

Article

An Improved Word Representation for Deep Learning Based NER in Indian Languages

Ajees A P , Manju K and Sumam Mary Idicula

Department of Computer Science, Cochin University of Science and Technology, Kochi-682022, India; manju@mec.ac.in (M.K); sumam@cusat.ac.in (S.M.I.)

* Correspondence: ajeesap@cusat.ac.in; Tel.: +91-9061859697

Received: 27 April 2019; Accepted: 27 May 2019; Published: 30 May 2019



Abstract: Named Entity Recognition (NER) is the process of identifying the elementary units in a text document and classifying them into predefined categories such as person, location, organization and so forth. NER plays an important role in many Natural Language Processing applications like information retrieval, question answering, machine translation and so forth. Resolving the ambiguities of lexical items involved in a text document is a challenging task. NER in Indian languages is always a complex task due to their morphological richness and agglutinative nature. Even though different solutions were proposed for NER, it is still an unsolved problem. Traditional approaches to Named Entity Recognition were based on the application of hand-crafted features to classical machine learning techniques such as Hidden Markov Model (HMM), Support Vector Machine (SVM), Conditional Random Field (CRF) and so forth. But the introduction of deep learning techniques to the NER problem changed the scenario, where the state of art results have been achieved using deep learning architectures. In this paper, we address the problem of effective word representation for NER in Indian languages by capturing the syntactic, semantic and morphological information. We propose a deep learning based entity extraction system for Indian languages using a novel combined word representation, including character-level, word-level and affix-level embeddings. We have used ‘ARNEKT-IECSIL 2018’ shared data for training and testing. Our results highlight the improvement that we obtained over the existing pre-trained word representations.

Keywords: Named Entity Recognition; Bi-LSTM-CRF; Indian languages; affix embedding; character-based word composition; agglutinative languages

1. Introduction

The information available on the internet is increasing drastically. Annual growth in the number of Internet users is also increasing. Lots of texts and images are added to the internet every second. This information is stored on the web in an unstructured manner. Finding the relevant information from this unstructured data is very time-consuming. The importance of information extraction (IE), a sub-branch of Artificial Intelligence is worth mentioning at this point. Information Extraction transforms the unstructured text into a structured form that is convenient for machine level processing. IE plays important roles in information retrieval, question answering, summarization and so forth [1]. NER is one of the subdomains of IE, which originated in the sixth message understanding conference (MUC-6) [2].

Dayong Wu et al. reports that 60% of the total queries in search engines are named entities [3]. Hence identification of named entities from unstructured text has got significant attention in query processing. Question answering systems also make use of Named Entity Recognition. Answers to most of the questions containing the keyword ‘where’, points to an entity belonging to the class ‘location’. Hence, recognizing such entities from unstructured text is crucial for applications like

question answering systems [4]. Similarly, questions containing the keyword ‘who’ seek for an answer belonging to the class ‘person’. Building structured information from unstructured text is crucial for such applications. Geographical navigation systems employ NER systems to access information about nearby places. Such systems store information about named entities with their geographical coordinates in a database.

Most of the online publication sites hold millions of scholarly articles and review papers. There can be a substantial number of papers related to a single topic with slight variations. Retrieving the relevant articles from this plethora of information is a challenging task. Categorizing the articles based on entities present in them can produce good results and save the effort required for shortlisting the articles. Integrating a NER API to the online publication sites can accelerate this categorization process and hence the retrieval performance. NER techniques can also be incorporated in content recommendation systems where new contents and ideas of today’s world are recommended in an automated way. Netflix, which stands as a pillar example of success in content recommendation, creates wonders for the fortunes of a media company by making their platforms more attractive and event addictive. Customer support systems can make use of NER tools by categorizing the customer feedbacks/queries based on the entities present in them. Those feedbacks/queries are assigned to the relevant department within the organization for proper handling.

NER systems can be successfully deployed in various semantic processing pipelines of multimedia and social network data. Amato et al. have exploited NER modules to derive instances of relevant concepts from digitized multimedia documents [5]. These concepts were used to develop a knowledge management system which can facilitate easy document management and retrieval in e-government applications. Fantacci et al. have plugged the NER module in a social network based emergency management system [6]. Social networks, which play a major role in bidirectional information exchange, can be directly involved in crisis management for emergencies. Entities such as locations and disaster events were extracted from social media text and disseminated as emergency alerts to citizens located in emergency areas. Kokkinogenis et al. have deployed the NER module to identify the traffic-related messages retrieved from Twitter automatically [7]. These messages are used to perform the real-time sensing of traffic information. Such information can be further used for providing insights into the flaws of mobility networks.

The availability of local language enabled keywords and smartphones steer the increased use of social media platforms and conversational systems by the Indian language users. This leads to a drastic increase in the Indian language content over the web and is likely to continue at the same rate since the Indian language internet user base is growing at 18% as compared with 3% growth of English [8]. According to a study by Zamora, almost all the domains including e-tailing, digital classifieds, digital payments, online government services and so forth will be benefited by the support of their own local language by 2021 [9]. Due to the severe data sparsity problem faced by Indian languages, most of the earlier works in Indian languages were based on feature engineering over classical machine learning techniques. These features include POS features, gazetteer features, morphological features and so forth which are also challenging to acquire for resource-scarce languages. There comes the advantage of deep learning system which hardly demands hand-crafted features. Several works are reported for NER in Indian languages using deep learning techniques [10–15]. However, most of them are language-dependent and relying on external resources. Further, developing a language-independent and deep learning based framework for Indian languages is a challenging task. NER in Indian languages is still an open challenge as compared with English and other European languages. This challenge is constituted by a set of characteristics of Indian languages, namely ‘no concept of capitalization’, ‘no closed set vocabulary’, polysemy and ambiguity.

