *Car Wazeer halai tho*). Such change in sentence order makes the identification of NEs more difficult.
Agglutinative and inflectional morphology	An NE may either change its type or even may not remain a name when connected with other words, which makes Sindhi NER a challenging task. For example, a person's name (مختيار -*Mukhtiyar*) does not remain an NE when combined with '*khud*' forms (خود مختيار *khud-mukhtiyar*), which means 'autonomous'.

More recently, NN-based models have attained a state-of-the-art status in NER [6,15,22,23] and other sequence tagging tasks [8,24]. These models primarily rely on the feature representation at character-level [10,11], word-level [25,26], and joint character-level and word-level [9,13] to encode a word boundary and its context. Moreover, task-specific neural representation learning [27,28] has been widely used to boost the performance of NER. However, the limitation of the NN-based encoder-decoder models is their fixed-length connection [29] of the intermediate semantic vector because the encoder compresses the sequence of information into a fixed-length vector.

The proposed CaBiLSTM model has the ability to tackle the problems of context sensitivity, spelling variations, and ambiguities related to Sindhi NER. Firstly, we pretrain character-level and word-level SdGloVe and SdfastText on the recently proposed unlabeled corpus [30] and use task-specific representations. Then, we propose a CaBiLSTM model by incorporating task-oriented CRL based on joint character-level and word-level representations. The proposed model takes character-level input to learn the character representations. Then, the character representations are transformed into contextual word features. The output of the final bidirectional long short-term memory (BiLSTM) encoder is fed to self-attention [31] through a hidden layer before decoding [11] to tackle the limitation of the encoder-decoder. Finally, we employ the CRF decoder for the prediction of the label sequence.

We compare the baselines and the proposed CaBiLSTM model by exploiting external SdGloVe, SdfastText, task-specific, and CRL-based representations. The synopsis of our novel contributions is provided below:

- Our main contribution is the design of a neural model for Sindhi NER by taking into account relations between the entity pairs. The proposed CaBiLSTM model learns the token-level structure of sentences to capture the dependency of a whole sentence. We combine self-attention into the BiLSTM encoder to deeply capture semantic information and lexical features to boost the model's performance. The CaBiLSTM model relies on the contextual representations that include task-specific NE-based knowledge.
- To alleviate the low-resource problem for training neural models, we train word-level and character-level SdGloVe and SdfastText representations on the unlabeled corpus. To highlight the significance of the proposed model, we compare the neural baseline models with the proposed CaBiLSTM model while exploiting pretrained SdGloVe, SdfastText, and task-specific character-level and word-level representations.
- We investigate the performance gap between classical (pretrained) and task-specific contextual word representations in RNN variants of long short-term memory (LSTM), BiLSTM, and BiLSTM-CRF to predict and classify NEs in the SiNER dataset. Furthermore, we analyze the influence of dropout on the recurrent dimensions to mitigate the overfitting problem and evaluate the context window size (CWS) for selecting optimal hyperparameters. We attain new state-of-the-art results and outperform existing systems on the publicly available SiNER dataset.

The remaining sections of the paper are organized as follows. The previous work on the Sindhi NER task and a related variety of recent neural NER models are discussed in Section 2. A detailed description of the proposed methodology is given in Section 3. The experimental setup to evaluate different approaches is illustrated in Section 4, while Section 5 discuses the results and analysis. Lastly, Section 6 provides the summary of this research.

2. Related Work

In this section, we address the state-of-the-art models related to Sindhi NER techniques and deep learning-based approaches in NER.

2.1. Sindhi Named Entity Recognition

The approaches utilized for Sindhi can be classified into rule-based and machine learning-based. The rule-based methods have primary deficiencies, such as they do not have the potency and manageability [32]. Firstly, they require a high development cost and the continuous maintenance of rules when new NEs are added to the language. Secondly, one must have fluency and expert skills in the target language for rules generation. Additionally, the rules are generic and cannot be applied to other languages. Thirdly, the generation of rules is practically more time-consuming. A pioneer rule-based Sindhi NER model was proposed by Hakro et al. [33] with a reported accuracy of 97%. Their dataset consists of 200k tokens and ten entity types. However, their work lacks the open-source implementation for further verification and extension. Afterwards, Nawaz et al. [34] proposed a rule-based method by using an indexing approach to deal with the contextual ambiguities related to the Sindhi NER. However, their work lacks experimental results, which signify the usage of rules to deal with ambiguities for developing an automatic NER system. Another rule-based system [2] addresses the Sindhi NER problem, which was evaluated on a small number of 936 tokens only, with an accuracy of 98.71%. Their approach is also language-dependent and tested on a small number of tokens. Moreover, NER is a sequence labeling problem usually evaluated through classification metrics, such as precision, recall, and F1-score [35], but Nawaz et al. [34] and Jumani et al. [2] used accuracy metrics for assessing their rule-based models.

2.2. Neural NER Approaches

Language-independent NER models like ours have been proposed in the past without any language-specific setting. Lample et al. [9] introduced neural architectures mainly based on the BiLSTM network that relied on character-level, word-level representations and reported the best results in four languages. Kuru et al. [36] proposed a character-based NER model using the BiLSTM network. It is empirically shown that the character-level representation learning is superior compared to word-level representations as the basic input features. Misawa et al. [13] used a convolutional neural network (CNN) for character-level representation learning, BiLSTM encoder, and CRF for tag prediction. Empirical results show that their model outperformed state-of-the-art neural models in the Japanese NER. Das et al. [37] used word representations to tackle the NER problem in the Bengali language, with the hypothesis that word-level representations are useful because they belong to the same NE category, such as the name of an organization in the vector space of embedded words. Their proposed technique outperformed standard baseline approaches that use cluster labels of word representations and gazetteers constructed from Wikipedia.Huang et al. [12] integrated self-attention in the BiLSTM-CRF to address the problem of Chinese NER using character-level and word-level representation learning. Their results show that the model yields an increased F1-score on two different datasets. Moreover, Jia and Ma [11] also introduced character-level representation learning for the Chinese NER task. Their model is mainly based on a self-attention mechanism that dynamically decides how much information to use from character-level or glyph-level components. The empirical results on two datasets show that their proposed model performs better than existing state-of-the-art Chinese NER systems. Our work also relates to language-independent NER approaches [11,13,38]. A hybrid NER approach proposed by [39] integrates the Support Vector Machine (SVM), dictionaries of Persian NEs, and grammar rules for the Persian language. Their hybrid model yields comparable performance with existing state-of-the-art NER models in other languages. More recently, Yamada et al. [6] proposed a neural model by incorporating the word-level and entity-level contextualized representations, entity-aware self-attention, and bidirectional transformer, which obtain state-of-the-art results. Luo et al. [40] proposed a BiLSTM-based model by incorporating contextual representations and label embedding attention. Their empirical results show the state-of-the-art performance on three benchmark NER datasets. A BiLSTM-based model proposed by [15] tackles the problem of boundary tag sparsity and surpasses the existing state-of-the-art NER systems. These models extract and classify NEs, such as person names, location, organization, date, and others, from the labeled datasets. We propose a CaBiLSTM neural model for NE extraction and classification by considering relations between the entity pairs. Our proposed model relies on joint character-level and word-level contextual representations, which contain task-specific knowledge or the knowledge-base of NEs, a BiLSTM encoder, self-attention, and a CRF decoder.

3. Methodology

This section describes neural baseline models, the pretraining of classical SdGloVe, SdfastText, and task-specific character-level and word-level representations, CRL, and how our CaBiLSTM model is designed for the Sindhi NER task.

