Neural Computing and Applications
https://doi.org/10.1007/s00521-020-04881-z

S.I. : APPLYING ARTIFICIAL INTELLIGENCE TO THE INTERNET OF THINGS

A deep neural network-based model for named entity recognition for Hindi language

Richa Sharma[1] · Sudha Morwal[1] · Basant Agarwal[2] · Ramesh Chandra[3] · Mohammad S. Khan[4]

Received: 5 December 2019 / Accepted: 20 March 2020

Abstract
The aim of this work is to develop efficient named entity recognition from the given text that in turn improves the performance of the systems that use natural language processing (NLP). The performance of IoT-based devices such as Alexa and Cortana significantly depends upon an efficient NLP model. To increase the capability of the smart IoT devices in comprehending the natural language, named entity recognition (NER) tools play an important role in these devices. In general, the NER is a two-step process that initially the proper nouns are identified from text and then classify them into predefined categories of entities such as person, location, measure, organization and time. NER is often performed as a subtask while processing natural languages which increases the accuracy level of a NLP task. In this paper, we propose deep neural network architecture for named entity recognition for the resource-scarce language Hindi, based on convolutional neural network (CNN), bidirectional long short-term memory (Bi-LSTM) neural network and conditional random field (CRF). In the proposed approach, initially, we use skip-gram word2vec model and GloVe model to represent words in semantic vectors which are further used in different deep neural network-based architectures. In the proposed approach, we use character- and word-level embedding to represent the text that includes information at fine-grained level. Due to the use of character-level embeddings, the proposed model is robust for the out-of-vocabulary words. Experimental results show that the combination of Bi-LSTM, CNN and CRF algorithms performs better as compared to the other baseline methods such as recurrent neural network, long short-term memory and Bi-LSTM individually.

Keywords Neural networks · Machine learning · Sequence labeling · Deep learning · Convolutional neural network · Bi-LSTM

✉ Mohammad S. Khan
khanms@mail.etsu.edu

Richa Sharma
sharma.ric1@gmail.com

Sudha Morwal
sudha_morwal@yahoo.co.in

Basant Agarwal
basant.cse@iiitkota.ac.in

Ramesh Chandra
ramesh.chandra@ntnu.no

[1] Department of Computer Science and Engineering, Banasthali Vidyapith, Kota, India

[2] Department of Computer Science and Engineering, Indian Institute of Information Technology Kota, Kota, India

[3] Department of ICT and Natural Sciences, Norwegian University of Science and Technology, Alesund, Norway

[4] Department of Computing, East Tennessee State University, Johnson City, TN 37614-1266, USA

1 Introduction

Humans communicate with smart IoT devices such as Alexa and Siri in natural language; thus, natural language processing is becoming an integral part of IoT devices [38–40]. To accomplish the straightforward human–computer interaction, it is preferred for machines to comprehend natural languages. In order to incorporate natural language understanding features in machines, several NLP techniques are used. The NLP techniques depict how machines can be used to process and analyze natural languages. In previous years, several NLP applications such as text summarization, question-answering, machine translation and smart conversational chatbots have been developed with the ability to understand natural language. During the development of such NLP applications, named entity recognition (NER) task is applied as a preprocessing

Published online: 04 April 2020

step to improve the performance of the overall system. NER is a subtask of information extraction (IE) field that seeks to locate and classify named entities present in text into predefined categories such as person, location, measure, organization and time. A named entity can be defined as a proper name of a real-world object such as name of person, location and date. The types of named entities are highly domain specific in biomedical data named entities such as medications, diseases, chemical ingredients and symptoms. NER is also known as entity identification, entity extraction and entity chunking. For example, consider a sentence.

नरेंद्र मोदी भारत के प्रधानमंत्री हैं।

NER task is to identify and then classify named entities as follows:

[नरेंद्रमोदी]$_{\text{Person}}$ [भारत]$_{\text{Location}}$ [के]$_{\text{Other}}$ [प्रधानमंत्री]$_{\text{Designation}}$ [हैं।]$_{\text{Other}}$.

The integration of NER in NLP applications increases the accuracy level of these applications. For example, question–answering system [2], machine translation system [3] and text clustering [4] and smart IoT devices [1] performed well after integration of NER. The smart IoT devices are now being used in smart homes, where named entities are categorized according to voice-activated devices. For example, for smart refrigerators, named entities would be food names, and for music players, it would be song titles [1].

Nowadays, these NLP applications are also available for the Hindi language; therefore, a state-of-the-art Hindi NER tool is essential to increase the performance of these applications. Most of the available systems for Hindi NER use traditional approaches such as handcrafted rules with gazetteers and machine learning-based models. Handcrafting rules are time-consuming and require either intensive knowledge of grammar or language experts to write rules. On the other hand, several machine learning-based methods have been used to implement NER including hidden Markov model [5, 6], support vector machine [7], conditional random field [8] and combination of these methods. Again, these methods rely on an intensive knowledge of grammar and handcrafted features. Handcrafting of features is difficult due to the lack of linguistic resources in Hindi. Additionally, there are several challenges in the Hindi language such as free word order, no capitalization, lack of labeled data, ambiguity in proper nouns, scarcity of linguistic resources and being morphologically rich, which need to be addressed while developing NER tool for Hindi.

In order to handle these challenges, deep learning-based models have shown significantly good results in recent times. It uses multiple layers to extract higher-level features progressively from raw text in unsupervised setting. Moving toward in this direction, a deep learning-based technique, namely recurrent neural networks (RNNs) [9], has shown state-of-the-art results in several NLP applications. However, RNNs are unable to deal with long-distance dependencies due to vanishing gradient problem and lean toward the most recent input sequence. This limitation is not suitable for NER task where knowledge of context information is essential for better performance. To overcome the limitation of RNN, long short-term memory (LSTM) [10] can be used which is a variant of RNN. In this, a LSTM cell deals with vanishing gradient problem by re-injecting past information at later time. LSTMs have *forget gate* that helps to keep only relevant information and helps to learn extremely long-distance dependencies easily and thus also resolves the issue of free word-order sentences to a large extent. However, LSTMs suffer from the unavailability of future context information as it considers only past context information, whereas, for NER, it has been observed that both past and future context information is helpful to classify a word with its correct named entity class. Thus, to access both left and right contexts, a wrapper layer called bidirectional is used. This layer is applied on LSTM to read input data in both directions, i.e., left to right and right to left; hence, it is known as Bi-LSTM [11].

In general, the word embeddings can be generated either by learning word embedding directly during training (task-specific embedding) or trained separately using a domain-specific corpus using word2vec algorithm [12] or by loading embedding vectors from a pre-trained embeddings (such as Google Word2Vec or GloVe). Pre-trained word embeddings are typically used where less amount of training data available and network is not able to learn powerful features on its own. However, word-level embeddings are unable to capture explicit character-level information such as prefix and suffix, and these features are very useful, especially with rare words. In the proposed approach, the character-level features are incorporated by applying a convolutional neural network (CNN). Santos and Guimaraes [13] have applied CNN to extract character-level information in Spanish and Portuguese NER.

