International Journal of Cognitive Informatics and Natural Intelligence
Volume 15 • Issue 3 • July-September 2021

Entity-Extraction Using Hybrid Deep-Learning Approach for Hindi text

Richa Sharma, Banasthali Vidyapith, India
Sudha Morwal, Banasthali Vidyapith, India
Basant Agarwal, Indian Institute of Information Technology, Kota, India

ABSTRACT

This article presents a neural network-based approach to develop named entity recognition for Hindi text. In this paper, the authors propose a deep learning architecture based on convolutional neural network (CNN) and bi-directional long short-term memory (Bi-LSTM) neural network. Skip-gram approach of word2vec model is used in the proposed model to generate word vectors. In this research work, several deep learning models have been developed and evaluated as baseline systems such as recurrent neural network (RNN), long short-term memory (LSTM), Bi-LSTM. Furthermore, these baseline systems are promoted to a proposed model with the integration of CNN and conditional random field (CRF) layers. After a comparative analysis of results, it is verified that the performance of the proposed model (i.e., Bi-LSTM-CNN-CRF) is impressive. The proposed system achieves 61% precision, 56% recall, and 58% F-measure.

KEYWORDS

Convolutional Neural Network, Deep Learning, Distributed Representation, Feature Engineering, Machine Learning, Natural Language Processing, Neural Networks, Sequence Labeling

INTRODUCTION

Language is a necessary entity for communication. To easiness the way of human-computer interaction, it is preferred for machines to comprehend natural languages. In order to incorporate natural language understanding feature in machines, several Natural Language Processing (NLP) techniques are used. 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 have been developed for many natural languages. During the development of such NLP applications, a task namely Named Entity Recognition (NER) is often performed as a preprocessing step. NER is a two-step process which is used to identify proper nouns from text and classify them into predefined categories such as person, location, measure, organization, time, etc. Integration of NER in NLP applications increases the accuracy level of these applications. For example, question-answering system (Greenwood & Gaizauskas, 2003), machine-translation system (Babych & Hartley, 2003) and text clustering (Toda & Kataoka, 2005) performed well after integration of NER. Nowadays, these applications are also available for the Hindi language, therefore a state-of-art Hindi NER tool is essential to increase the performance of these NLP 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

DOI: 10.4018/IJCINI.20210701.oa1

This article, published as an Open Access article on April 23rd, 2021 in the gold Open Access journal, the International Journal of Cognitive Informatics and Natural Intelligence (converted to gold Open Access January 1st, 2021), is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

language experts to write rules. Furthermore, several machine learning based methods have been used to implement NER including hidden markov model (Chopra, Joshi, & Mathur, 2016; Morwal, Jahan, & Chopra, 2012), support vector machine (Ekbal & Bandyopadhyay, 2010), conditional random field (Ekbal & Bandyopadhyay, 2009) and combination of these methods. Again, these methods rely on an intensive knowledge of grammar and handcrafted features. Handcrafting of features are difficult due to 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, etc. which need to be addressed while developing NER tool for Hindi.

In order to handle these challenges, deep learning based models are highly appreciated. These models don't require feature engineering task instead of models learn features from raw data automatically. A deep learning based technique namely Recurrent Neural Networks (RNN) (Goller & Kuchler, 1996) have shown state-of-art results in several NLP applications. However, RNNs are unable to deal with long distance dependencies and lean towards 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, another deep learning based technique named as Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) can be used which is a variant of RNN. LSTMs have forget gate that helps to learn extremely long distance dependencies easily, thus resolves the issue of free word order sentences to a large extent. However, LSTMs consider only past context information and ignore future context. Whereas for NER, it has been observed that both past and future context information is helpful for better performance. Thus to access both left and right context, a variant of LSTM known as Bidirectional LSTM (Bi-LSTM) (Schuster & Paliwal, 1997) is used as a wrapper layer of LSTM. Another issue is a scarcity of labeled data in Hindi which can be resolved by learning features using pre-trained word embedding. This pre-trained word embedding is developed through the huge amount of unlabeled data. Word embedding is a vector representation of words in which words having the same meaning, have a similar representation (Mikolov, Chen, Corrado, & Dean, 2013). However, lone word embedding is unable to explore explicit character level information like prefix and suffix and these features could be very useful especially with rare words where word embedding are weakly trained. The character level features can be incorporated by applying a Convolutional Neural Network (CNN). Santos and Guimaraes (2015) have applied CNN to extract character level information in Spanish and Portuguese NER. A Conditional Random Field (CRF) (Lafferty, McCallum, & Pereira, 2001) approach can be 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 implementation of NER in many other languages such as Italian (Basile, Semeraro, & Cassotti, 2017), English (Gregoric, Bachrach, & Coope, 2018), Spanish (Lample, Ballesteros, Subramanian, Kawakami, & Dyer, 2016), Portuguese (de Castro, da Silva, & da Silva Soares, 2018), Japanese (Misawa, Taniguchi, Miura, & Ohkuma, 2017), Chinese (Zhang & Yang, 2018) etc. and have shown very good results. Research motive of this experiment is to observe 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 BiLSTM-CNN-CRF for implementation of Hindi NER.