3.1. Representation Learning

The representation learning aims to capture the useful semantic, syntactic, and morphological information from raw text [25,26] or labeled text [27] for multiple NLP tasks [3], including NER [13,41]. Generally, characters or words are converted into vector representations of real numbers in the embedding process. Such representations capture the syntax and semantics of the given vocabulary [30], enabling us to perform various mathematical operations. This section illustrates pretrained uncontextualized representation learning algorithms of SdfastText, SdGloVe, and task-specific contextual representations.

3.1.1. Uncontextualized Representation Learning

GloVe [25] and fastText [26] are examples of pretrained uncontextual word representation models, which encode each word from the given input in a fixed vocabulary as a vector that captures semantic information. Typically, such representations are trained over a large corpus, such as Common Crawl or Wikipedia corpus [28]. We utilize an open-sourced recently developed large Sindhi text corpus [30] for unsupervised uncontextual representation learning using GloVe and fastText models. We jointly pretrain SdfastText https://github.com/facebookresearch/fastText/#building-fasttext-for-python (accessed on 17 July 2020) and SdGloVe https://github.com/minimaxir/char-embeddings/blob/master/create_embeddings.py (accessed on 17 July 2020) using open-source implementations.

1. GloVe: The well-known GloVe [25] model was developed at Stanford. The word representation can be derived by factorizing the log of the co-occurrence matrix by minimizing the cosine distance between words to ensure a high co-occurrence probability [42]. The resulting word representations show the linear substructure of the words in vector space.
2. fastText: The fastText [26] model was open-sourced by Facebook. This sub-word model, based on the bag of n-gram characters, is dominant over the skip-gram model [43]. The vector representation is obtained by taking the sum of the vectors of the n-grams appearing in the word [28]. The underlying principle behind this method relies on the information encoding in sub-word representations [30].

3.1.2. Task-Specific Contextual Representation Learning

The limitation of the pretrained word embedding model is the representation of a word in a single embedding without encoding its context [23].To tackle this drawback, two different word representation techniques have been developed: one is pattern-based word-level encoding, and the other is LSTM-based character-level word encoding. The former (word-level) approach makes it possible to learn pattern-based word representations by encoding them with different literals but with the same representations [8,9]. However, the character-level representations are obtained by passing each character within a word through the LSTM block [27]. Thus, the LSTM network captures prefixes and suffixes from the given input text [44]. Such representations have the advantage of handling the out-of-the-vocabulary (OOV) problem because the network can learn all character representations by sharing the morpheme-level information from a moderate or even small [11] corpus. We use the BiLSTM network to learn task-specific [27] character-level and word-level representations. Our task-oriented strategy includes the knowledge-base of NE into the training process in order to reveal the functional attributes of words in the embedding space. In this way, the baselines are trained using character-level and regular word representations. We get two important benefits from adding character-level representations into our baseline as well as the proposed model.

- Infrequent and OOV (unseen) words with low-quality representations can get extra information from character glyphs and morphemes.
- The character-based representations act as a highly generalized model of typical character-level patterns, allowing the word representations to act as a memory storing exceptions to these patterns.

3.2. Neural Baseline Models

A series of previous works adopted LSTM [45] and BiLSTM [46] networks in a variety of sequence classification tasks with other variants, such as self-attention and CRF. Sindhi NER is a novel and challenging task [17,21]. Therefore, firstly, we exploit LSTM [45], BiLSTM [46], and BiLSTM-CRF [24] networks to form strong baselines and then add more variants, including character-level and word-level representations, self-attention, and CRF decoder. In word-level baseline architectures, the sequence of words is given as an input to learn word-level representations. Our character-level baseline models consider each sentence as a sequence of characters [36] and output a label for each character. Thus, the word-level labels are converted to character-level labels. The output of the LSTM or BiLSTM encoder is mapped to the number of labels through a dense layer for decoding. In this way, every token in a given sentence gets a probability distribution for the possible NER label to select its maximum probability. The neural baseline models are illustrated as below:

- WLSTM: The unidirectional LSTM network [24] with 200d forward $\overleftarrow{h}$ hidden layers. The WSTM baseline model relies on word-level representation learning.
- WBiLSTM: The word-level BiLSTM network [24] with 200d forward $\overrightarrow{h}$ and 200d backward $\overleftarrow{h}$. The combination of both $\overrightarrow{h}$ and $\overleftarrow{h}$ resulted in 400d. The WBiLSTM baseline model is exploited using softmax, CRF, and self-attention on the pretrained SdGloVe, SdfastText, and task-oriented character-level and word-level representations.
- CBiLSTM: The character-level BiLSTM network [36] with 200d forward $\overleftarrow{h}$ and 200d backward $\overrightarrow{j}$ hidden LSTM layers, similar to WBiLSTM. The CBiLSTM baseline model is also exploited with softmax, CRF, and self-attention using SdGloVe, SdfastText, and task-oriented character-level and word-level representations.

3.3. The Proposed CaBiLSTM Model

The above-presented baseline neural models are not as strong as models based on the feature combinations. Thus, a simple NN-based model can not tackle ambiguous cases that rely on the combination of features and inter-word dependencies. To tackle these issues, we propose a contextual feature-enriched CaBiLSTM model for the Sindhi NER task, which consists of five different types of layers (see Figure 1) stacked one by one: (1) character-level bidirectional encoder; (2) word-level contextual representation layer; (3) BiLSTM network; (4) hidden layer; (5) Self-attention; (6) decoder. We illustrate every step of our proposed CaBiLSTM model as follows:

3.4. Character-Level Bidirectional Encoder

Following the majority of sequence tagging models [9,11,13], the bidirectional encoder considers each sentence as a sequence of characters as input and outputs a label distribution for each character by passing through $\overleftarrow{LSTM}$ and $\overrightarrow{LSTM}$ layers. For a given sentence $S = \{c_1, c_2, \ldots, c_n\}$, the final vector is created by a Sindhi character feature $e_i^c \in \mathbb{R}^{d_c}$ and an NER feature $e_i^n \in \mathbb{R}^{d_p}$. Formally, the embedding vector of each character c_i can be formulated as $e_i = e_i^c \oplus e_i^n$, where $i \in \{1, 2, \ldots, n\}$; e_i^c and e_i^n are Sindhi character representations and NER feature representations, respectively; d_c and d_n are the dimension of Sindhi character representations and the NER feature vector; $\oplus$ denotes concatenation operations. The character-level representations capture shape and morphological information and infer unseen (OOV) words by sharing information about morpheme-level regularities.

3.5. Contextual Representation Layer

In this layer, each output of the character representations passed through forward $\overleftarrow{LSTM}$ and backward $\overrightarrow{LSTM}$ is concatenated to get the contextual representation of a corresponding word. The dimension of an input layer is similar to the feature size. The final word-level representations are obtained by concatenating bidirectional forward $\overrightarrow{LSTM}$ and backward $\overleftarrow{LSTM}$ layers, which efficiently encode the contextual information. Such representations capture the semantic and syntactic information of the given input. In this way, the CaBiLSTM model jointly encodes contextual representations at the $Word_{Character}$ level. Then, the output of the BiLSTM encoder is mapped to the number of labels through a dense layer and a linear activation function for the input to the self-attention layer. Moreover, we implement a task-oriented strategy [27] for CRL, which is different from the pretrained SdGloVe and SdfastText representations.