Furthermore, a conditional random field (CRF) [14] approach is integrated as a top layer to increase the tagging accuracy by jointly decoding the tag sequence instead of decoding each tag independently. Several neural network-based models have been applied for the implementation of NER in many other languages such as Italian [15], English [16], Spanish [17], Portuguese [18], Japanese [19]and Chinese [20] and have shown very good results. The aim of this work is to analyze the efficiency of deep learning-based techniques on Hindi NER without using language-specific features and linguistic resources. This paper presents a hybrid model which is a combination of Bi-LSTM–CNN–CRF for the implementation of Hindi NER.

The paper is designed as follows: Sect. 2 gives a detailed history of developed models for NER. Section 3 presents the architecture of the proposed approach. Section 4 discusses the experimental setup, hyper-parameter tuning and results. Section 5 presents the conclusion and future task.

2 Related work

Conventional Hindi NER systems can be classified as rule-based systems and machine learning-based systems.

2.1 Rule-based NER system

Rule-based system for Hindi NER was developed by Kaur and Kaur [21] and worked for three new named entities named as money, direction and animal/bird. In their system, different tables were created in database and named entities were extracted from these tables. Often, rule-based approach is applied with a combination of machine learning approach to boost the performance of system [6, 28]. In [35], authors used fuzzy temporal rules for predicting breast cancer. In [36], authors developed a fuzzy rule-based content recommendation system. Since rule-based approach is very time-consuming and requires language experts to design efficient rules, researchers were focusing on machine learning-based approaches. This motivation developed several machine learning-based NER models for Hindi.

2.2 Machine learning-based NER system

Several machine learning methods such as hidden Markov model (HMM), maximum entropy, support vector machine and conditional random fields have been applied to implement NER system for Hindi language. Performance and corpus details of the best system in each mentioned approach are shown in Table 1. In this series, the first hidden Markov model-based NER system was developed by Morwal et al. [22]. In this system, authors performed NER task on two corpuses, tourism corpus and NLTK Indian corpora. Another two HMM-based NER systems were developed by Gayen and Sarkar [23] and Chopra et al. [5]. Further, machine learning-based system was developed by Saha et al. [24]. In [24], authors used maximum entropy approach with some language-specific features such as orthographic features (decimal, digits), affixes, parts of speech (POS) and gazetteers. Similarly, support vector machine approach of machine learning was experimented by Ekbal and Bandyopadhyay [7] to complete NER task for two Indian languages, Bengali and Hindi, where language-independent features and lexical context patterns were used as features. In the series of support vector machine-based models, another two NER systems were developed by Saha et al. [25] and Devi et al. [26]. In [25], authors worked on biomedical domain and used a novel kernel function for support vector machines. Another machine learning technique named as conditional random field was applied by Ekbal and Bandyopadhyay [8] and Sigh et al. [27] for NER task. Their systems used BIO standard format of tag to tag the sequence, where BIO stands for beginning, inside and other tag.

To increase the performance of NER system, several hybrid systems [6, 28] were developed by combining rule-based and machine learning-based approach, the detail of which is shown in Table 1. However, all these systems highly relied on human-designed features and linguistic resources such as gazetteers and POS taggers. On the other hand, to eliminate the need for linguistic resources and manually designed features, researchers have started working on deep learning models.

2.3 Enhanced neural network (Deep learning)-based NER systems

First, Collobert et al. [29] designed a NER model for the English language in which features were learned from word embedding trained on a huge amount of unlabeled data. In this series, Chiu and Nichols [30] proposed a NER model for English that combined Bi-LSTM with CNN to induce character-level information in the model. Similarly, Ma and Hovy [31] designed a NER model for English which was a combination of deep learning techniques, i.e., Bi-LSTM–CNN–CRF. In their model, CRF was integrated as a top layer to jointly decode labels for the whole sentence.

On the other hand, for Hindi NER systems, Athavale et al. [32] developed a model in which pre-trained word embedding was used with Bi-LSTM and softmax layer. Their system also used POS tag information of all sequences. Another deep learning-based model was developed by Gupta et al. [33] that used a gated recurrent unit (GRU) with character- and word-layer embedding and applied the model on code-mixed Indian social media text. A neural network-based model comprised of Bi-LSTM-CNN was developed by R. Murthy [37]. In [37], both word embedding and sub-word features were concatenated and fed through Bi-LSTM to softmax layer to classify named entities. However, deep learning-based models are not extensively explored for Hindi text, whereas several state-of-the-art deep learning-based NER models have been developed for other languages. Deep learning-based NER models can be developed for the Hindi language also which may produce state-of-the-art results without the use of handcrafted features and any sort of linguistic resources.

Table 1 Summarized report of previous approaches on NER task for Hindi language

S. N	Paper	Attribute	Classifier	Dataset	F-measure (%)
1	Saha et al. [24]	Static word feature, context lists, dynamic NE tags, word suffix, word prefix, POS information, numerical word	Maximum entropy	243 K words corpus, collected from Hindi newspaper 'Dainik Jagaran'	81.52
2	Ekabal and Bandyopadhya [8]	Prefix and suffix information, word length, first word, previous word tag information, POS tag, digit information, gazetteer lists	Conditional random field	IJCNLP-08 NERSSEAL shared task data	80.93
3	Saha et al. [25]	Current word, previous word information, prefix and suffix feature, NE tags of the previous words	Support vector machine	243 K words corpus, collected from Hindi newspaper 'Dainik Jagaran'	83.56
4	Biswas et al. [28]	Prefix and suffix, context word feature, digit information	Hidden Markov model+ maximum entropy	List of gazetteers annotated in Shakti standard format	71.95
5	Morwal et al. [22]	Frequency of a tag, frequency of a tag sequence, frequency of a word	Hidden Markov model	NLTK Indian Corpus	96
6	Kaur and Kaur [21]	Several rules with new "no name entity rule"	Rule-based+ list lookup	Data collected from e-copies of Hindi newspaper such as 'Dainik Jagaran' and 'Punjab Kesari'	95.77
7	Athavale et al. [32]	POS tag	Bi-LSTM	ICON 2013 NLP tools contest	77.48
8	Singh et al. [27]	Char N-gram for suffix, patterns for punctuation, emoticons, numbers, numbers in strings, previous tag information	Decision tree, CRF, LSTM	Hindi–English code-mixed 3,638 tweets (68,506 tokens) from last 8 years scrapped from Twitter	94 (decision tree), 95 (CRF), 95 (LSTM)

Table 1 shows the previously developed Hindi NER system with their F-measure and dataset information.

Table 1 shows the comparative analysis of different NER systems for Hindi language. However, comparing only F-measure of different NER systems is not justifiable to select the best NER system as many different factors such as dataset, domain factor, language factor, size of the corpus and types of entities identified also affect the system's performance.