At the very beginning, NER was solved using hand-crafted rules, lexicon, ontologies and orthographic features [16–18]. Most of them were domain-specific and language-dependent models relying on hand-crafted rules and external resources. These systems were followed by machine learning techniques relying highly on hand-crafted features. Later, Collobert et al. (2011) came up

with a neural network based NER system with minimal feature engineering [19]. They did not depend on any domain-specific or language-specific resources like lexicons, ontologies, rules and so forth. However, the current trend is in building neural architectures for NER that combines character-level, sub-word level and word level embedding features in a domain-independent way [20–22].

While traditional techniques are more dependent on hand-crafted features, deep learning techniques rely on their key capability called hierarchical feature learning, where the higher level features are learned automatically from lower level features. However, one of the major questions that come at this point is how to effectively represent a word before feeding it into neural networks. Many researchers addressed this question through their works on NER, where they attempted different representations for words including character-level representation, word-level representation and sub-word level representation. The second question to be addressed at this point is how to handle the OOV (Out-of-vocabulary) problem. Since word embeddings can only be obtained for words available in the corpus used to build the word embedding model, such embedding vectors are not available for unknown words (OOV words) which are relatively common in NER problem. To handle this problem, character-based word compositional models, which build the vector representation of a word from its compositional characters, are introduced. Both convolutional neural networks [23] and BiLSTMs (Bidirectional Long Short Term Memory) [24] were applied for this purpose. In this paper, we further explore an effective representation for words in Indian languages through a combination of character-level, word-level and affix-level embeddings. In our approach, we augment the methodology used by Vikas Yadav et al. in 2018 [25]. The only difference is in the way in which character-based word vectors are generated and frequent affixes are identified for each language. In their work, they had used Bi-LSTM to generate character-based word embeddings. While the frequent affixes are identified from the training data used for NER. Unlike their system, we have used convolutional neural networks to create character-based word embeddings. Moreover, frequent affixes in a language are identified from a general corpus (unannotated), which is much bigger in size as compared with the training data size. Experimental results on Indian language NER showed that our novel word representation improved the existing benchmark results on the same dataset.

A lot of challenges are there in implementing automatic Named Entity Recognition system for Indian Languages. Languages like English and Spanish support capitalization feature in their scripts. But capitalization feature is not supported in Indian Languages. The morphological richness of the languages is also a problem in recognizing named entities. The rich morphology of Indian Languages allows the addition of suffixes and prefixes to morphemes, thereby adding meaningful context and semantics to words. Many words in Indian Languages are formed by the repeated addition of suffixes to their stems [26]. Case information attached to the nouns is also a problem in the identification of named entities. Inflectional property of Indian Languages results in the appearance of the same word in different forms. This leads to poor probabilities and sparse data problem in statistical methods. It is the most specific characteristic of Indian languages, which makes the problem of computation very hard. Most of the words in Indian Languages are agglutinated. Agglutination leads to the formation of new complex words which are difficult to handle. Non-availability of resources like standard datasets, POS taggers, morphological analyzers, dictionaries and so forth, are also a hindrance to the development of NER systems for Indian Languages.

Since Indian Languages are morphologically rich and agglutinative in nature, the morphological features of the languages seem to be essential for language processing applications. Moreover, the words in Indian languages contain more semantic information than the words in Western languages. For example, the phrase ‘to Rahul’ in English can be expressed by a single word ‘*raahulinu*’ in Malayalam. Similarly, the phrase ‘not like that’ in English can be expressed by a single word ‘*appadiyillay*’ in Tamil. Hence, acquiring maximum information from individual words can give better approximation about the semantic category of words. The same word can be expressed in different forms without any change in the semantic class of the word. For example, even though the words ‘*raamante*’ and ‘*raamanile*’ are having different forms, both of them belongs to the same semantic class

‘person’. Statistical methods, which are based on the frequency count of the elementary units (words) fail in this situation. In such cases, affix embeddings can shed some light towards the semantic category of words from the context of affixes in a particular sentence.

This paper is organized into six major sections. The second section briefly reviews the related work. A short description of the dataset used is given in Section 3. Section 4 explains the proposed method. Section 5 deals with experiments and results. The final section concludes the paper with some directions on future works.

2. Related Works

NER is one of the popular research areas in the Natural Language Processing (NLP) community. It is a sequence labelling problem where each token in a sequence is labelled with its corresponding semantic classes. Given a tokenized text, the objective is to identify different categories in the text such as location, organization, event and so forth. Different solutions have been proposed to extract and classify named entities in a text document. Each of them has its own strengths and weaknesses. They can be broadly classified into four namely-Dictionary based, Rule-based, machine learning based and hybrid methodologies [27]. In the dictionary-based approach, a finite set of named entities are stored in a lexicon which acts as a look-up table to identify the entities in a text document [28,29]. Rule-based systems employ hand-crafted rules for identifying named entities from unstructured text [30,31]. They are designed for specific domains like clinical text, medical reports and so forth and are specific to the languages for which the rules are written. Lots of rules are required for entity recognition. Moreover, they are not portable and robust. Such limitations motivated the researchers in this field to develop machine learning based approaches, where we can effectively avoid the language specific and domain-specific barriers.