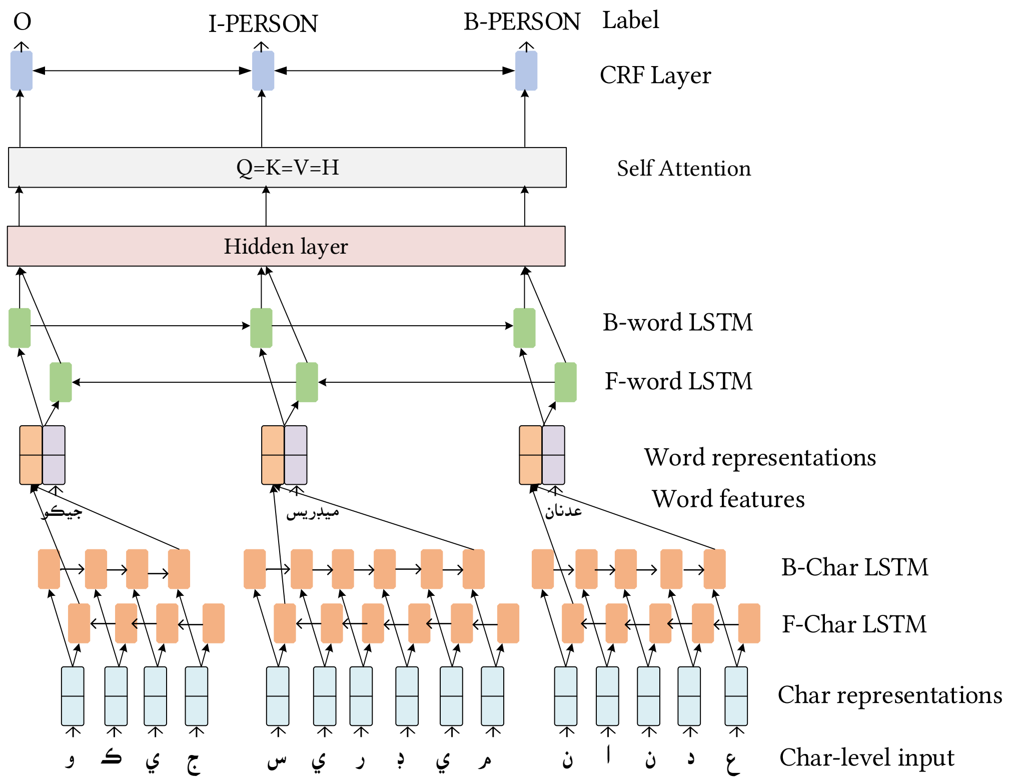


Figure 1. The architecture of our proposed CaBiLSTM model. The initial step is to map each sentence as a sequence of characters to output a label distribution for each character position. Afterwards, the BiLSTM network transforms the sequence of tokens into the word representations. In this way, word-level features are composed of the character-level BiLSTM network. Then, the concatenated output of contextual representations is fed to the self-attention layer through a hidden layer. Finally, the CRF jointly decodes the best chain of NE tags of the given input instead of decoding each label independently.

3.6. Bidirectional Network

Inherently, unidirectional LSTM [45] encode the information in one direction $\overrightarrow{h_t}$. The BiLSTM [46] has the ability to encode the sequences from both right $\overrightarrow{h_t}$ as well as left $\overleftarrow{h_t}$ contexts. The basic idea is to present each sequence into forward $\overrightarrow{h_t}$ and backward $\overleftarrow{h_t}$ hidden states h_t to capture the past and future information for context representation. For example, by giving an input sentence $s = (w_1, w_2, w_3, \ldots, w_n)$, which contains n words. The BiLSTM network computes the forward $\overrightarrow{h}$ and then backward $\overleftarrow{h}$ hidden states. Finally, the context vector is created by concatenating both hidden states, $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, for the final output $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$ of the network. However, the LSTM hidden state h_t takes information only from past, knowing nothing about the future. Thus, we use BiLSTM to extract contextual features. The processing flow in the BiLSTM network can be expressed as follows:

$$\begin{bmatrix} \tilde{c}_t \\ f_t \\ o_t \\ i_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W^\top \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + b \right) \tag{1}$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{2}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3}$$

where i_t, f_t, and o_t indicate the input, forget, and output gates; $\sigma()$ represents the sigmoid function; $W^\top$ and b are the trainable parameters; x_t is an input vector of the current time step; $\odot$ represents the dot product function.

3.7. Self-Attention Layer

The limitation of an encoder-decoder model is their fixed-length connection [29] of the intermediate semantic vector because the encoder compresses the whole sequence of information into a fixed-length vector. This approach has two shortcomings: (1) The semantic vector cannot fully represent the sequential information, which results in the information in the input sequence not being fully utilized. (2) An input sequence's length grows when the later-entered information may overwrite the first-entered information. To tackle the limitations of the encoder-decoder, self-attention [11,47–49] is implemented to extract as much feature information as possible from the input sequences. Thus, to enhance the ability of the proposed model, and we incorporate token-level multi-head attention to the output $H = [h_1, h_2, h_3 \dots h_n]$ of concatenated BiLSTM states to learn the internal structure of sentences and capture token-level dependency. Self-attention [31] before the CRF [11] automatically focuses on the specific Sindhi tokens that play a crucial role in NER and captures the essential semantic information while reducing the focus on less important information in the input sequence. We use the scaled dot product to calculate the similarity between query Q and key K matrices to obtain the weights of each word. Then, we normalize the obtained score and calculate the weights using softmax. Afterwards, we obtain the final attention by a weighted sum of the value matrix V and the weights. The intuition of the self-attention is described as follows:

$$\text{attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{4}$$

where $Q \in \mathbb{R}^{n \times 2d_h}, K \in \mathbb{R}^{n \times 2d_h}$, and $V \in \mathbb{R}^{n \times 2d_h}$ are the query, key, and value matrices, respectively. d_h is the dimension of both $\overrightarrow{h}$ and $\overleftarrow{h}$ hidden units of BiLSTM, which equals $2d_h$. Firstly, the attention mechanism linearly maps **Q**, **K**, and **V** in h times with different weights matrices. Then h projections perform parallel scaled dot-product attention. Finally, these results of an attention layer are concatenated and once again mapped to get the new representations. In our setup, we use self-attention as a set $Q = K = V = H$, which aims to capture the inter-word dependencies in the input sequence, where H denotes the output of the BiLSTM network. Formally, the function can be stated as follows:

$$\text{h}_i = \text{attn}\left(Qw_i^Q, Kw_i^K, Vw_i^V\right) \tag{5}$$

$$\mathbf{H}' = (\text{h}_i \parallel \dots \parallel \text{head}_h)\mathbf{w}_o \tag{6}$$

where $w_i^Q \in \mathbb{R}^{2d_h \times d_k}$, $w_i^K \in \mathbb{R}^{2d_h \times d_k}$, and $w_i^V \in \mathbb{R}^{2d_h \times d_k}$ are the trainable projection parameters, and $d_k = 2d_h/h$. $w_o \in \mathbb{R}^{2d_h \times 2d_h}$ are the trainable parameters.

3.8. Decoder

The tag decoder or an output layer is the last stage in an NER model that takes the input of context-dependent representations and outputs a tag sequence corresponding to the input. The CRF is an effective approach for sequence tagging problems [50] because it learns the scoring function from tag pairs, such as [B-I-O] in the training phase [24]. It is useful to consider the correlation in the neighboring labels [8] and model the tag sequences jointly with CRF [51] instead of modeling them independently. For example, in NER with a standard (B-I-O) tagging scheme [52], I-GPE can not follow I-ORG. Therefore, the modeling of independent assumptions would be impossible. Formally, given a sentence $X = \{x_1, x_2, \dots, x_n\}$ with a predicted label sequence $Y = \{y_1, y_2, \dots, y_n\}$, the *tanh* function is used firstly to predict the confidence score for each possible label:

$$o_i = \tanh(W_c h_i' + b_c) \tag{7}$$

where W_c and b_c are the trainable parameters.