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

RELATED WORK

Conventional Hindi NER systems can be classified as rule-based systems and machine learning based systems. First rule based system was developed by Y. Kaur and R. Kaur (2015) and worked for three new named entities named as money, direction and animal/bird. Since rule based approach is very

time consuming and requires language expert to design efficient rules, researchers were focusing on machine learning based approaches.

Several machine learning based NER models have been developed for Hindi. Hidden markov model based NER systems were developed by Morwal et al. (2012), Gayen and Sarkar (2014) and Chopra et al. (2016). Furthermore, maximum entropy based model was developed by Saha et al. (2008) with language specific features and gazetteers. Similarly, NER systems based on support vector machine were developed by Ekbal and Bandyopadhyay (2010) for Bangali and Hindi where lexical context patterns were used as features and by Saha et al. (2010) where novel kernel function was used. Another support vector machine based model was developed by Devi et al. (2016). Conditional random field based model was applied by Ekbal and Bandyopadhyay (2009) and Sigh et al. (2018) which used BIO standard format of tag where BIO stands for Beginning, Inside and Other tag. To increase the performance of NER system several hybrid systems (Biswas, Mishra, Sitanath_biswas, & Mohanty, 2010; Chopra, Jahan, & Morwal, 2012; Srivastava, Sanglikar, & Kothari, 2011) were developed by combing rule based and machine learning based approach. 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. In this direction, Collobert et al. (2011) designed a model for the English language in which features were learned from word embedding trained on a huge amount of unlabeled data. Chiu and Nichols (2016) proposed a model for English NER that combined Bi-LSTM with CNN to induce character level information in the model. Ma and Hovy (2016) designed a model for English NER which was a combination of three techniques of deep learning i.e. Bi-LSTM-CNN-CRF. In their model, CRF was integrated as a top layer to jointly decode labels for the whole sentence.

For Hindi NER systems, Athavale et al. (2016) developed a model in which pre-trained word embedding was used with Bi-LSTM layer and a softmax layer. Another deep learning based model developed by Gupta et al. (2018) which used a Gated Recurrent Unit (GRU) with character and word embedding layer and applied the model on code-mixed Indian social media text. Deep learning based models are not extensively explored for Hindi text whereas several state-of-art deep learning based NER models have been developed for other languages (Basile et al., 2017; Chiu & Nichols, 2016; Ma & Hovy, 2016; Misawa et al., 2017). Deep learning based models can be developed for the Hindi language also which may produce state-of-art results without the use of handcrafted features and any sort of linguistic resources.

PROPOSED APPROACH

Unlike previously developed Hindi NER systems, the proposed system doesn't perform any feature engineering task as deep neural networks automatically identify both word- and character-level features. Deep Learning is a set of machine learning algorithms that is stimulated by the artificial neural networks. Most of deep learning methods used neural networks. Artificial neural networks are mathematical modeling of structure and functionality of the brain. For feature extraction and transformation, deep learning applies several layers of nonlinear processing units. In which, output of the previous layer is given as input to next layer. In this manner, with each successive layer deep learning algorithms try to learn multiple levels of concept of increasing complexity/abstraction. For learning purpose, both supervised and unsupervised learning methodology are used in deep learning. Deep learning architectures comprised of several hidden layers where each layer participates in representing data in a better way. For this purpose, Word embedding is used to represent semantic information of words i.e. words with same meaning have similar representation. These word vectors are trained on unannotated corpus and also bridge the gap of the scarcity of labeled data to some extent. In this paper, an end to end neural network based model is proposed to implement NER. The proposed approach is demonstrated in figure 1.