Most of the researches in Named Entity Recognition is conducted using machine learning based approaches [32–34]. They are of two types- supervised and unsupervised. Supervised methods demand training data that has been annotated for a specific task. The annotation process is a tedious one which requires a lot of time and human effort [35]. Here each word in the tagged corpus is replaced by a set of features capable of representing the meaning of that word. The tags of the words act as supervisors to tune the model parameters, whereas unsupervised methods work in the absence of labelled data. There is no supervisor in the case of unsupervised learning. They try to learn representations from the data. These representations are later used for entity recognition tasks. The hybrid approach is a combination of two or more approaches to combine the strengths and avoid the weaknesses in those techniques [36]. They can be easily adapted to new domains. Ensemble classifiers are examples of the hybrid approach, where a combination of two or more classifiers is employed to overcome the weakness of each other.

HMM, CRF, SVM and so forth were the most common supervised learning algorithms employed for NER. In 2002, Reference [37] used an HMM-based NER system for English and reported 96.6% F-score on MUC-6 data. They employed different orthographic features, trigger words and words from gazetteers to identify the named entities. A comparative study between HMM and MEMM has been performed by Malouf [38]. He experimented with the impact on NER of different features like capitalization, word position, and so forth. His system reported an F-score of 73.66% on the Spanish CoNLL 2002 data set. However, the winners of CoNLL 2002 were Carreras et al., who used binary AdaBoost classifiers to identify the named entities [39]. They used various features like trigger words, capitalization, bag of words and so forth to depict binary relations and these relations were used to predict the entity labels. They obtained an overall F-score of 81.39% on the Spanish CoNLL 2002 data set.

Neural Architectures for NER have received attention since the work of Collobert et al. in 2011 [19]. They used pre-trained word embeddings over vanilla neural networks for different NLP tasks such as POS tagging, chunking, NER, and so forth. Later, pre-trained character embeddings were utilized in the same way for languages like Chinese [40,41]. Based on how the words are

represented in the network, neural architectures for NER can be broadly classified into word level, character-level and combination of the above two. Huang et al. [42] presented a word level neural architecture by stacking a Conditional Random Field (CRF) layer over BiLSTM layer and providing an embedded representation of words as input. They achieved considerable improvement in performance over Collobert's vanilla neural network model. Later, similar systems were applied to different domains such as biomedical NER [43], multilingual POS tagging [44] and drug NER [45] where similar improvements were observed. In character-level architectures, sentences are treated as sequences of characters. These sequences of characters, along with their corresponding labels, are passed through RNNs (Recurrent Neural Networks) during the training time. The label for each character, predicted during the testing time, is converted into word-level tags via post-processing. This approach was first highlighted by Kim et al. in 2016 [46]. They used highway networks above the convolutional networks on a sequence of characters. This representation is passed through an LSTM + Softmax layers to predict the final tag sequence. Character-based models were most successful in languages like Chinese where they have achieved near state of the art results [47–49]. Kuru et al. employed similar architecture for NER on seven different languages, including Spanish and Dutch [50]. They have used the Viterbi algorithm to convert the tag prediction over characters to word-level tags. Further, Ling et al. designed a character-level representation for words using RNNs [24]. These representations are provided to the Bi-LSTM layer to disambiguate the tag. This system was able to achieve the state of art results in many domains, including POS tagging and NER.

After exploring the word-level and character-level representations, researchers turned their attention towards the combinational representations where we can take the benefit of both the individual representations. Ma and Hovy proposed the first combinational model in 2016 [20]. They represented words as a combination of word embedding and character-level word compositional vector. This representation is passed over a Bi-LSTM CRF stack to predict the final tag sequence. This model reported an F-score of 91.21% on the CoNLL 2003 English data set. A similar network was implemented by Limpsopatham et al. in 2015, where they concatenated word embeddings with CNN over characters and orthographic features [51]. They attained an F-score of 65.89% on twitter NER data.

Lample et al. introduced an architecture which concatenates the pre-trained word embeddings with Bi-LSTMs over the characters of the word [52]. This representation is passed through a Bi-LSTM-CRF stack to predict the final label sequence. They obtained an F-score of 90.94% on CoNLL 2002 data. Later, Bharadwaj et al. added phoneme representations in addition to the word level and character-level features [22]. They also incorporated an attention mechanism over a sequence of characters in the word. This model became the state-of-the-art system (85.81%) for Spanish NER until 2018. Finally, the most similar architecture towards ours proposed by Yadav et al. in 2018 [25]. They combined affix-level features along with frequently explored word + character models. They considered the frequent affixes in the training data as the true morphemes of the language. These affixes were randomly initialized to embedding vectors and were learned during training. Their model reports a new benchmark on CoNLL Spanish, Dutch and German data sets.