3 Proposed approach

Most of the existing NER tools developed for Hindi language were dependent upon handcrafted feature engineering. We propose a deep learning-based hybrid approach which is a combination of the Bi-LSTM, CNN and CRF algorithms. The proposed approach for named entity recognition captures both word- and character-level features from raw text automatically rather than handcrafting the features. In this paper, an end-to-end neural network-based model is proposed to implement NER. The steps of the proposed approach are demonstrated in Fig. 1.

Since Hindi is resource-scarce language and we have a very less amount of training data, we use pre-trained word embedding approach to represent the input words in vector form. To create pre-trained word embedding model, we first learn the embedding for Hindi words on a large unlabeled text corpus in an unsupervised setting using two different word embedding approaches, skip-gram [12] and GloVe [34]. Hence, we create two pre-trained embedding models: One pre-trained embedding is generated using skip-gram approach and another is generated using Glove algorithm. These word vectors are used to initialize the words of input text during training the model. This pre-trained embedding model bridges the gap of the scarcity of labeled data to some extent. To build pre-trained word embedding, we use the unlabeled corpus of 27,000 sentences of Hindi language, provided by Indian Language Technology Proliferation and Deployment Centre (TDIL). Our model uses both pre-trained embedding model for vectorization of input words. For this purpose, the input sentence is first broken down into their constituent words and each word further initialized with pre-trained word embedding vectors. The output of the word vectorization step is a matrix of l*d for each input sentence, where l is

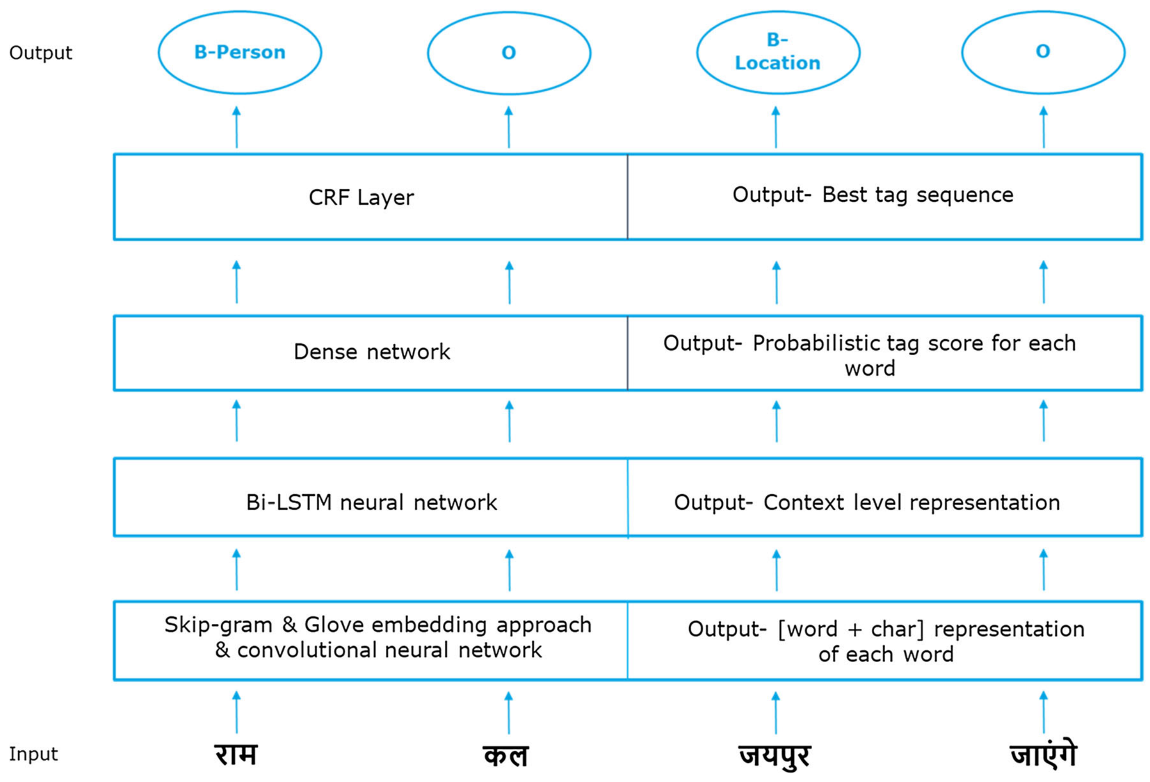
Output
B-Person
O
B-Location
O
CRF Layer
Output- Best tag sequence
Dense network
Output- Probabilistic tag score for each word
Bi-LSTM neural network
Output- Context level representation
Skip-gram & Glove embedding approach & convolutional neural network
Output- [word + char] representation of each word
Input
राम
कल
जयपुर
जाएंगे

Fig. 1 Block diagram of proposed process

equal to the number of words in the input sentence and d is the dimension of the vector.

To integrate the character-level information of a word in the proposed model, each character of a word in vector form (character embedding) is fed as an input to the convolution layer. Here, convolution layer is a one-dimensional convolutional neural network (1D CNN) that scans several characters at a time and extracts information from focused n-gram characters. This convolutional layer also produces vector representation of a word same as word embedding, but by scanning the character-level composition of that word. The convolutional layer uses several filters for scanning purposes and produces a vector corresponding to each filter. Further, 1-max-pooling operation is performed on each vector to obtain the most significant features. The obtained feature vector is known as the character-level representation of a word. The network architecture of the proposed model is depicted in Fig. 2.

Subsequently, both the word embedding and the character-level representations are concatenated to form a one vector representation of a word. However, these representations are insufficient as both representations are not concerned about context-level information of a word, and this information plays a crucial role in NER. For this purpose, the concatenated vector is fed as input to Bi-LSTM as shown in Fig. 1. Bi-LSTM appends the context-level information to the vector and produces a probabilistic tag score for each word with a dense layer as a final outcome as shown in Fig. 1. This output is fed into the CRF layer to find out the best tag sequence for a sentence from all possible tag sequences as shown in Fig. 1. The algorithm of the proposed model is illustrated in the following. The different layers of the proposed model are described in the subsequent subsections.

```
Algorithm:
Input: Sentence S
Output: Best_tag_Sequence
Dataset: training set t
Maximum length of S: wmax
Maximum length of w in S: cmax
Number of words in t: n
1. for each $w_k$ in t do
        if $w_k \in$ pre-trained_word_vectors        where $k=1,2,3\ldots n$
            $wv_{k*d} \leftarrow$ pre-trained_word_vectors ($w_k$)
        else
            $wv_{k*d} \leftarrow$ random_vector
        end if
   end for
2. for each S in t do
        $Sw \leftarrow [w_1, w2, w3 \ldots w_{max}]$        where $w_i \in s$, $i=1,2,3\ldots max$
        for each w in S do
            $wc \leftarrow [c_1, c_2, c_3 \ldots c_{cmax}]$ where $c_j \in w$, $j=1,2,3\ldots cmax$
        end for
        $Sc \leftarrow [wc_1, wc_2, wc_3 \ldots wc_{max}]$
        $Swm_{max*d1} \leftarrow$ Embedding(Sw, weights=wv)
        $Scm_{max*cmax*d2} \leftarrow$ Embedding(Sc, weights=random_initialization)
        $Scnn_1 \leftarrow$ Conv1D($Scm_{max*cmax*d2}$, filters=100)
        $Scnn_2 \leftarrow$ Conv1D($Scnn_1$, filters=64)
        $Scnn_3 \leftarrow$ Conv1D($Scnn_2$, filters=32)
        $Scm_{max*d3} \leftarrow$ 1-MaxPooling($Scnn_3$)
        $Swc \leftarrow$ Concatenate($Swm_{max*d1}$, $Scm_{max*d3}$)
        $Swc \leftarrow$ Bi-LSTM(Swc, units=$u \in \{200,150\}$)
        $P\{t\} \leftarrow$ Dense(Swc, units=n_tags)
        Best_tag_sequence $\leftarrow$ CRF($P\{t\}$)
end for
```