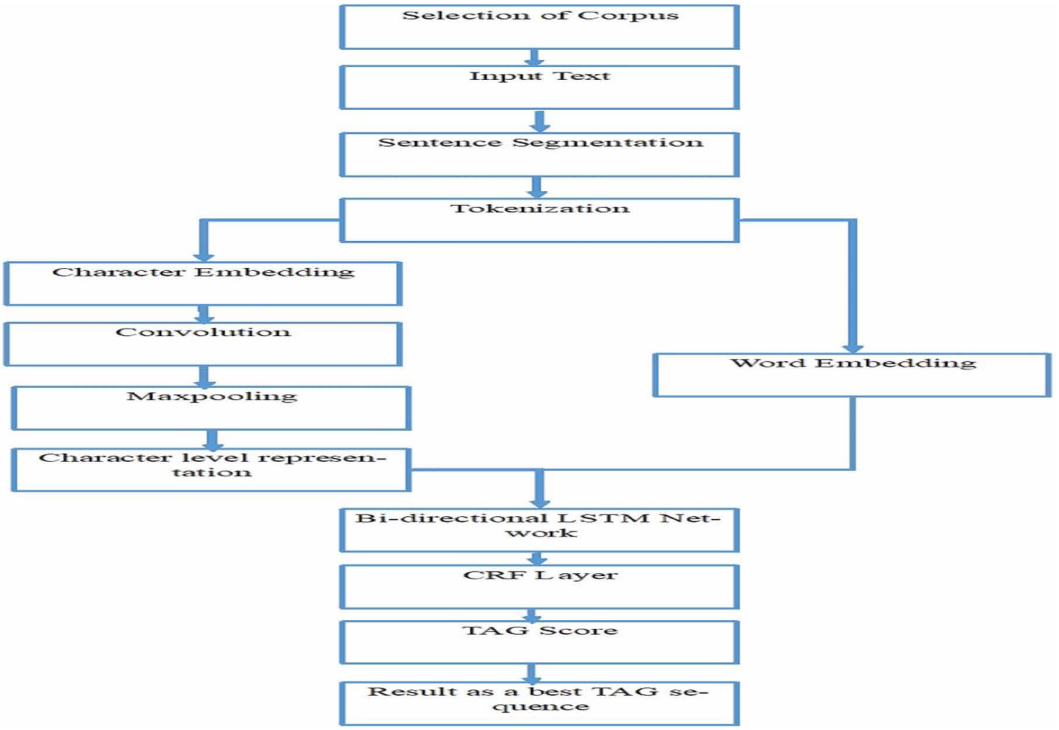
An artificial neural network is a web of artificial neurons whose functionalities almost simulate the human brain and its neurons. Unlike the human brain, artificial neurons are formed and controlled mathematically. An artificial neural network doesn't process words of text form instead of it requires words in vector form for input. For this purpose, each unique word of a sentence is represented by real-valued feature vector (word embedding) of dimension 'd'. This vector captures syntactic and semantic information of word. There are two ways of creating word vectors. First, word embedding can be created using pre-trained model. These models are trained on huge unlabeled corpus and generated word vectors for that corpus as an output. These word vectors can then be used as an embedding matrix to initialize the words of the training data of labeled corpus during the training of the model. Another way is to use pre-defined embedding layer with some distribution function to create word vectors and train them at the time of training. In this research work, word vectors are created with the help of a pre-trained model. To accomplish this purpose, word vectors are created and trained on an unlabeled corpus of 27,000 sentences of Hindi language, provided by Indian Language Technology Proliferation and Deployment Centre (TDIL) during this research work. For this training, skip-gram approach (Mikolov et al., 2013) of word2vec tool is applied which yielded word vectors as an output with the desired dimension. This pre-trained model is loaded with 48029 word vectors during this research work. Every word of the unlabeled corpus is now represented in vector form with 300 dimensions. Afterward, each sentence of training and test dataset is tokenized and each token is initialized with pre-trained word embedding vectors as shown in figure 1. To integrate character level features, each token is split into its associated characters and character embedding is formed for every character using the embedding layer. This character embedding is fed as an input to the convolution layer. After applying max-pooling operation character level representation is produced as shown in figure 1. Subsequently, both word embedding and character level representation is concatenated to form as one vector representation and fed as an input to bidirectional LSTM. Bi-LSTM appends the context level information to this vector and produces a tag sequence as a final outcome. This output is fed into the CRF layer to find out the best tag sequence for a sentence from all possible tag sequences. All experiments have been performed on NER tagged Hindi text corpus.