Deep learning models were also explored for Indian languages. Nagesh Bhattu et al. reported a two-stage architecture utilizing the character-level and word-level features associated with words [53]. The second stage of their architecture makes use of a Bi-LSTM layer to disambiguate the tags. The model achieved an average accuracy of 97.45% on IECSIL (Information Extractor for Conversational Systems in Indian Languages) test data. Thenmozhi et al. reported a neural machine translation architecture to accomplish the task of NER [13]. Instead of using the pre-trained word embeddings, they used an embedding layer to create the vector representation of words. They achieved an overall accuracy of 96.11% on the same dataset. Sagar et al. implemented an end to end language-independent system for entity extraction [14]. They used pre-trained word embedding created using word2vec to obtain semantic information about words. The character-level features about words are generated using two successive convolutional layers. Finally, both the representations are concatenated to obtain the final word representation. This representation is passed through a Bi-LSTM network to generate the

required tag sequence. Khushleen [12] used a Bi-LSTM network with subword aware representation for words reported by Bojanowski et al. [54]. They could obtain an overall accuracy of 96.53% on IECSIL test data.

3. Dataset Details

We evaluate our system on ARNEKT-IECSIL 2018 shared data <https://github.com/BarathiGanesh-HB/ARNEKT-IECSIL> (accessed on 8 May 2012) [55]. ARNEKT-IECSIL was a track in FIRE-2018 with the objective of improving the information extraction systems on Indian languages. FIRE, Forum for Information Retrieval and Evaluation is a South Asian counterpart of TREC and CLEF started in the year 2008 intending to meet the challenges in multilingual information access [56]. The data set contains training and testing data for five Indian languages, namely Hindi, Tamil, Kannada, Telugu and Malayalam. The training data contain words and their corresponding labels in a two column format. The sentences are separated using a ‘newline’ keyword. Similarly, test data contains words without labels separated using ‘newline’ keyword between sentences. Except for Kannada, all the other language datasets contain a sufficient number of sentences (more than 60,000). The training data for each language contain nine tags including name, location, organization, event, things, occupation, number, date/time and other. Table 1 gives details about the dataset used.

Table 1. Data set details.

Dataset	Filetype	FileLength	# Sentences	# Words	# Unique Words
Hindi	Train	1,548,570	76,537	1,472,033	87,842
	Test	519,115	25,513	493,602	43,797
Kannada	Train	318,356	20,536	297,820	73,712
	Test	107,325	6846	100,479	34,200
Malayalam	Train	903,521	65,188	838,333	143,990
	Test	301,860	21,730	280,130	67,361
Tamil	Train	1,626,260	134,030	1,492,230	185,926
	Test	542,225	44,677	497,548	89,529
Telugu	Train	840,904	63,223	777,681	108,059
	Test	280,533	21,075	259,458	51,555

4. Proposed Method

Our objective is to build a deep learning-based entity extraction system for Indian languages. Traditional neural networks have shown outstanding performance over the last decade. Still, they have limitations. They are not up to the mark for capturing the sequential information, where the current state is affected by its previous states. They assume all its inputs and outputs as independent of each other. But if we want to predict the next word in a sequence, it is better to know the preceding words in the sequence. Sequence to sequence learning in deep learning is a promising solution for that. The sequence to sequence learning models try to find the optimum output sequence corresponding to the input sequence, as shown in Figure 1. Deep neural networks are powerful tools in various fields of NLP like language modelling, speech recognition, machine translation, caption generation and so forth. Hierarchical feature learning is the main characteristic of deep neural networks [57]. Higher level features are learned automatically from lower level features. The entire network can learn complex functions that can map the input to output directly. Through this paper, we also want to explore the effectiveness of affix embeddings for Indian language NER.

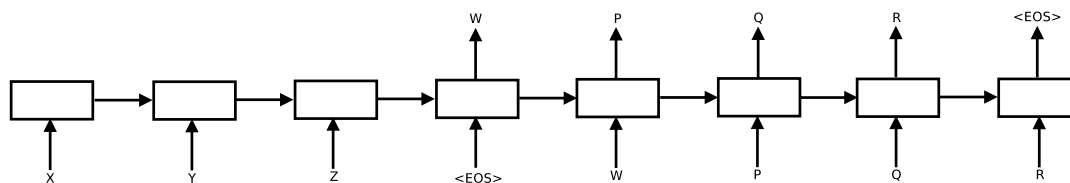


Figure 1. “XYZ” is the input sequence and “PQR” is the output sequence [58].

In Indian languages, a word often consists of various morphemes due to their agglutinative characteristics. Hence, it is not desirable to consider only the complete word as a processing unit. Instead, we consider affixes at both ends of the word as additional features for NER. Our base model is similar to Reference [25], where we apply the combined representation of character-level, word-level and affix-level features over Bi-LSTM-CRF stack. Our model differs from their architecture in how the character-level and affix-level features are generated. In their architecture, they have used an LSTM layer over the embedded representation of characters to generate the character-level word embedding. The affix-level embeddings are created by identifying the frequent affixes from the training data. But in our model, we have used convolutional neural networks with various kernel sizes to generate the character-level representation of words. Moreover, the affix-level features are generated from a general corpus (not labelled) quite bigger than the NER training corpus. Without loss of generality, the proposed word representation model can be straightforwardly applied for any agglutinative languages.