3.1 CNN layer

CNN is an effective approach to induce local region information in the form of word- and character-level features. CNN induces morphological information such as prefix or suffix from the character of words and converts it into neural representation. The process of generating character-level information is shown in Fig. 3. Firstly, each sentence of the training dataset is tokenized and each token is separated into its corresponding characters. Embedding layer is used to generate a vector for each character with a dimension of 'g' and with random initialization. In this work, each character is represented as a 30-dimensional vector and a uniform distribution is used for initialization. These vectors (character embedding), corresponding to each character in a word, are used as input to the convolution layer as shown in Fig. 3. Before passing input into the convolutional layer, a configured dropout operation is performed for regularization purposes. The proposed model generates character-level feature vector by using three convolutional layers with 100, 64 and 32 filters, respectively. Each convolutional layer uses 'relu' as activation function, 3 as region size and 1 as strides value. For every filter, one feature map is produced as an output of convolution operation; for example, here 32 feature maps are produced as an output as the last convolutional layer used 32 filters. The dimensionality of each feature map is a function of strides parameter of the convolution operation. Therefore, to produce a fixed-length vector, a 1-max-pooling operation is performed on each feature map which extracts maximum value from each feature map. The

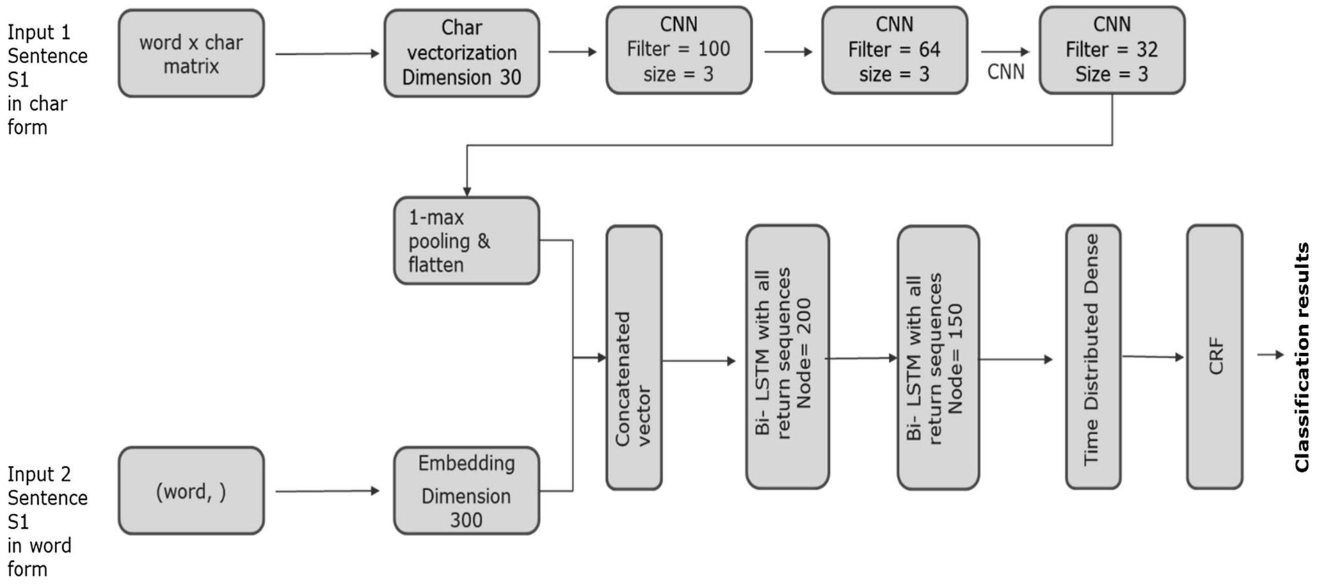

Fig. 2 Network architecture of proposed approach

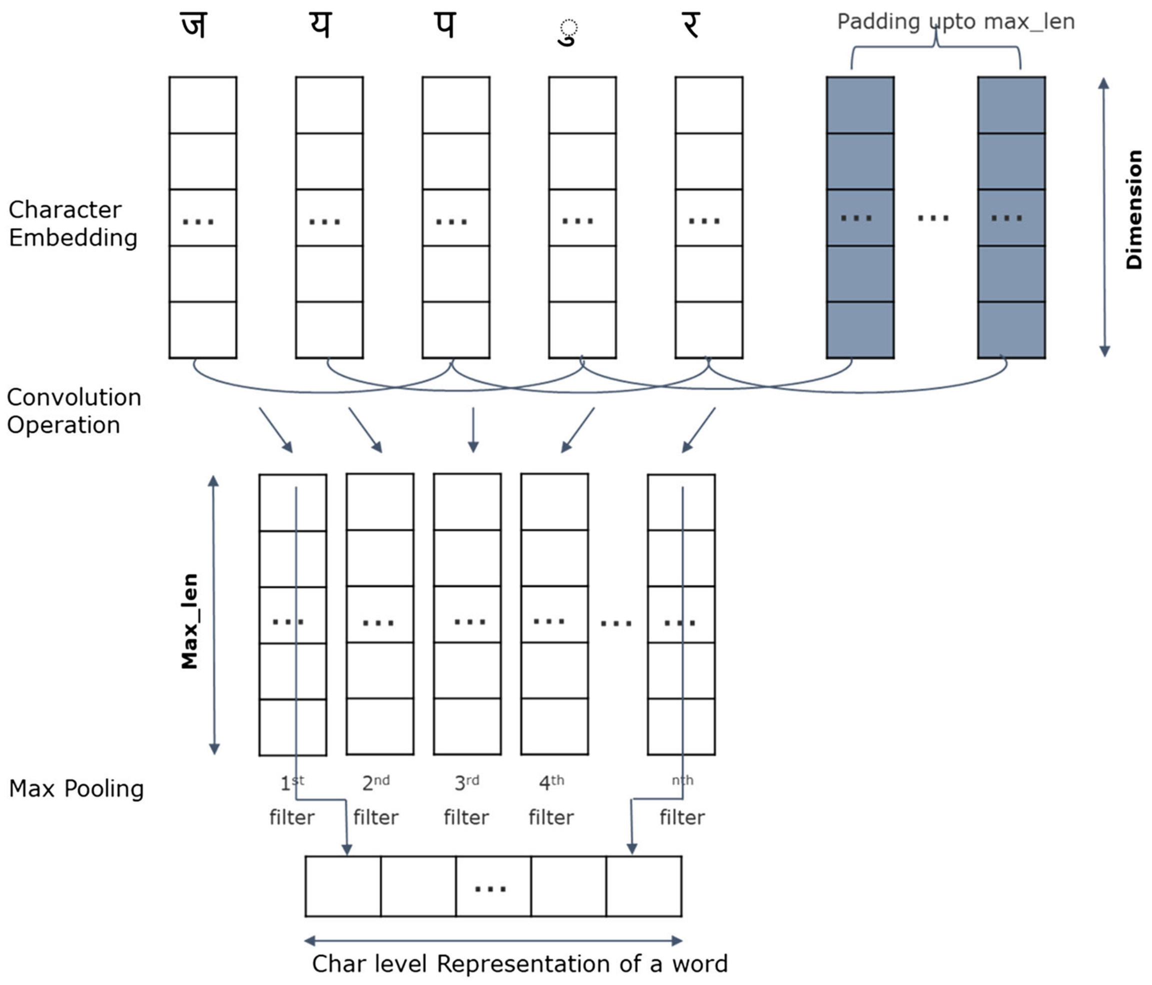

Fig. 3 Process of convolution neural network

dimension of this vector is the same as the number of filters. For example, it would be 32 in this work as 32 filters are used in the last convolutional layer during convolution operation. This vector represents character-level features of a word.

3.2 Bidirectional LSTM layer