The layers of the proposed approach are described as follows:

CNN Layer

CNN is an effective approach to induce character level features during NLP tasks. As the name implies 'Convolutional neural network' is a special kind of network in which convolution operation is performed in place of general matrix multiplication in at least one of their layers. 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 figure 1. First, each sentence of the training dataset is tokenized and each token is divided into its corresponding characters. Embedding for each character is generated using the embedding layer with an output dimension of 'g' and with random initialization. In this work, each character is represented as 30 dimensional vector and a uniform distribution is used for initialization. This character embedding is used as input to the convolution layer as shown in figure 1. Before passing character embedding into the convolutional layer, a configured dropout operation is performed for regularization purpose. The proposed model performs convolution operation by using 30 filters where each filter has region size as 3 and strides as 1. For every filter, one feature map is produced as an output of convolution operation. The dimensionality of each feature map is a function of the filter's region size. Therefore, to produce a fixed-length vector, a 1-max-pooling operation is performed on each feature map which extracts a scalar value from each feature map. The dimension of this vector is the same as the number of filters. For example, it would be 30 in this work. This vector represents character-level information.

Figure 1. The network architecture of the proposed model



Bi-Directional LSTM Layer

As the name implies, Bi-directional LSTM layer combines two RNN layers moves in two opposite direction i.e. forward and backward through time. Forward RNN moves through time, from the start of the sequence towards the end of the sequence while backward RNN moves through time from the end of the sequence towards the start of the sequence. Thus, forward RNN preserves past information from the sequence whereas backward RNN preserves future information from the sequence. In this work, the bi-directional LSTM layer is applied which can handle variable-length input and can capture past and future context word information. The character-level representation computed by CNN and word embedding vector of a word is concatenated. The proposed system performs a configured dropout operation before concatenation of vectors for regularization purpose. This concatenated vector is fed into Bi-LSTM layer as its input to learn contextual information as shown in figure 1. In Bi-LSTM network, this input is processed in terms of two distinct hidden states to attain both past and future context information through forward and backward state respectively. Later, these two hidden states are concatenated and produce a vector as a final outcome of Bi-LSTM layer which contains contextual information of words in a sentence. A dropout operation is also performed on input and output of Bi-LSTM layer. Batch implementation is used to process multiple sentences at the same time.

CRF Layer

CRF layer is used as an output layer of the model as shown in figure 1. The motive of 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 a neural model, the output of the bidirectional LSTM layer is passed through a dense layer to generate tag scores. This input consists of the score of each tag for every word of a sentence

and is given to CRF layer. CRF process the input and finds a label sequence which has the highest prediction score for a sentence. This process can also be performed using a softmax layer however, CRF adds some constraints to final predicted tags and verifies their validity. These constraints are learned by CRF layer from 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 predicted tag sequence.

EXPERIMENTAL SETTINGS

All experiments including baseline and proposed model are performed on Keras 2.2.4 with Tensorflow backend support. The code for the implementation is written in Python 3.7.

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 is shown in Table 1. The training set contains 3776 sentences whereas test data have 702 sentences. A part of 409 sentences from training data is further separated for validation data. This corpus consists of 20 named entities which are categorized mainly into three categories such as ENAMEX, NUMEX, and TIMEX. ENAMEX contains 11 named entities such as person, entertainment, location, organization, materials, livthings, artifact, locomotive, plants, disease, and facilities. NUMEX contains 4 named entities such as distance, count, money and quantity. Similarly, TIMEX contains 5 named entities named as period, day, month, year and date. Those tokens which are not fit any of mentioned label tagged as 'O' i.e. others. In this research work, 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. For example हिमाचल/B-LOCATION प्रदेश/I-LOCATION. Table 1 shows statistics of tags in the corpus.