The CRF incorporates the transition information between succeeding NE tag sequences to obtain optimal tags over the entire sentence. In this way, CRF jointly decodes the best chain of NE tags instead of decoding each label individually. For a given sentence $X = \{x_1, x_2, \ldots, x_n\}$, the definition of the probability for the prediction result is given as follows:

$$s(X, y) = \sum_{i=1}^{n} (o_{i,y_i} + T_{y_{i-1},y_i}) \tag{8}$$

where $y \in$ (B-I-O) labels; the scoring function $s(X,i)_{y_i}$ is the output of the hidden layer at ith token; T denotes the scores of any two adjacent labels, and $T_{y_{i-1}}$ represents the score from the successful transfer of the label y_{i-1} to the label y_i; o_{i,y_i} represents the confidence score of the y_i-th label of each token.

The decoding is the search for a sequence of NE tags y with the highest conditional probability. We compute the probability p of the label sequence $\hat{y}$ and output the label sequence $\bar{y}$ by using the Viterbi algorithm [53]. The loss function is defined as:

$$p(\hat{y} \mid \mathbf{X}) = \frac{e^{s(x,\hat{y})}}{\sum_{\bar{y} \in Y_x} e^{s(x,\bar{y})}} \tag{9}$$

$$\text{loss} = -\sum_{i=1}^{N} \log p\left(\hat{y}^{(i)} \mid x^{(i)}\right) \tag{10}$$

$$\bar{y} = \arg\max_{y \in Y_x} s(X, y) \tag{11}$$

We use the gradient backpropagation method to minimize the loss function.

4. Experimental Setup

In this section, we illustrate the experimental setup for the performance evaluation of the baselines and the proposed CaBiLSTM model on the SiNER dataset. Since our baselines and the CaBiLSTM model rely on labeled training data, no external resources are used, such as gazetteers. We use the TensorFlow [54] in python for the implementation of baselines as well as CaBiLSTM models. All the experiments are run using a single GTX 1080Ti Nvidia GPU at SMILE Lab. UESTC, China.

4.1. Dataset

We utilize the recently introduced gold-standard SiNER dataset [21] with the most commonly used (B-I-O) tagging scheme [35]. The SiNER dataset contains a large number of NEs, annotated with the Sindhi news corpus. The complete statistics of the dataset used in training, development, and test sets are depicted in Table 2 with the number of NEs in each label, and the format of the SiNER is shown in Table 3. The B-prefix before an NE label indicates the beginning of an NE, I-prefix before a label denotes the nested NE, and O-tag denotes that a token does not belong to any NE.

4.2. Evaluation Metrics

Generally, an NER system is evaluated by identifying boundaries and types of named entities. Hence, an NE is considered correct if recognized with both boundaries and its type match the ground truth [35]. We compute precision (P%), recall (R%), and F1-score (F1%) on the number of True-Positives (TP), False-Positives (FP), and False-Negatives (FN), respectively. The true-positive NEs are recognized by the system that match the ground truth, false-positive NEs are recognized by the system but do not match the ground truth, and false-negative NEs annotated in the ground truth but not recognized by the NER system. Precision measures the ability of the NER system to present only correct entities. Recall measures the ability of the NER system to recognize all NEs in the given input dataset. Moreover, F1-score is the harmonic mean of precision and recall. We report a micro-average score, which is an effective measure [35,55] on the large classes in a test collection. It aggregates the contributions of NEs from all categories to compute the average by equally treating all of them.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{12}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

Table 2. The statistics of named entities in the SiNER dataset for the training, development, and test sets. The highest proportion named entities is Person names, followed by miscellaneous and geopolitical entities.

NE Type	Label	Train Set	Valid Set	Test Set
Person names	PERSON	21,515	10,540	3478
Title of a person	TITLE	5122	4703	2009
Organizations	ORG	1943	1934	741
Geopolitical Entities	GPE	9652	4814	3041
Locations	LOC	4628	1085	360
Nationalities	NORP	4678	4210	2950
Names of Buildings	FAC	568	586	130
Events, incidents	EVENT	885	399	575
Languages	LANGUAGE	972	174	52
Art work, title of books and songs	ART	1172	276	67
Miscellaneous	OTHERS	16,297	6681	4601

Table 3. The format of the SiNER dataset; the Roman transliteration of each token is given for the ease of reading.

Named Entity	Label	Roman Transliteration
عدنان	B-PERSON	Adnan
ميندريس	I-PERSON	Menderes
جيڪو	O	Jaiko
ترڪي	B-GPE	Turkey
جو	O	Jo
1960	B-OTHERS	1960
-	O	-
1950	B-OTHERS	1950
وزير	B-TITLE	Vazeere
اعظم	B-TITLE	Azam
رهيو	O	Rahyo
.	O	.

4.3. Training Setup

The training setup consists of two phases. In the first phase, we train non-contextual pretrained word-level and character-level SdGloVe and SdfastText representations. In the second phase, we train and evaluate all the baselines and the proposed CaBiLSTM model.

4.3.1. Uncontextualized Representations

For the pretraining of SdfastText, we use sub-sampling [43] and negative sampling (NS) to select the minimum and maximum length of character n-grams (charn) [28] for character-level representations. Furthermore, the minimum word count (minw) is also an important parameter to discard rare or less frequent words in the fastText [26] algorithm. Notably, the size of the input vocabulary decreases at a large scale by ignoring more words. Similarly, the vocabulary size increases by considering rare words. Therefore, ignoring words with a frequency of less than four in SdfastText yields better results [30]. We train SdGloVe with AdaGrad by choosing the same CWS (see Figure 2) and the dimension of the character-level, word-level representations, and filter out stop words [30] in the preprocessing step for SdGloVe. The optimal hyperparameters for pretraining of SdfastText and SdGloVe are shown in Table 4.

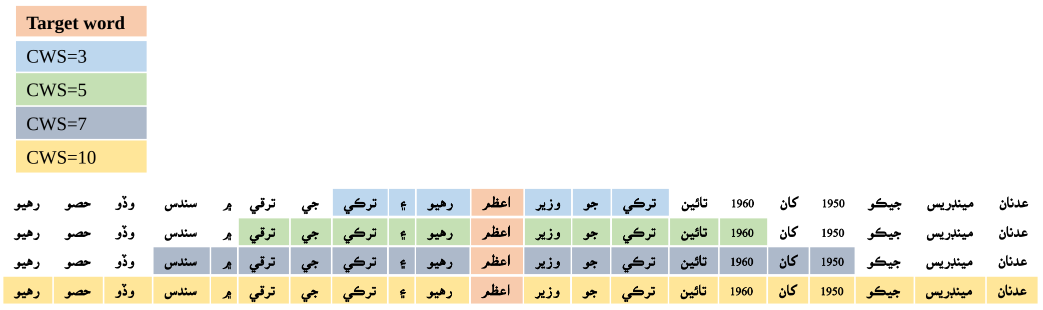

Target word
CWS=3
CWS=5
CWS=7
CWS=10
عدنان ميندريس جيڪو 1950 کان 1960 تائين ترڪي جو وزير اعظم رهيو ۽ ترڪي جي ترقي ۾ سندس وڏو حصو رهيو

Figure 2. An example of a sentence taken from the SiNER dataset to select the context window size (CWS). The writing direction of Sindhi Persian-Arabic is right-to-left.

Table 4. Optimal hyperparameters for pretrained word-level and character-level (Char) SdGloVe and SdfastText representations, neural baselines, and the proposed CaBiLSTM model. † denotes the parameters for SdGloVe, SdfastText, and task-specific representations. ‡ denotes the parameters for SdfastText only.