4.1. Generalized Word Representation

The pre-trained word embedding models may not contain word vectors for all the words present in the training data. Table 2 demonstrates this point, where it shows the presence percentage of different named entities in the training data over the FastText word embedding file. We observe an overall missing (OOV) of 4.17% words from training data. Here comes the necessity of a generalized word representation, which can effectively alleviate the OOV problem faced by the pre-trained word embedding models. Even though the OOV words are replaced by dummy vectors, the additional vector representations can give a clue about the actual class of the word.

Table 2. Presence percentage of the training set entities in FastText word embedding files.

Language	Event	Things	Org	Occupation	Name	Location	Other	Average Presence
Hindi	99.69	99.33	99.23	99.48	94.96	98.91	96.38	98.28
Kannada	98.85	97.11	96.85	96.92	89.17	96.94	89.4	95.03
Malayalam	94.86	96.65	97.17	95.72	90.71	96.52	86.14	93.96
Tamil	98.34	98.3	97.95	96.93	91.72	95.13	93.05	95.91
Telugu	98.9	99.16	98.72	98.72	83.65	99.15	93.48	95.96
Class Avg.	98.12	98.11	98.00	97.55	90.04	97.33	91.69	95.83

Figure 2 shows the generalized representation of a word using a compositional approach. This representation combines features from different aspects of a word, namely character-level, word-level and affix level. The character-level features are generated using Convolutional neural networks with various kernel sizes. The word-level features are identified using pre-trained word embeddings. And the last component is the affix-level feature corresponding to the most frequent affixes in a language. All these features are concatenated to form the final vector representation of a word.

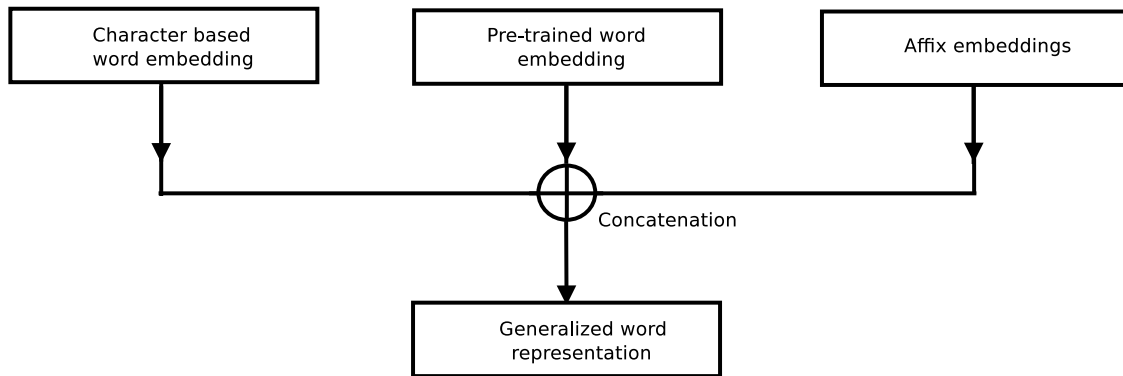


Figure 2. Generalized word representation.

4.2. Convnet Based Character Level Word Representation

Systems combining character-level features of the word, to the word-level features proved to be strong for NER [47,59,60]. The first successful work on character-based compositional word embedding was proposed by Dos Santos and Zadrozny in 2014 [23]. They used convolutional neural networks to constitute word vectors from character embeddings encoded by column vectors in an embedding matrix. Here, we augment the system developed by Ma et al. with a slight variation [20]. We have used convolutional kernels with various filter sizes (ranging from 1 to 6) to capture subword information. The length of meaningful subword units in Indian languages appears to be larger than that of the same in western languages. That is the reason why we decided to go with convolutional filters of various (comparably larger) kernel sizes. The concatenated output of different convolutional filters through max pooling layer act as the character-based word embedding vector. Figure 3 demonstrates the architecture of the character-based word vector generation model.

Given a word ‘w’ composed of ‘m’ characters $c_1, c_2, c_3, \dots, c_m$, where $c_i \in V_c$ is the character vocabulary set. Let $C_1, C_2, C_3, \dots, C_m$ be the character embedding vectors that encode the characters $c_1, c_2, c_3, \dots, c_m$ present in a word ‘w’. The character embeddings are obtained by matrix-vector product as given in Equation (1).

$$C_1 = W_c V_c \quad (1)$$

where W_c is the embedding matrix, $W_c \in R_{d_c \times |V_c|}$ and V_c is the one-hot vector representation of a particular character. ‘ d_c ’ is a hyperparameter corresponding to the size of the character embedding. Hence, each word is transformed into a sequence of character embeddings $C_1, C_2, C_3, \dots, C_m$. We apply convolutional kernels to each of the sliding context window of size k. The resulting vectors are passed through a max-pooling layer to generate the maximum value. We then concatenate these vectors from different convolutional kernels to produce the required word embedding vectors. These vectors are expected to capture information from different n-grams of the same word.