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 can be calculated as:

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

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

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

Table 1. Statistics of tags in corpus

Sr. 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.	Materials	105	31
6.	Livthings	179	29
7.	Artifact	202	8
8.	Locomotive	48	24
9.	Plants	43	12
10.	Disease	45	22
11.	Facilities	72	12
12.	Distance	185	18
13.	Count	127	39
14.	Money	34	17
15.	Quantity	45	12
16.	Period	92	8
17.	Year	77	2
18.	Month	126	29
19.	Day	5	3
20.	Date	20	1
21.	Others	59603	11053

Hyper-Parameter Settings

In this research work, hyper-parameters are estimated approximately. Several hyper-parameters such as number of epochs, batch size, number of LSTM units, number of filters, filter size, activation functions, dropout and optimizers are estimated. These parameters are tuned in a specific order. First, authors tune number of LSTM units with different values such as 50, 100, 200, 300 and 400. LSTM units with 200 perform better than other values of LSTM units. So, 200 units are used as LSTM dimensions which turned out to be 400 after applying a bi-directional wrapper layer. Afterward, epoch and batch size is tuned with different values and showed that the system performs well on 30 epoch with 64 batch size. Batch size in a range of power of 2 such as 32, 64 and 128, etc. should be used for performance reasons. Authors further tune hyper-parameters in a combination of activation function and optimizers with all other parameters remain same. Authors found that the system performs well with softmax activation function and rmsprop optimizer. However, in the case of CRF integrated models relu performs better than softmax. Lastly, models are tuned for regularization with different dropout rates. Dropout is a regularization method which is used to reduce overfitting problem occurs during the training of the model. Dropout is set as input and output of RNN and its variant. The system performs well on 0.6 dropout rate. For CNN integrated models, hyper-parameters such as the number of filters, the dimension of character embedding, region size are tuned. In this research work, 30 dimensional character embedding is used to represent characters in vector form. For convolution operation, 30 filters with 3 as region size and tanh as activation function is used.

Due to the lack of large labeled corpus, pre-trained word vectors are used to initialize word vectors of the training corpus. Here, pre-trained word vectors are generated on large unlabeled corpus using word2vec tool. Word2vec tool has two approaches to generate word embedding named as continuous bag-of-word (CBOW) and skip-gram. The skip-gram model learns the embedding by predicting the surrounding words for a given current word. Authors experiment with both approaches with different dimension values. Authors found that the system performs well with the skip-gram approach with 300 dimensions of vectors. The hyper-parameters used for this experiment are shown in Table 2.

Results

In this work, authors run experiments on three baseline models as RNN, LSTM and Bi-LSTM and then integrate these baseline models with CRF, CNN and CNN-CRF. Table 3 shows the results of all experiments. All these models are run with 300 dimensional word embedding and with same hyper-parameters. First, the system is run for three baseline models as RNN, LSTM and Bi-LSTM separately. As per results, Bi-LSTM outperforms both RNN and LSTM on all evaluation metrics. Bi-LSTM achieves 61% precision, 60% recall and 60% f-measure which is far better than RNN and LSTM as shown in Table 3. Subsequently, the CRF layer is integrated with all these three models to increase the efficiency of the system to decode the best tag sequence. Further, experiments are run on RNN-CRF, LSTM-CRF and Bi-LSTM-CRF models. Bi-LSTM-CRF achieves 60% f-measure which is greater than f-measure of both RNN-CRF and LSTM-CRF. Afterward, CNN layer is integrated to incorporate character level information with word embedding. Authors run further experiments on RNN-CNN, LSTM-CNN and Bi-LSTM-CNN models with same hyper-parameters. Results of these experiments are also shown in Table 3. As per results, the performance of Bi-LSTM-CNN is observed better than both RNN-CNN and LSTM-CNN for precision and f-measure. In this experiment, Bi-LSTM-CNN achieves 54% f-measure. However, this f-measure is less than individual Bi-LSTM model which is 60%. Finally, CRF layer is integrated with all three models i.e. RNN-CNN, LSTM-CNN and Bi-LSTM-CNN for jointly decoding of tag sequence instead of decoding each tag independently for a sentence. An experiment is run on RNN-CNN-CRF, LSTM-CNN-CRF and Bi-LSTM-CNN-CRF models with same hyper-parameters and obtains the results as shown in Table 3. The proposed approach i.e. Bi-LSTM-CNN-CRF outperforms the other two combinations RNN-CNN-CRF and LSTM-CNN-CRF on all evaluation metrics. Bi-LSTM-CNN-CRF model achieves 61% precision, 56% recall and 58% f-measure.