Bidirectional LSTM (Bi-LSTM) layer combines two LSTM layers moving in two opposite directions, i.e., forward and backward, through time. Forward LSTM moves through time, from the start of the sequence toward the end of the sequence, while backward LSTM moves through time from the end of the sequence toward the start of the sequence as shown in Fig. 4. Thus, forward LSTM preserves past information from the sequence, whereas backward LSTM preserves future information from the sequence. Thus, this layer can handle variable-length input and can capture past and future context information from the sequence. In Bi-LSTM layer, this input is processed in terms of two distinct hidden states to attain both past and future context information through forward and backward LSTM, respectively. Later, these two hidden states are concatenated and produce a vector as a final outcome of Bi-LSTM layer as shown in Fig. 4 which contains contextual information of words in a sentence.

3.3 CRF layer

In the proposed model, CRF layer is used as an output layer as shown in Fig. 1. The motive of using CRF is to consider the value of neighboring tags while assigning the tag to the current word. CRF chooses the best tag sequence for a sentence after considering all possible tag sequences and a correlation between adjacent tags. In the proposed model, the output of the Bi-LSTM layer is passed through a dense layer which yields probabilistic tag scores for every word of a sentence. This output is fed into CRF layer. CRF processes the input and finds the best label sequence which has the highest prediction score for a sentence as shown in Fig. 5. The tag sequence can be determined without CRF layer; however, CRF adds some constraints to predict the best tags and verifies their validity. These constraints are learned by CRF layer from the training dataset during training and increase tagging accuracy. For example, in IOB2 annotation 'O I-label' tag sequence is invalid as the first label of one named entity should start with B- not I-, i.e., valid tag sequence can be O B-label. Consequently, CRF increases the accuracy of the predicted tag sequence.

4 Experimental settings

All experiments including baseline and proposed model are performed on NVIDIA GTX GeForce GPU with Keras 2.2.4 and Tensor flow backend support.

4.1 Dataset description

The proposed model is evaluated on NER-tagged standard corpus having 4478 Hindi sentences, provided by TDIL. The class-wise bifurcation of tags in training and test data