Formally, the character-level embedding of each word ‘w’ is calculated as follows,

$$V_c = \max_{1 \leq i \leq m} [W_{conv} Z_m + b_{conv}] \quad (2)$$

where W_{conv} and b_{conv} are parameters of the model and Z_m is the concatenation of character embeddings expressed as

$$Z_m = \left(C_{m-(k-1)/2}, \dots, C_{m+(k-1)/2} \right) \quad (3)$$

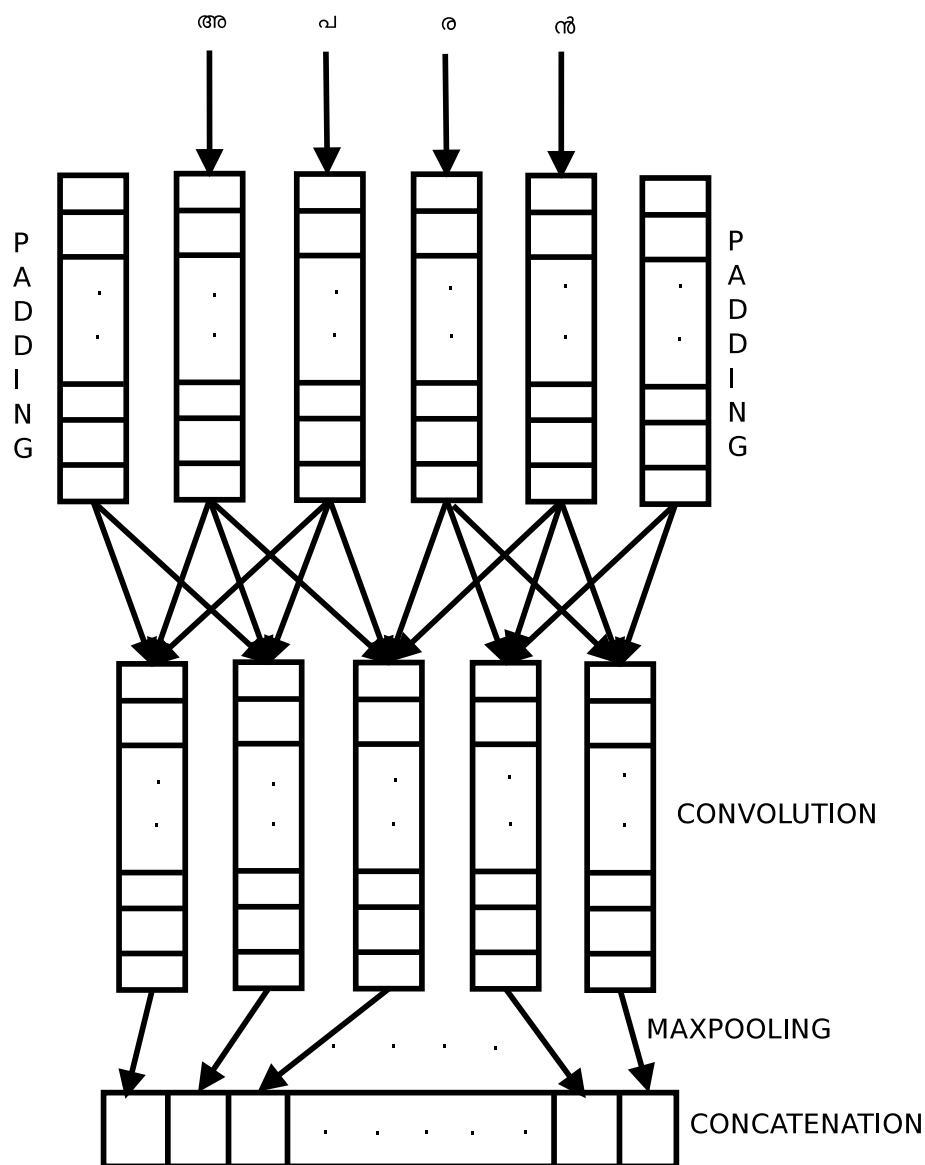


Figure 3. Architecture of the character-based word composition model.

The convolution operation is applied to find the simple patterns of embedding vectors of different n-grams over the character sequence. Among different n-grams the following maxpooling layer try to extract position invariant n-gram features. Therefore, the character-based word composition model is expected to detect unvarying local spelling features from the character sequence.

4.3. Pre-Trained Word Representation

Word embeddings are proven tools for capturing the context of a word along with its syntactic and semantic similarities. They are the representation of words in n-dimensional space. They are also well known for modelling the relationship with other words in a corpus. They are typically created by applying a large collection of unannotated text over shallow neural networks through an unsupervised process such as CBOW model. In our study, we have used FastText 300-dimension word embeddings for each language in the training set [61]. FastText, an extension to word2vec by Facebook, can efficiently represent words from their n-gram units. The embedding vector for a particular word is generated as the sum of all its n-gram vectors.

4.4. Affix Level Word Representation

To approximate the true affixes of a language, we identified the most frequent n-gram prefixes and suffixes of words in each language from an unannotated corpus (specific to each language). Since the most frequent n-gram affixes likely to behave like the true morphemes of the language, we decided to learn a task-specific representation for them. To identify the n-gram size and the threshold frequency of affixes, we experimented with various combinations of n (n-gram size) and T (threshold frequency) such as n = 2, 3, 4, 5 and T = 100, 300, 500, 750, 1000. The best results from the experiments were obtained, when the n-gram size was 3 and the threshold frequency was 500. Before training, the true affixes present in the training data are identified using a dictionary lookup method with the identified affixes (from unannotated corpus). During training time, the affix embeddings are initialized randomly and later tuned to learn a task-specific semantic representation. Finally, the individual representations are concatenated to construct a full vector representation for each word.