Table 2. Hyper-parameters for all experiments

Sr. No	Hyper-parameter	Value
1.	Character embedding dimension	30
2.	Word embedding dimension	300
3.	Filters	30
4.	Region size	3
5.	Dropout	0.6
6.	LSTM units	200
7.	Optimization	Rmsprop
8.	Epoch	30
9.	Batch size	64

CONCLUSION AND FUTURE WORK

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, time, etc. 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 for Hindi is proposed. The proposed model is an integration of bi-directional LSTM, CNN and CRF layer which identifies and classifies named entities from Hindi text. The performance of all models are impressive without the use of any linguistic resources, feature engineering and data preprocessing. The proposed system is also capable to handle several challenges introduced by the Hindi language while implementation of NER. The system achieves impressive results on Hindi NER while comparing with other state-of-art systems developed for Hindi NER. Since the size of available training and testing datasets are small, these datasets would be enhanced as a future task for better performance of the system.

Table 3. Results for the Hindi NER task

Sr. No	Model	Precision	Recall	F-measure
1.	RNN	47	54	50
2.	LSTM	48	54	51
3.	Bi-LSTM	61	60	60
4.	RNN-CRF	57	58	58
5.	LSTM-CRF	60	55	57
6.	Bi-LSTM-CRF	64	57	60
7.	RNN-CNN	45	44	45
8.	LSTM-CNN	45	49	47
9.	Bi-LSTM-CNN	62	47	54
10.	RNN-CNN-CRF	47	53	50
11.	LSTM-CNN-CRF	48	53	51
12.	Bi-LSTM-CNN-CRF	61	56	58

REFERENCES

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. *Proceedings of the 13th International Conference on Natural Language Processing, {ICON} 2016*, 154–160.

Babych, B., & Hartley, A. (2003). Improving machine translation quality with automatic named entity recognition. *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*, 1–8. doi:10.3115/1609822.1609823

Basile, P., Semeraro, G., & Cassotti, P. (2017). Bi-directional LSTM-CNNs-CRF for Italian Sequence Labeling. CLiC-It 2017, 18.

Biswas, S., Mishra, M. K., Sitanath-Biswas, S. A., & Mohanty, S. (2010). A two stage language independent named entity recognition for indian languages. *International Journal of Computer Science and Information Technologies, 1*(4), 285–289.

Chiu, J. P. C., & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357–370. doi:10.1162/tacl_a_00104

Chopra, D., Jahan, N., & Morwal, S. (2012). Hindi named entity recognition by aggregating rule based heuristics and hidden markov model. *International Journal of Information*, *2*(6), 43–52. doi:10.5121/ijist.2012.2604

Chopra, D., Joshi, N., & Mathur, I. (2016). Named Entity Recognition in Hindi Using Hidden Markov Model. *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, 581–586. doi:10.1109/CICT.2016.121

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, *12*(Aug), 2493–2537.

de Castro, P. V. Q., da Silva, N. F. F., & da Silva Soares, A. (2018). Portuguese Named Entity Recognition Using LSTM-CRF. *International Conference on Computational Processing of the Portuguese Language*, 83–92. doi:10.1007/978-3-319-99722-3_9

Devi, G. R., Veena, P. V, Kumar, M. A., & Soman, K. P. (2016). Entity Extraction of Hindi-English and Tamil-English Code-Mixed Social Media Text. *Forum for Information Retrieval Evaluation*, 206–218.

dos Santos, C., & Guimarães, V. (2015). Boosting Named Entity Recognition with Neural Character Embeddings. *Proceedings of the Fifth Named Entity Workshop*, 25–33. doi:10.18653/v1/W15-3904

Ekbal, A., & Bandyopadhyay, S. (2009). A conditional random field approach for named entity recognition in Bengali and Hindi. *Linguistic Issues in Language Technology*, *2*(1), 1–44.