Model	Hyperparameter	Range
Representation learning	Epochs †	100
	Learning rate †	0.025
	CWS †	7
	*P*retrained word representations † *d*	300
	Task-specific word representations *d*	300
	CRL representations *d*	300
	Char representations † *d*	64
	minn *Char* count ‡	3
	maxn *Char* count ‡	10
	NS ‡	20
	minw ‡	3
Neural models	Learning rate	0.02
	Decay rate	0.05
	Gradient normalization	0.82%
	h layers	200
	Dropout λ	0.25
	Batch size	32
	Epochs	40

4.3.2. Training Neural Models

Several hyperparameters and configurations are evaluated in our experiments, and we picked optimal parameters that work well on the SiNER dataset. We used early stopping [56] for regularization. According to our experiments, the early stopping is based on the performance on the validation sets, and the model converges in around 40 epochs. We use Adam Optimizer [57] with default settings. We use similar CWS and dimensions of the pretrained character-level and word-level representations, as well as CRL. We update the parameters of neural baselines and the CaBiLSTM model after each mini-batch size of

32 sentences and shuffle the order of mini-batches before each epoch. We use both forward and backward BiLSTM layers of size 200 and apply variational dropout [58], which applies the same dropout mask [59] $\forall tokens$ in the same sentence. The gradient normalization [60] is used to stabilize the training procedure by setting the maximum normalization [59] to 0.82% to prevent gradients from diverging. All the hyperparameters are adjusted according to the final performance of the neural model on the test sets by random search. The details of hyperparameters are summarized in Table 4.

5. Results and Analysis

5.1. Baseline Results

We conducted several experiments on the SiNER dataset to determine whether the performance of the neural baseline models of LSTM, BiLSTM, and BiLSTM-CRF genuinely rely upon the character-level and word-level representations, an attention mechanism, or otherwise due to setting up more model parameters. All the baseline models are used to predict the NEs, including the name and title of a person, names of organizations, places, locations, nationalities, buildings, events, languages, artwork, and miscellaneous, in the SiNER dataset. The training data are used to train the models using similar hyperparameters (see Table 4). The development data are used for choosing the best configurations, and the presented results are based on the test set. Table 5 shows an average precision, recall, and F1-score on the test set. The results in the table demonstrate the slight performance difference between pretrained word representations of SdGloVe, SdfastText, and task-specific character-level and word-level representations. The first WLSTM baseline model yields weak results compared to WBiLSTM using pretrained SdGloVe, SdfastText, and task-specific representations. It is because the BiLSTM model is capable of capturing contexts better than LSTM with its bidirectional layers. Thus, we do not employ more variants to WLSTM, such as character-level representations, self-attention, and the CRF. Thanks to the bidirectional model that has access to both left and right contexts of the given input for better contextual representations. The BiLSTM network yields better overall results with self-attention mechanisms and CRF with character-level and word-level representation learning. However, the character-level BiLSTM-Attention-CRF model yields superior results than other baselines in SdGloVe and SdfastText by yielding F1-scores of 87.61% and 89.27%, respectively. Notably, the BiLSTM-Attention-CRF baseline model surpasses all the baselines by attaining the F1-score of 89.42% and 90.39% using the task-specific character-level and word-level representations, respectively.

5.2. Parameter Sensitivity

For parameter sensitivity analysis, we apply dropout [58] to mitigate the over-fitting problem and CWS for context selection for the CaBiLSTM model. Dropout λ ensures the robustness of the neural model without depending on the change of weights. To avoid overfitting, we employ the variational dropout before the character-level input to the BiLSTM encoder. At the start of training, the dropout randomly removes some neurons without changing the input and output layers. Thus, the network loses some hidden layers due to the deletion of neurons. After training, the network restores the temporarily deleted part. For the parameter adjustment, we select a two-dimensional tuple that corresponds to the input and output dropout. For better training, after parameter sensitivity analysis, we finally set it to $\lambda = 0.25$. We also evaluate the performance of the CaBiLSTM model using various CWS settings. The large CWS means considering more words in both the left and right context (see Figure 2), taking more training time. For instance, in $CWS = 10$, the size of the context will be larger than $CWS = 5$, which will capture twice as much contextual input. Both parameters influence the average performance of the proposed model, which can be seen in Table 6 and Figure 3. In Table 6, we found a good trade-off in the model performance by applying a dropout rate of 0.25 and a position of $CWS = 7$, respectively.

Table 5. Results of neural baseline models on the SiNER test set on the pretrained and task-specific character-level and word-level representations. Bold font denotes the best results.

Representation	Model	Variants	Precision%	Recall%	F1-Score%
SdGloVe	WLSTM	softmax	84.41	84.85	84.73
	WBiLSTM	softmax	86.33	85.76	85.92
	CBiLSTM	softmax	86.74	85.9	86.15
	WBiLSTM	CRF	87.19	86.39	86.79
	CBiLSTM	CRF	87.64	86.71	86.83
	WBiLSTM	Attention, CRF	86.59	87.46	87.32
	CBiLSTM	Attention, CRF	**87.93**	**87.41**	**87.61**
SdfastText	WLSTM	softmax	85.23	85.94	85.24
	WBiLSTM	softmax	87.62	86.34	86.58
	CBiLSTM	softmax	87.84	86.55	87.16
	WBiLSTM	CRF	87.69	87.8	87.63
	CBiLSTM	CRF	88.36	88.72	88.24
	WBiLSTM	Attention, CRF	88.89	88.53	88.79
	CBiLSTM	Attention, CRF	**89.45**	**89.42**	**89.27**
Task-specific	WLSTM	softmax	86.48	86.63	85.82
	WBiLSTM	softmax	87.61	86.28	86.9
	CBiLSTM	softmax	88.35	87.59	87.26
	WBiLSTM	CRF	88.62	88.91	88.58
	CBiLSTM	CRF	**89.34**	**89.38**	**89.21**
	WBiLSTM	Attention, CRF	**89.45**	**89.29**	**89.42**
	CBiLSTM	Attention, CRF	**90.48**	**90.52**	**90.39**

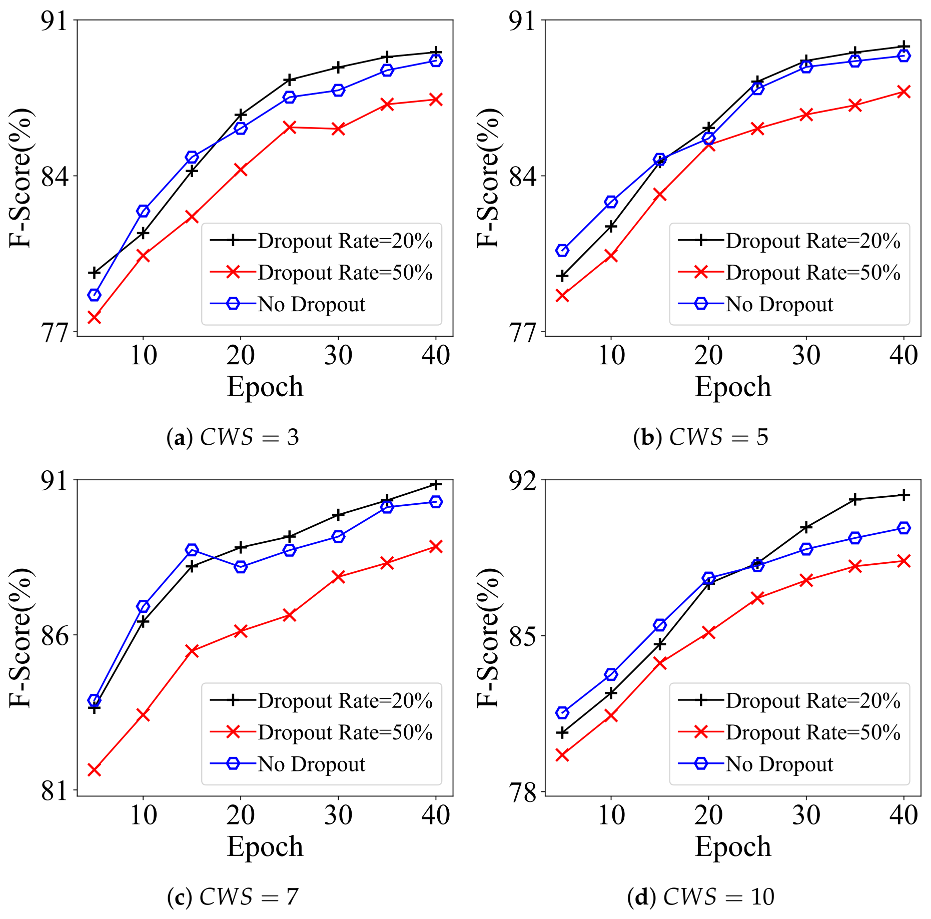
Table 6. The impact of context (length of CWS) and various dropout rates for the CaBiLSTM model on the SiNER test set using the contextual representation learning approach. The bold results highlight the best performance.