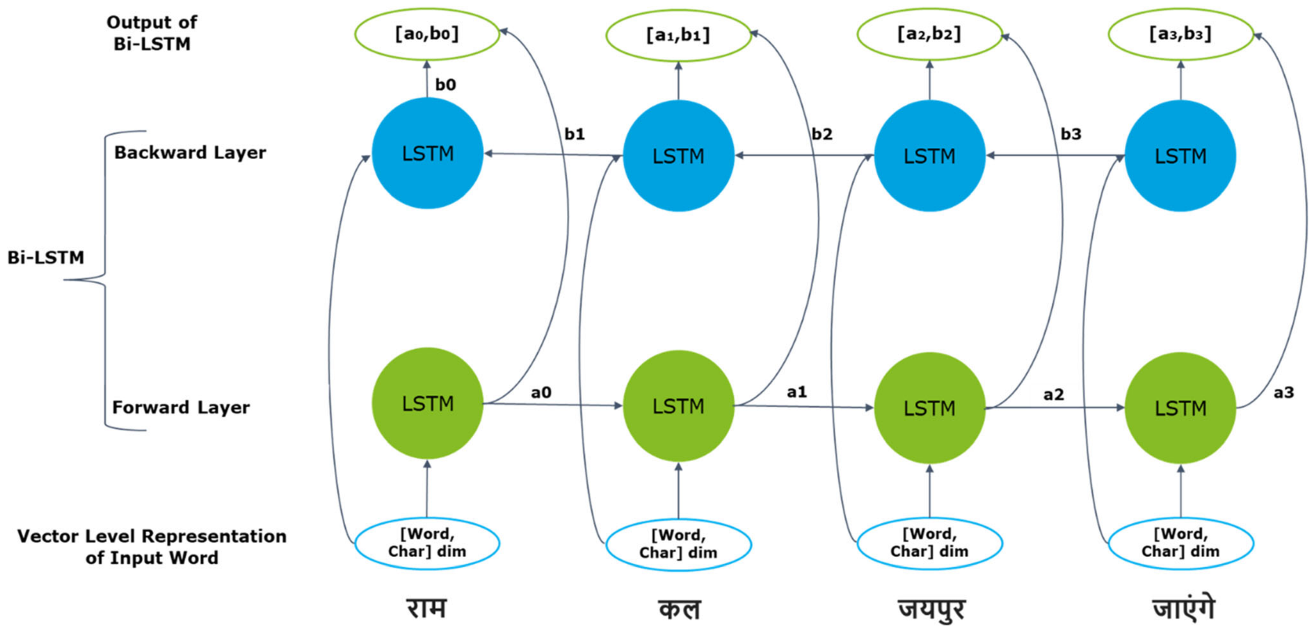

Fig. 4 Bi-LSTM layer

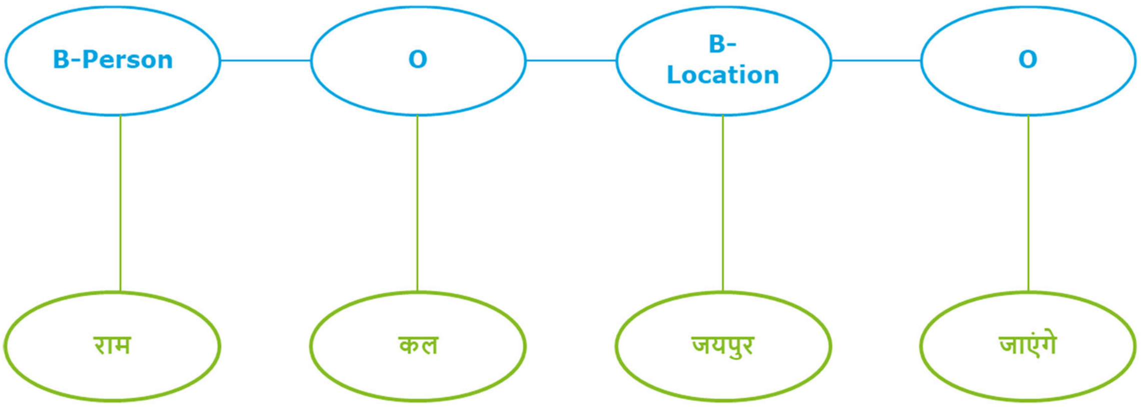

Fig. 5 CRF layer

Table 2 Statistics of tags in corpus

S. no.	Tag name	Training + validation set	Test set
1	Person	562	86
2	Entertainment	332	34
3	Location	3012	490
4	Organization	101	47
5	Livthings	179	29
6	Distance	185	18
7	Count	127	39
8	Period	92	8
9	Month	126	29
10	Others	77,967	14,268

is shown in Table 2. The training set contains 3776 sentences, whereas test data have 702 sentences. Further, 409 sentences from the training set are used for validation purposes during training. This corpus consists of nine named entities which are classified as person, entertainment, location, organization, livthings, distance, count, period and month. Those tokens which are not fit any of the mentioned label tagged as ‘O,’ i.e., others. In this paper, the model employs the IOB2 tagging scheme for tagging the tokens. IOB2 stands for inside, outside, beginning. B-tag shows the beginning of a specific chunk, and I-tag shows inside value of that chunk. Table 2 shows the statistics of tags in the corpus.

4.2 Performance measure

Generally, performance for the NER system can be estimated by calculating correct entities identified by the system and total named entities available in the corpus. In this research work, the performance of the system is evaluated in terms of precision, recall and F-measure. Precision (P) can be calculated as:

$$P = \frac{Number\ of\ correct\ named\ entities\ extracted}{Total\ named\ entities\ extracted}$$

Similarly, recall (R) can be calculated as:

$$R = \frac{Number\ of\ correct\ named\ entities\ extracted}{Total\ named\ entities\ present\ in\ corpus}$$

F-measure is the harmonic mean of precision and recall, calculated as:

$$F\text{-}measure = \frac{2PR}{P+R}$$

4.3 Hyper-parameter settings

The optimal values of hyper-parameters are determined using grid-search hyper-parameter optimization technique during the experiments. We implement grid search with different values of batch size, number of units in each layer, optimizers, dropout rate and activation functions. Total 768 configuration sets are run on NVIDIA GTX GeForce GPU to get the best values of these hyper-parameters. In these configurations, we set number of LSTM units with different values such as 50, 100, 150 and 200 and batch size such as 16, 32, 64 and 128. We show the performance of various hyper-parameters in Fig. 6. The proposed model performs well with 16 batch size and 200 LSTM units for both skip-gram and glove embedding as shown in Fig. 6c, d. In this grid, we apply activation functions such as ‘relu,’ ’tanh,’ ‘softmax’ and ‘softplus’ with optimizers such as ‘rmsprop,’ ‘adam,’ ‘adadelta’ and ‘adagrad.’ System performs well with softplus and relu activation function with rmsprop optimizer as shown in Fig. 6a, b.

Simultaneously, models are tuned for regularization with different dropout rates such as 0.5, 0.6 and 0.7.