Formally, assume we are given a Malayalam sentence $S_{[1:n]}$ that is a sequence of n words, where 'n' is the maximum length of the sentence. In our case, we have limited it to 30 for the ease of computation. Each word ' w_i ' in the sentence is converted into a composition of vectors corresponding to that word. These vectors include character-level word composition vector, pre-trained word embedding vector and affix embedding vectors. Hence each word in the sentence is replaced by a vector of d-dimension, where $d = d_c + d_w + d_a$ such that d_c = dimensionality of character-level word embedding, d_w = dimensionality of pre-trained word embedding and $d_a = d_s + d_p$ is the dimensionality of affix embeddings, where d_s = dimensionality of suffix embedding and d_p = dimensionality of prefix embedding. Equations (4) and (5) show the mathematical representation of a single sentence.

$$S = [w_1, w_2, w_3, w_4, \dots, w_n] \quad (4)$$

$$S = [[v_{c1}, v_{c2}, v_{c3}, \dots, v_{cc}, v_{w1}, v_{w2}, v_{w3}, \dots, v_{ww}, v_{a1}, v_{a2}, v_{a3}, \dots, v_{as}]_1, \\ [v_{c1}, v_{c2}, v_{c3}, \dots, v_{cc}, v_{w1}, v_{w2}, v_{w3}, \dots, v_{ww}, v_{a1}, v_{a2}, v_{a3}, \dots, v_{as}]_2, \dots] \quad (5)$$

where $v_{c1}, v_{c2}, v_{c3}, \dots, v_{cc}$ corresponds to character-based word vector, $v_{w1}, v_{w2}, v_{w3}, \dots, v_{ww}$ corresponds to pretrained word vector and $v_{a1}, v_{a2}, v_{a3}, \dots, v_{as}$ corresponds to affix embedding vector.

Figure 4 demonstrates the complete architecture of our entity recognition model. The model accepts sequences of length 'n'. 'Wp', 'Wc', 'Wa' are pre-trained word embedding, character-based word embedding and affix embedding respectively. P0, P1, P2 and so forth, are the emission scores coming from the Bi-LSTM layer. Each of them indicates the probability of a particular tag for a particular word. Hence, 'm' corresponds to the maximum number of tags in the tag set. Final predictions are made by the CRF layer based on these probabilities.

We have used two Bi-LSTM layers to learn the input sequence since they are capable of learning long-term dependencies. Bi-LSTMs are a bidirectional variant of LSTMs. They are well known for their ability to capture all the past and future input features through two RNNs in forward and backward directions. RNNs are special kind of artificial neural networks designed to process sequences. LSTMs are a variant of RNN intended to stay away from the long-term dependency problem. Remembering information for long intervals of time is their default behaviour. They are designed by adding the special type of gates to RNNs. RNNs can use their contextual information when mapping between sequences. But the major drawback of RNN is that range of context in practice is quite limited. LSTM overcomes this drawback through a set of gates namely input gate, output gate and forgets gate. These gates help LSTM cells to store and access information over long periods.