CWS	No Dropout			Dropout = 0.25			Dropout = 0.5		
	P%	R%	F%	P%	R%	F%	P%	R%	F%
3	88.74	89.41	88.43	89.67	90.28	89.55	88.24	88.39	89.17
5	90.21	90.47	90.36	90.86	90.61	90.83	89.37	89.42	89.28
7	90.79	90.94	90.89	**91.43**	**91.76**	**91.25**	90.79	90.29	90.59
10	90.84	90.9	90.82	**91.13**	**91.32**	**91.28**	90.36	90.72	90.24

It can be observed from Figure 3 that 0.25 dropout converges slower than the one with no-dropout but avoids over-fitting. However, the training error is higher with a dropout of 0.5 than a dropout of 0.25 or without dropout. It seems to be under-fitting. The results in Table 6 are presented in precision, recall, and F1-score on the development set. It can be observed that $CWS = 7$ is a better choice in all cases. Hence, we choose the optimal $CWS = 7$ and apply 0.25 dropout to baselines and the proposed CaBiLSTM model for balancing a trade-off between efficiency and accuracy.

5.3. Final Results

The baselines and CaBiLSTM models had several components that we could tweak to understand their impact on the overall performance. We analyzed the effect of pretrained SdGloVe, SdfastText representations, task-specific character-level, word-level representations, self-attention, and softmax, CRF classifiers. It can be observed in the performance comparison of baselines and the CaBiLSTM model that F1-score obtained by CaBiLSTM (see Table 7) is higher than all the baselines for three types of representation learning. The word-level and character-level representations almost yield close results to each other on the GloVe.



(**a**) $CWS = 3$ (**b**) $CWS = 5$ (**c**) $CWS = 7$ (**d**) $CWS = 10$

Figure 3. The performance of the CaBiLSTM model with different dropout rates and context settings (*CWS*) on the SiNER development set.

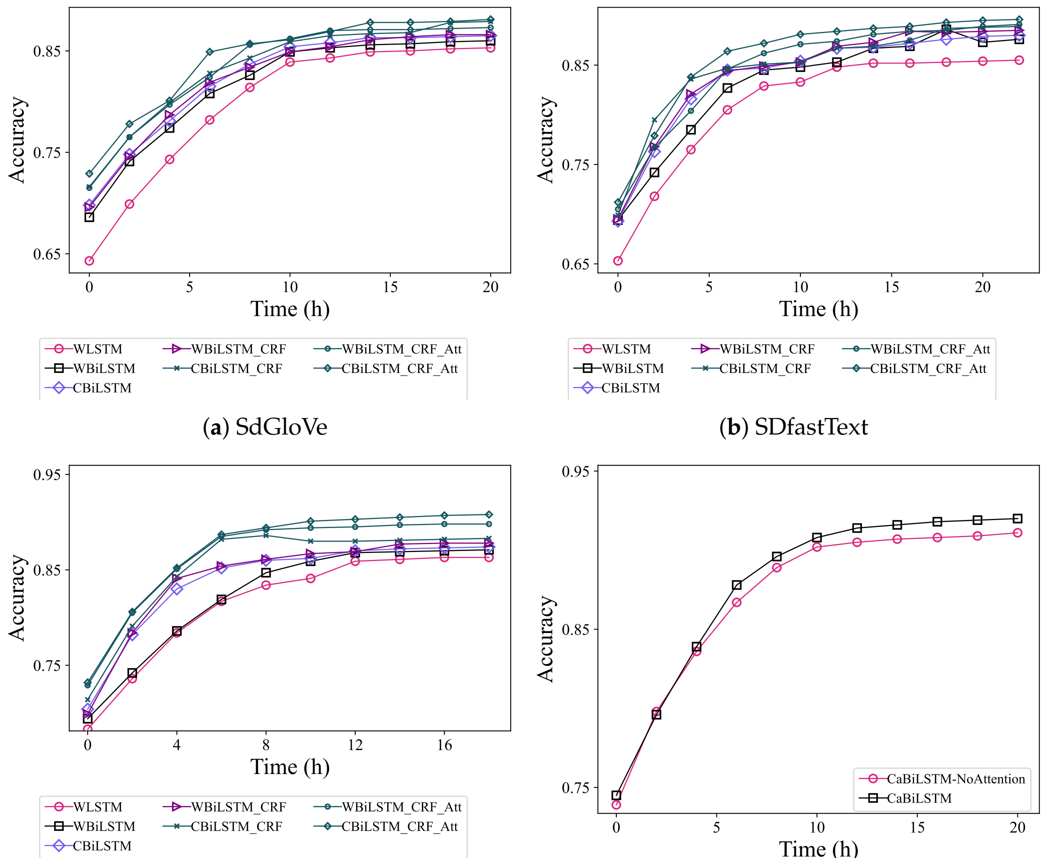
Table 7. The results of the proposed CaBiLSTM network on the SiNER test set. Bold font denotes the best results.

Representation	Variants	Precision%	Recall%	F1-Score%
SdGloVe	No-attention	87.92	88.27	88.16
	Attention	88.37	89.25	88.79
SdfastText	No-attention	89.74	**90.18**	89.76
	Attention	**90.27**	**90.64**	**90.11**
CRL	No-attention	**90.58**	**90.86**	**90.82**
	Attention	**91.43**	**91.76**	**91.25**

In contrast to SdGloVe, the character-level representations achieve better performance than word-level representations in SdfastText. However, the task-specific character-level and word-level representations achieve the best performance. It is because representation learning with the BiLSTM model captures the contextual features. The presented results also demonstrate that the CRF is dominant over softmax. The CRF gains +0.87% and +0.58% on the word-level WBiLSTM and character-level CBiLSTM models with SdGloVe representations. Similarly, CRF is also beneficial in SdfastText and task-specific representation learning, respectively. Moreover, self-attention has boosted the F1-score in all the experiments. The task-specific word-level, as well as character-level representations, are dominant along with self-attention and CRF. Moreover, it is observed that CRL gave us the most significant increase in the overall performance of +2.46% and +1.14% in F1-score over pretrained SdGloVe and SdfastText representations with the CaBiLSTM model. Furthermore, the training accuracy and time of each baseline and proposed model are depicted in Figure 4. The lower training accuracy of the SdGloVe (see Figure 4a) model indicates that the model configuration does not capture the complexity of the given training data. The training accuracy of all baseline models is better with SdfastText (see Figure 4b) representations than SdGloVe. However, neural baseline models achieve high training accuracy using task-specific representations (see Figure 4c) as compared to both pretrained SDGloVe and SdfastText representations. Overall, the CaBiLSTM model attains more accuracy in less training time than all the baselines, as depicted in Figure 4d. It can be observed that task-specific joint representation learning as CRL is more beneficial for performance gain. We also present the label-wise F1-score of our CRL-based CaBiLSTM model in Figure 5. This Figure shows that NEs of ORG, LOC, EVENT, and ART get lower F1-scores due to the context-sensitivity and other ambiguities discussed in the earlier section. In contrast, the other NEs, such as NOPR, FAC, and LANGUAGE, get high F1-scores.