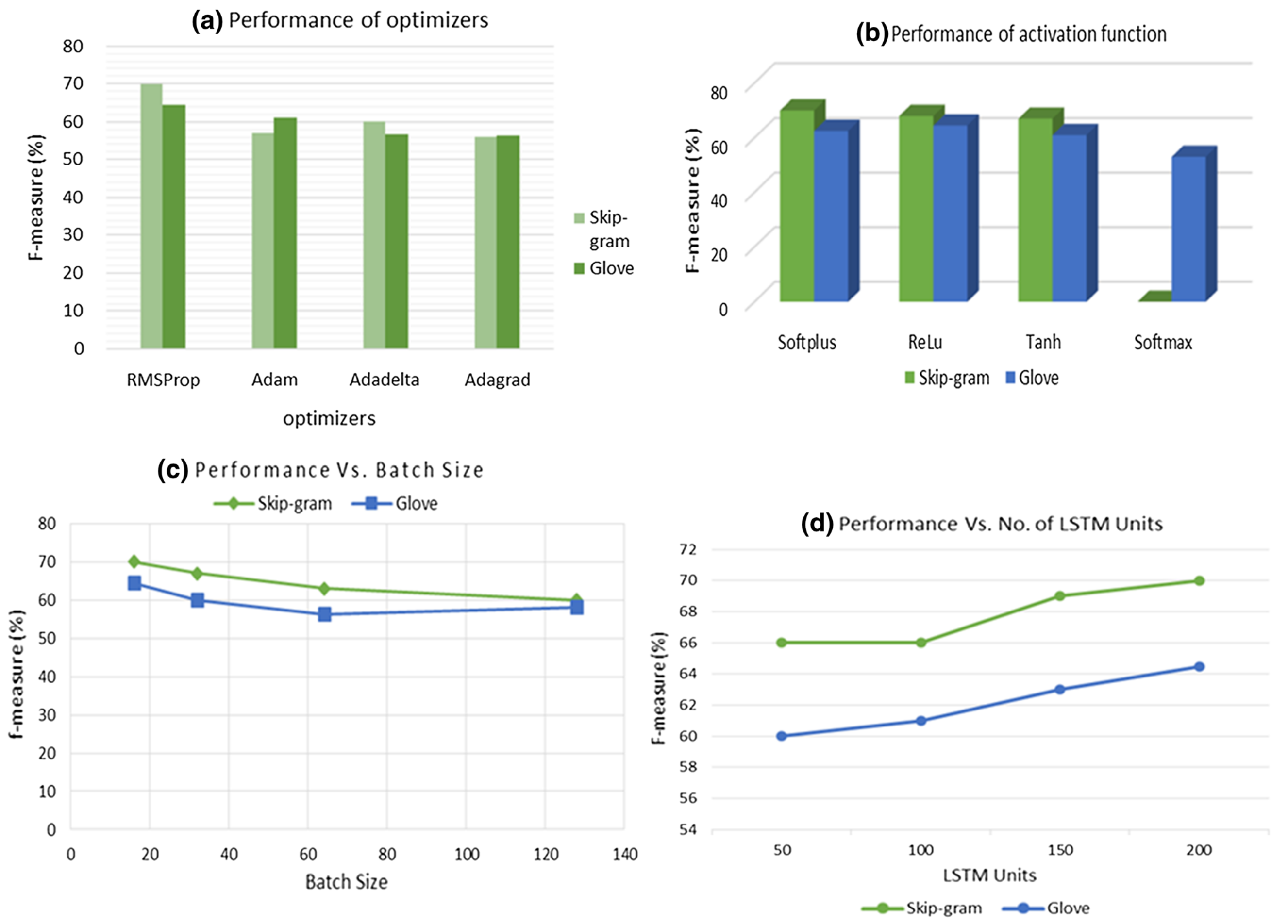

Fig. 6 Performance with different hyper-parameters

Dropout is a regularization method which is applied to overcome the overfitting problem that occurs during the training of the model. Some connections are randomly dropped out by applying the Dropout layer before and after the RNN and CNN layers. The system performs well on 0.5 dropout rate for both skip-gram and glove embedding. We also apply early stopping with 70 epochs to reduce overfitting. In this paper, 30-dimensional character embedding is used to represent characters in vector form. For convolution operation, we apply three layers of 1D convolution with 100, 64 and 32 filters, respectively. In all convolution layers, *relu* is used as activation function instead of *tanh*. The hyper-parameters used for this experiment are shown in Table 3.

4.4 Results

We evaluate various variants of the proposed approach to make sure the robustness of the proposed model. We develop various deep learning-based models for the NER tasks such as long short-term memory (LSTM), recurrent neural network and bidirectional LSTM individually. These models are considered as baseline models. All the NER models are developed using two kinds of pre-trained embedding models, i.e., skip-gram and Glove embedding. We also develop deep learning architectures by combining different combinations of the algorithms such as RNN–CNN, RNN–CRF and RNN–CNN–CRF. We show the results of all the NER models with both the embeddings with respect to the precision, recall and F-measure in Tables 4 and 5.

Experimental results show that recurrent neural networks perform better than convolutional neural networks. It is due to the reason that the recurrent neural networks capture the sequential information, whereas the CNN includes information of local context. In addition, it is clear from the results shown in Tables 4 and 5 that conditional random field (CRF) model improves the performance of the recurrent neural networks. For example, the RNN individually gives 54.9% F-measure, whereas RNN with CRF produces 58% F-score.

Further, it is observed from results shown in Tables 4 and 5 that the Bi-LSTM performs better than LSTM and RNN neural architectures when both the pre-trained

Table 3 Hyper-parameters for all experiments

S. no.	Hyper-parameter	Value
1	Character embedding dimension	30
2	Word embedding dimension	300
3	Filters	100, 64, 32
4	Region size	3
5	Dropout	0.5
6	LSTM units	200
7	Optimizer	RmsProp
8	Epoch	70
9	Batch size	16

Table 4 Results of various approaches for NER task with skip-gram embedding

Classifier	Precision (%)	Recall (%)	F-measure (%)
RNN–baseline	52	56	54.9
RNN–CNN	53	56	54.4
RNN–CRF	58	58	58
RNN–CNN–CRF	54	58	56
LSTM–baseline	58	57	57.4
LSTM–CNN	59	55	56.9
LSTM–CRF	63	57	59.8
LSTM–CNN–CRF	66	56	60.5
Bi-LSTM–baseline	68	57	61.9
Bi-LSTM–CNN	68	58	62.6
Bi-LSTM–CRF	75	61	67.2
Bi-LSTM–CNN–CRF	69	71	70

Table 5 Results of various approaches for NER task with GloVe embedding

Classifier	Precision (%)	Recall (%)	F-measure (%)
RNN–baseline	57	50	53
RNN-CNN	63	46	53.2
RNN–CRF	65	50	56.8
RNN–CNN–CRF	54	56	55.4
LSTM–baseline	64	48	55
LSTM–CNN	63	50	55.6
LSTM-CRF	61	56	58.2
LSTM–CNN–CRF	56	58	57
Bi-LSTM–baseline	66	54	59.8
Bi-LSTM–CNN	70	53	60.2
Bi-LSTM–CRF	67	58	62.3
Bi-LSTM–CNN–CRF	73	57	64.5

embeddings are used. For example, Bi-LSTM produces F-score of 61.9% which is higher as compared to 54.9% and 57.4% given by RNN and LSTM, respectively. Bi-LSTM learns the information from two directions, i.e., from previous context information to the future context information (forward direction) and from future context information to the past context information (backward direction). It is also free from the gradient vanishing problem which is a problem with recurrent neural network. It can be seen from the results that LSTM performs better than RNN network.

When the Bi-LSTM is joined with CRF model, the performance is improved significantly as compared to other models, and it gives F-score of 67.2%, whereas 61.9% is given by Bi-LSTM individually as shown in Table 4. The Bi-LSTM with CNN also improved from 61.9 to 62.6% F-score with skip-gram embedding.

The proposed model which is a combination of Bi-LSTM, CNN and CRF algorithms (Bi-LSTM-CNN-CRF), with both pre-trained embeddings, achieves the highest F-score as compared to Bi-LSTM–baseline, Bi-LSTM–CNN and Bi-LSTM–CRF models (as per results in Tables 4 and 5). The proposed model Bi-LSTM–CNN–CRF achieves 70% and 64.5% f-score with skip-gram and glove embedding, respectively. However, Bi-LSTM–CRF and Bi-LSTM–CNN achieve higher f-score as compared to the Bi-LSTM–baseline, which clearly proved that integration of these layers with Bi-LSTM boosts the performance of baseline model.

As per results in Tables 4 and 5, LSTM–CNN–CRF and RNN–CNN–CRF achieve higher f-score than their baseline models, respectively. Furthermore, it can be observed from these results that the system performs better with skip-gram embedding than glove embedding. Consequently, the proposed model Bi-LSTM–CNN–CRF with skip-gram pre-trained embedding outperforms all other models on all evaluation metrics.