Ekbal, A., & Bandyopadhyay, S. (2010). Named entity recognition using support vector machine: A language independent approach. *International Journal of Electrical. Computing Systems in Engineering*, *4*(2), 155–170.

Gayen, V., & Sarkar, K. (2014). *An HMM Based Named Entity Recognition System for Indian Languages: The JU System at ICON 2013.* CoRR, abs/1405.7

Goller, C., & Kuchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. *Proceedings of International Conference on Neural Networks (ICNN'96), 1*, 347–352. doi:10.1109/ICNN.1996.548916

Greenwood, M. A., & Gaizauskas, R. (2003). Using a named entity tagger to generalise surface matching text patterns for question answering. *Proceedings of the Workshop on Natural Language Processing for Question Answering (EACL03)*, 29–34.

Gregoric, A. Z., Bachrach, Y., & Coope, S. (2018). Named entity recognition with parallel recurrent neural networks. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics,* 2, 69–74.

Gupta, D., Ekbal, A., & Bhattacharyya, P. (2018). A Deep Neural Network based Approach for Entity Extraction in Code-Mixed Indian Social Media Text. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).*

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735 PMID:9377276

Kaur, Y., & Kaur, E. R. (2015). Named Entity Recognition (NER) system for Hindi language using combination of rule based approach and list look up approach. *International Journal of Scientific Research and Management*, *3*(3).

Lafferty, J., McCallum, A., & Pereira, F. C. N. (2001). *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. Academic Press.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural Architectures for Named Entity Recognition. *Proceedings of the 2016 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies*, 260–270.

Ma, X., & Hovy, E. (2016). End-to-end Sequence Labeling via Bi-directional {LSTM}-{CNN}s-{CRF}. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* 1, 1064–1074. doi:10.18653/v1/P16-1101

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *1st International Conference on Learning Representations, {ICLR} 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Misawa, S., Taniguchi, M., Miura, Y., & Ohkuma, T. (2017). Character-based Bidirectional LSTM-CRF with words and characters for Japanese Named Entity Recognition. *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, 97–102. doi:10.18653/v1/W17-4114

Morwal, S., Jahan, N., & Chopra, D. (2012). Named entity recognition using hidden Markov model (HMM). *International Journal on Natural Language Computing*, *1*(4), 15–23. doi:10.5121/ijnlc.2012.1402

Saha, S. K., Narayan, S., Sarkar, S., & Mitra, P. (2010). A composite kernel for named entity recognition. *Pattern Recognition Letters*, *31*(12), 1591–1597. doi:10.1016/j.patrec.2010.05.004

Saha, S. K., Sarkar, S., & Mitra, P. (2008). A hybrid feature set based maximum entropy Hindi named entity recognition. *Proceedings of the Third International Joint Conference on Natural Language Processing, 1*.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. doi:10.1109/78.650093

Singh, V., Vijay, D., Akhtar, S. S., & Shrivastava, M. (2018). Named entity recognition for hindi-english code-mixed social media text. *Proceedings of the Seventh Named Entities Workshop*, 27–35. doi:10.18653/v1/W18-2405

Srivastava, S., Sanglikar, M., & Kothari, D. C. (2011). Named entity recognition system for Hindi language: A hybrid approach. *International Journal of Computational Linguistics*, *2*(1), 10–23.

Toda, H., & Kataoka, R. (2005). A search result clustering method using informatively named entities. *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management*, 81–86. doi:10.1145/1097047.1097063

Zhang, Y., & Yang, J. (2018). *Chinese NER Using Lattice LSTM*. ACL.

Richa Sharma is pursuing PhD in computer science and engineering from the Banasthali University, India. Her research area includes natural language processing (NLP), machine learning and deep learning. She received her Master's in computer science with specialization in machine learning from Banasthali University. She has worked as an Assistant Professor in the department of computer science & engineering at Global Institute of Technology, Jaipur.