5.4. Comparison with Existing Sindhi NER Systems

The CaBiLSTM model also surpasses previous Sindhi NER systems [21] based on the BiLSTM-CRF with GloVe and fastText representations. Table 8 shows that the CaBiLSTM with SdGloVe representations obtain +4.12% improvement in F1-score over previously reported results of BiLSTM-CRF-GloVe [21]. Moreover, the CaBiLSTM model with SdfastText also adds a +0.95% increase in the F1-score compared to the best-reported results of BiLSTM-CRF-fastText [21]. Furthermore, our proposed CaBiLSTM model surpasses all the neural baselines and the previous best-reported results by attaining a 91.25% F1-score.



(**a**) SdGloVe (**b**) SDfastText

(**c**) Task-specific representations (**d**) CRL

Figure 4. Training accuracy and time of neural baselines and the CaBiLSTM model on the SdGloVe, SdfastText, task-specific, and CRL-based representations.

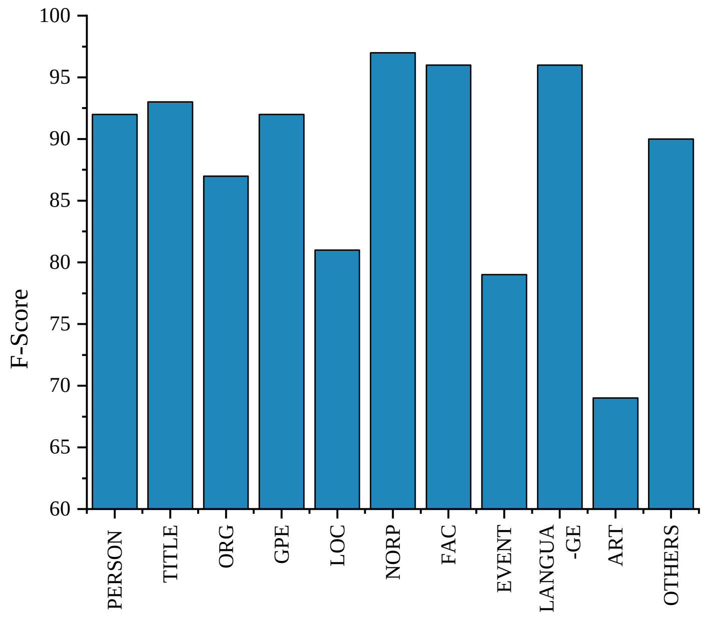


Figure 5. Label-wise F1-score of the CaBiLSTM model on the test set.

Table 8. The comparison of our proposed CaBiLSTM model with the previous works on SiNER dataset using SdGloVe and SdfastText representations. Bold font denotes the best results.

Paper	Model	Precision%	Recall%	F1-Score%
Existing work [21]	BiLSTM-CRF (GloVe)	84.40	84.93	84.67
	BiLSTM-CRF (fastText)	90.83	87.54	89.16
Our work	CaBiLSTM (SdGloVe)	88.37	89.25	88.79
	CaBiLSTM (SdfastText)	**90.27**	**90.64**	**90.11**
	CaBiLSTM (CRL)	**90.43**	**91.76**	**91.25**

Notably, the performance of the CaBiLSTM model surpasses all the baselines and the previously reported state-of-the-art results on the SiNER dataset. In particular, the CRL-based approach is dominant over SdGloVe, SdfastText, and task-oriented character-level and word-level representations.

6. Conclusions

In this paper, we propose the CaBiLSTM neural architecture for Sindhi named entity recognition and obtains excellent results in the entity extraction task. Based on the BiLSTM-Attention-CRF, we introduce task-specific contextual representation learning in the form of joint character-level and word-level representations. In addition, we exploit the neural baseline models, the impact of dropout, the context window size, and the pretrained representations, including SdGloVe and SdfastText. Our proposed CaBiLSTM model obtain new state-of-the-art performance by yielding the excellent F1-score of 91.25% on the SiNER dataset without relying on external language-specific resources, such as dictionaries or gazetteers. In the future, we intend to exploit the CaBiLSTM model on the other NER datasets. Moreover, our proposed model can be applied to sequence tagging tasks, such as part-of-speech tagging.