5 Conclusion and future work

The NER is an important task which is often performed while processing natural languages. NER is applied to extract the proper nouns from open domain text and classify them into predefined categories such as person, location, organization, date and time. It can be performed as a subtask during the implementation of many NLP applications. In this paper, an end-to-end neural network-based NER model is proposed for Hindi language. The proposed model is an integration of bidirectional LSTM, CNN and CRF layer which identifies and classifies named entities from Hindi text. The proposed system is also capable of handling several challenges of Hindi language such as free word-order language due to the use of Bi-LSTM, and it also handles the morphologically rich language with character-level CNN. The proposed model achieves better results on Hindi NER while comparing with other state-of-the-art systems developed for Hindi NER. Since the size of available training and testing datasets is small, these datasets would be enhanced as a future task for better performance of the system.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Park G, Kim H. Low-cost implementation of a named entity recognition system for voice-activated human-appliance interfaces in a smart home, sustainability (Switzerland) 10.
2. Greenwood MA, Gaizauskas R (2003) Using a named entity tagger to generalise surface matching text patterns for question answering. In: Proceedings of the workshop on natural language processing for question answering (EACL03), Citeseer, pp. 29–34
3. Babych B, Hartley A (2003) Improving machine translation quality with automatic named entity recognition. In: Proceedings of the 7th international EAMT workshop on MT and other language technology tools, improving MT through other language technology tools: resources and tools for building MT, Association for Computational Linguistics, pp 1–8
4. Toda H, Kataoka R (2005) A search result clustering method using informatively named entities. In: Proceedings of the 7th annual ACM international workshop on web information and data management, ACM, pp 81–86
5. Chopra D, Joshi N, Mathur I (2016) Named entity recognition in Hindi using hidden Markov model. In: 2016 second international conference on computational intelligence & communication technology (CICT), IEEE, pp 581–586
6. Chopra D, Jahan N, Morwal S (2016) Hindi named entity recognition by aggregating rule based heuristics and hidden Markov model. Int J Inf 2(6):43–52.
7. Ekbal A, Bandyopadhyay S (2010) Named entity recognition using support vector machine: a language independent approach. Int J Electr Comput Syst Eng 4(2):155–170
8. Ekbal A, Bandyopadhyay S (2009) A conditional random field approach for named entity recognition in Bengali and Hindi. Linguist Issues Lang Technol 2(1):1–44
9. Goller C, Kuchler A (1996) Learning task-dependent distributed representations by backpropagation through structure. In: Proceedings of international conference on neural networks (ICNN'96), vol 1, IEEE, pp 347–352
10. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
11. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
12. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: 1st international conference on learning representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, workshop track proceedings
13. dos Santos C, Guimaraes (2015) Boosting named entity recognition with neural character embeddings. In: Proceedings of the fifth named entity workshop, association for computational linguistics, Beijing, China, pp 25–33
14. Lafferty J, McCallum A, Pereira FC (2006) Conditional random fields: probabilistic models for segmenting and labeling sequence data
15. Basile P, Semeraro G, Cassotti P (2017) Bi-directional LSTM-CNNs-CRF for Italian sequence labeling, CLiC-it 2017 11-12 December 2017, Rome
16. Gregoric AZ, Bachrach Y, Coope S (2018) Named entity recognition with parallel recurrent neural networks. In: Proceedings of the 56th annual meeting of the association for computational linguistics (volume 2: short papers), pp 69–74
17. Lample G, Ballesteros M, Subramanian S, Kawakami K, Dyer C (2016) Neural architectures for named entity recognition. In: Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies, association for computational linguistics, San Diego, California, pp 260–270
18. P. V. Q. de Castro, N. F. F. da Silva, A. da Silva Soares, Portuguese Named Entity Recognition Using LSTM-CRF, in: International Conference on Computational Processing of the Portuguese Language, Springer, 83–92, 2018.
19. Misawa S, Taniguchi M, Miura Y, Ohkuma T (2017) 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, pp 97–102
20. Zhang Y, Yang J (2018) ChineseNER using lattice LSTM. In: ACL
21. Kaur Y, Kaur ER (2015) Named Entity Recognition (NER) system for Hindi language using combination of rule based approach and list look up approach. Int J Sci Res Manage 3 (3):2300–2306
22. Morwal S, Jahan N, Chopra D (2012) Named entity recognition using hidden Markov model (HMM). Int J Nat Lang Comput 1 (4):15–23
23. Gayen V, Sarkar K (2013) An HMM based named entity recognition system for indian languages: the JU System at ICON 2013, CoRR abs/1405.7397.
24. Saha SK, Sarkar S, Mitra P (2008) A hybrid feature set based maximum entropy Hindi named entity recognition. In: Proceedings of the third international joint conference on natural language processing, vol I, pp 343–349
25. Saha SK, Narayan S, Sarkar S, Mitra P (2010) A composite kernel for named entity recognition. Pattern Recogn Lett 31 (12):1591–1597
26. Devi GR, Veena P, Kumar MA, Soman K (2016) Entity extraction of Hindi–English and Tamil–English code-mixed social media text. In: Forum for information retrieval evaluation, Springer, pp 206–218
27. Singh V, Vijay D, Akhtar SS, Shrivastava M (2018) Named entity recognition for Hindi-English code-mixed social media text. In: Proceedings of the seventh named entities workshop, pp 27–35
28. Biswas S, Mishra M, Sitanathbiswas SA, Mohanty S (2010) A two stage language independent named entity recognition for Indian languages, IJCSIT). Int J Comput Sci Inf Technol 1 (4):285–289.
29. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12: 2493–2537
30. Chiu JP, Nichols E (2016) Named entity recognition with bidirectional LSTM-CNNs. Trans Assoc Comput Linguist 4:357–370

31. Ma X, Hovy E (2016) End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, pp 1064–1074
32. Athavale V, Bharadwaj S, Pamecha M, Prabhu A, Shrivastava M (2016) Towards deep learning in Hindi NER: an approach to tackle the labelled data sparsity. In: Proceedings of the 13th international conference on natural language processing, ICON 2016, Varanasi, India, December 17–20, 2016, pp 154–160
33. Gupta D, Ekbal A, Bhattacharyya P (2018) A deep neural network based approach for entity extraction in code-mixed indian social media text. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC-2018)
34. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), association for computational linguistics, Doha, Qatar, pp 1532–1543
35. Kanimozhi U, Ganapathy S, Manjula D, Kannan A (2019) An intelligent risk prediction system for breast cancer using fuzzy temporal rules. Natl Acad Sci Lett 42:227–232
36. Perumal S, Ganapathy S, Kannan A (2019) An intelligent fuzzy rule-based e-learning recommendation system for dynamic user interests. J Supercomput 75:5145–5160
37. Murthy R (2017) Named entity recognition using deep learning. In: 14th international conference on natural language processing. NLP Association of India (NLPAI)
38. Thangaramya K, Kulothugan K, Logambigai R, Selvi M, Ganapathy S, Kannan A (2019) Energy aware cluster and nero-fuzzy based routing algorithm for wireless sensor networks in IoT. Comput Netw 151:211–223
39. Ganapathy S, Kulothungan K, Muthurajkumar S et al (2013) Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. J Wirel Commun Netw 2013:271. https://doi.org/10.1186/1687-1499-2013-271
40. Sethukkarasi R, et al (2014) An intelligent neuro fuzzy temporal knowledge representation model for mining temporal patterns, pp 1167–1178.