Author Contributions: Conceptualization, W.A.; Formal analysis, J.K., R.K. and Y.R.; Funding acquisition, Z.X.; Methodology, W.A. and Z.X.; Project administration, Z.X.; Supervision, Z.X.; Validation, J.K. and R.K.; Visualization, Y.R.; Writing—original draft, W.A.; Writing—review & editing, J.K. and Y.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Key R&D Program of China (No.2018YFB1005100 & No. 2018YFB1005104).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yadav, V.; Bethard, S. A survey on recent advances in named entity recognition from deep learning models. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 2145–2158.
2. Jumani, A.K.; Memon, M.A.; Khoso, F.H.; Sanjrani, A.A.; Soomro, S. Named entity recognition system for Sindhi language. In Proceedings of the International Conference for Emerging Technologies in Computing, London, UK, 23–24 August 2018; Springer: Cham, Switzerland, pp. 237–246.
3. Li, J.; Sun, A.; Han, J.; Li, C. A survey on deep learning for named entity recognition. *IEEE Trans. Knowl. Data Eng.* **2020**, doi:10.1109/TKDE.2020.2981314.
4. Kanwal, S.; Malik, K.; Shahzad, K.; Aslam, F.; Nawaz, Z. Urdu named entity recognition: Corpus generation and deep learning applications. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2019**, *19*, 1–13.
5. Dias, M.; Boné, J.; Ferreira, J.C.; Ribeiro, R.; Maia, R. Named entity recognition for sensitive data discovery in Portuguese. *Appl. Sci.* **2020**, *10*, 2303.
6. Yamada, I.; Asai, A.; Shindo, H.; Takeda, H.; Matsumoto, Y. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), online, 16–20 November 2020; pp. 6442–6454.
7. Tran, Q.H.; MacKinlay, A.; Yepes, A.J. Named entity recognition with stack residual LSTM and trainable bias decoding. In Proceedings of the Eighth International Joint Conference on Natural Language Processing, Taipei, Taiwan, 27 November–1 December 2017; pp. 566–575.
8. Ma, X.; Hovy, E. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 7–12 August 2016; pp. 1064–1074.
9. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural architectures for named entity recognition. In Proceedings of the NAACL-HLT, San Diego, CA, USA, 12–17 June 2016; pp. 260–270.
10. Dong, C.; Zhang, J.; Zong, C.; Hattori, M.; Di, H. Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*; Springer: Cham, Switzerland, 2 December 2016; pp. 239–250.
11. Jia, Y.; Ma, X. Attention in character-Based BiLSTM-CRF for Chinese named entity recognition. In Proceedings of the 4th International Conference on Mathematics and Artificial Intelligence, Chegndu, China, 12–15 April 2019; pp. 1–4.
12. Huang, C.; Chen, Y.; Liang, Q. Attention-based bidirectional long short-term memory networks for Chinese named entity recognition. In Proceedings of the 4th International Conference on Machine Learning Technologies, Nanchang, China, 21–23 June 2019; pp. 53–57.
13. Misawa, S.; Taniguchi, M.; Miura, Y.; Ohkuma, T. Character-based bidirectional LSTM-CRF with words and characters for Japanese named entity recognition. In Proceedings of the First Workshop on Subword and Character Level Models in NLP, Copenhagen, Denmark, 7 September 2017; pp. 97–102.
14. Mukund, S.; Srihari, R.; Peterson, E. An information-extraction system for Urdu—A resource-poor language. *ACM Trans. Asian Lang. Inf. Process.* **2010**. *9*, 1–43.
15. Li, F.; Wang, Z.; Hui, S.C.; Liao, L.; Song, D.; Xu, J. Effective Named Entity Recognition with Boundary-aware Bidirectional Neural Networks. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 1695–1703.
16. Shah, D.N.; Bhadka, H. A survey on various approach used in named entity recognition for Indian languages. *Int. J. Comput. Appl.* **2017**, *167*, 11–18.
17. Ali, W.; Kehar, A.; Shaikh, H. Towards Sindhi named entity recognition: Challenges and opportunities. In Proceedings of the 1st National Conference on Trends and Innovations in Information Technology, Nawabshah, Pakistan, 24–26 February 2016.
18. Jamro, W.A. Sindhi language processing: A survey. In Proceedings of the International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT), Karachi, Pakistan, 5–7 April 2017; pp. 1–8.
19. Motlani, R.; Tyers, F.; Sharma, D.M. A finite-state morphological analyser for Sindhi. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), Portorož, Slovenia, 23–28 May 2016; pp. 2572–2577.
20. Motlani, R. Developing language technology tools and resources for a resource-poor language: Sindhi. In Proceedings of the NAACL Student Research Workshop, San Diego, CA, USA, 12–17 June 2016; pp. 51–58.
21. Ali, W.; Lu, J.; Xu, Z. SiNER: A large dataset for Sindhi named entity recognition. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 2946–2954.
22. Liu, M.; Zhang, Y.; Li, W.; Ji, D. Joint model of entity recognition and relation Extraction with Self-attention Mechanism. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2020**, *19*, 1–19.
23. Ronran, C.; Lee, S.; Jang, H.J. Delayed combination of feature embedding in bidirectional LSTM CRF for NER. *Appl. Sci.* **2020**, *10*, 7557.

24. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv* **2015**, arXiv:1508.01991.
25. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
26. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146.
27. Liu, Q.; Huang, H.Y.; Gao, Y.; Wei, X.; Tian, Y.; Liu, L. Task-oriented word embedding for text classification. In Proceedings of the 27th International Conference on Computational Linguistics COLING, Santa Fe, NM, USA, 20–26 August 2018; pp. 2023–2032.
28. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning word vectors for 157 Languages. In Proceedings of the Language Resources and Evaluation Conference, Miyazaki, Japan, 7–12 May 2018.
29. Liao, F.; Ma, L.; Pei, J.; Tan, L. Combined Self-Attention Mechanism for Chinese Named Entity Recognition in Military. *Future Internet* **2019**, *11*, 180.
30. Ali, W.; Kumar, J.; Lu, J.; Xu, Z. Word embedding based new corpus for low-resourced language: Sindhi. *arXiv* **2019**, arXiv:1911.12579.
31. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
32. Chiong, R.; Wei, W. Named entity recognition using hybrid machine learning approach. In Proceedings of the 2006 5th IEEE International Conference on Cognitive Informatics, Beijing, China, 17–19 July 2006; Volume 1; pp. 578–583.
33. Hakro, M.A.; Lashari, I.A. Sindhi named entity recognition (SNER). *Gov.-Annu. Res. J. Political Sci.* **2017**, *5* 143–154.
34. Nawaz, D.; Awan, S.; Bhutto, Z.; Memon, M.; Hameed, M. Handling ambiguities in Sindhi named entity recognition (SNER). *Sindh Univ. Res. J.-SURJ (Sci. Ser.)* **2017**, *49*, 513–516.
35. Sang, E.T.K.; De Meulder, F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL, Edmonton, AB, Canada, 31 May–1 June 2003; pp. 142–147.
36. Kuru, O.; Can, O.A.; Yuret, D. Charner: Character-level named entity recognition. In Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; pp. 911–921.
37. Das, A.; Ganguly, D.; Garain, U. Named entity recognition with word embeddings and Wikipedia categories for a low-resource language. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2017**, *16*, 1–19.
38. Wang, X.; Jiang, Y.; Bach, N.; Wang, T.; Huang, Z.; Huang, F.; Tu, K. Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning. *arXiv* **2021**, arXiv:2105.03654.
39. Dashtipour, K.; Gogate, M.; Adeel, A.; Algarafi, A.; Howard, N.; Hussain, A. Persian named entity recognition. In Proceedings of the 2017 IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), Oxford, UK, 26–28 July 2017; pp. 79–83.
40. Luo, Y.; Xiao, F.; Zhao, H. Hierarchical contextualized representation for named entity recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 8441–8448.
41. Liu, L.; Shang, J.; Ren, X.; Xu, F.F.; Gui, H.; Peng, J.; Han, J. Empower sequence labeling with task-aware neural language model. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February2018, pp. 5253–5260.
42. Yang, J.; Liu, Y.; Qian, M.; Guan, C.; Yuan, X. Information extraction from electronic medical records using multitask recurrent neural network with contextual word embedding. *Appl. Sci.* **2019**, *9*, 3658.
43. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
44. Zhang, M.; Yu, N.; Fu, G. A simple and effective neural model for joint word segmentation and POS tagging. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2018**, *26*, 1528–1538.
45. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.
46. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681.
47. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Zhang, C. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In Proceedings of the International Conference on Representation Learning, Vancouver, BC, Canada, 30 April–3 May 2018.
48. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; Zhang, C. DiSAN: Directional self-attention network for RNN/CNN-free language understanding. In Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), New Orleans, LA, USA, 2–7 February 2018; pp. 5446–5455.
49. Zukov-Gregoric, A.; Bachrach, Y.; Minkovsky, P.; Coope, S.; Maksak, B. Neural named entity recognition using a self-attention mechanism. In Proceedings of the IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, USA, 6–8 November 2017; pp. 652–656.
50. Lafferty, J.; McCallum, A.; Pereira, F.C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning, San Francisco, CA, USA 28 June–1 July 2001; Volume 1, pp. 282–289.

51. Sutton, C.; McCallum, A. An introduction to conditional random fields. *Found. Trends® Mach. Learn.* **2012**, *4*, 267–373.
52. Sang, E.F.; Veenstra, J. Representing text chunks. In Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics; Bergen, Norway, 8–12 June 1999; pp. 173–179.
53. Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* **2006**, *13*, 260–269.
54. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
55. Van Asch, V. *Macro- and Micro-Averaged Evaluation Measures*; CLiPS: Antwerpen, Belgium, 2013; Volume 49.
56. Caruana, R.; Lawrence, S.; Giles, C. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, January 2000; Volume 13.
57. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
58. Gal, Y.; Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–8 December 2016; pp. 1019–1027.
59. Reimers, N.; Gurevych, I. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. *arXiv* **2017**, arXiv:1707.06799.
60. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1310–1318.