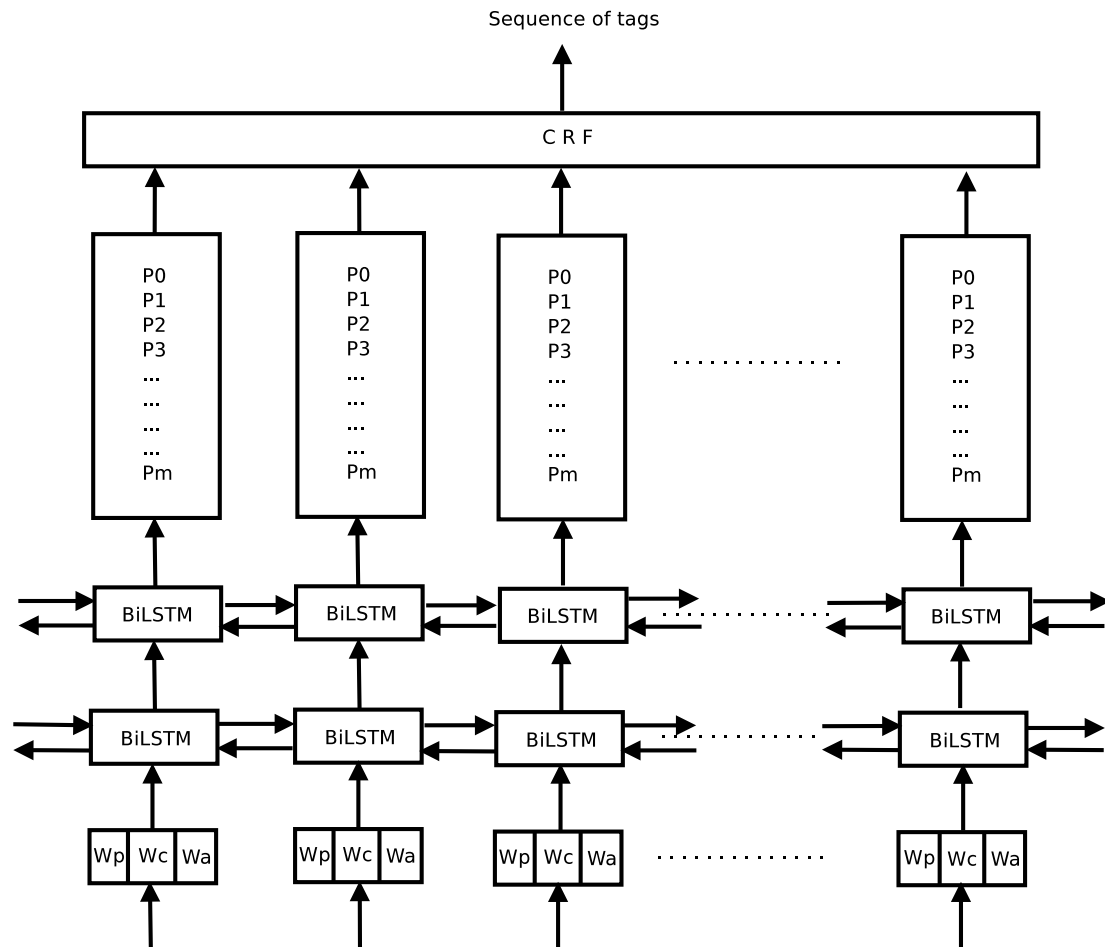


Figure 4. Architecture of the proposed system.

Both RNN and LSTM can preserve information in only one direction. But Bi-LSTM can preserve information from both the directions (front and back) simultaneously. The block diagram of a single layer of Bi-LSTM is shown in Figure 5. The Equations (6) and (7) calculate the forward function of Bi-LSTM.

$$a_h^t = \sum_{k=1}^K y_k^t W_{kh} + \sum_{h'=1, t>0}^H b_{h'}^{t-1} W_{h'h} \quad (6)$$

$$b_h^t = \Theta_h(a_h^t) \quad (7)$$

where y_t is a sequence input, a_t is the input to the LSTM unit h at time t , b_h is the activation function at time t , w_{kh} is the weight of the input k towards h . $w_{hh'}$ is the weight between the hidden layers h and h' . While the Equations (8) and (9) calculate the backward function of Bi-LSTM.

$$\frac{\delta O}{\delta W_{hl}} = \left(\sum_{t=1}^T \frac{\delta O}{\delta a_h^t} b_h^t \right) \quad (8)$$

$$\frac{\delta O}{\delta a_h^t} = \Theta_h(a_h^t) \left(\sum_{l=1}^L \frac{\delta O}{\delta a_h^l} W_{hl} + \sum_{h'=1, t>0}^H \frac{\delta O}{\delta a_{h'}^{t+1}} W_{hh'} \right) \quad (9)$$

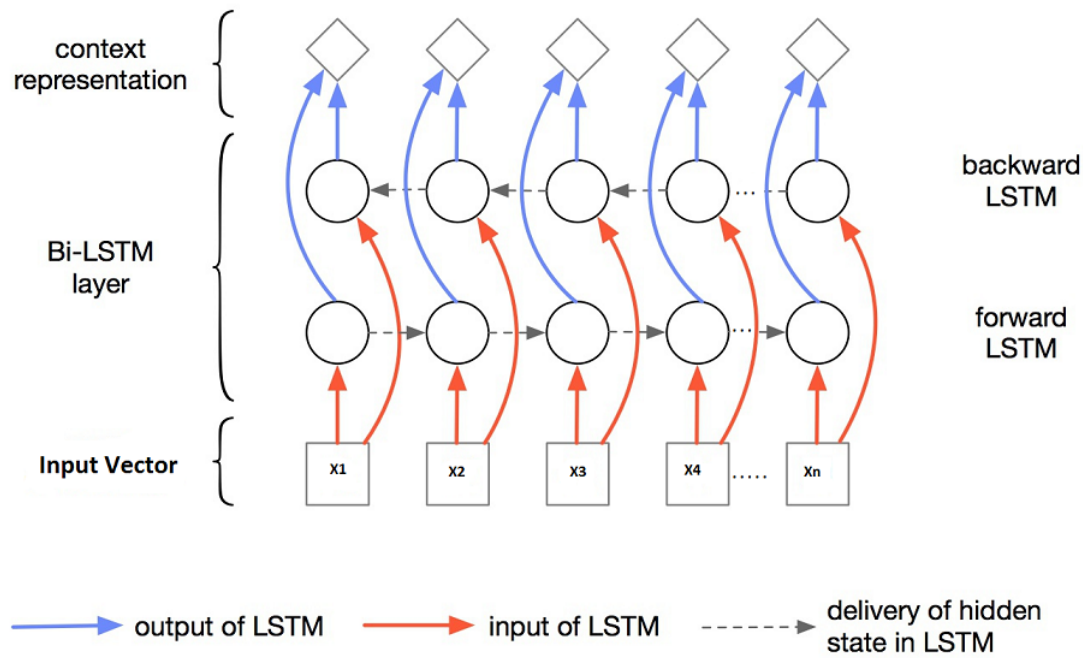


Figure 5. Bi-LSTM layer architecture.

Bi-LSTM-CRF, an extension to bidirectional LSTM models, can also model the dependency between the output layer labels in addition to the input feature dependencies. The CRF layer used here is to make the final predictions, given the probabilities for various tags for each word. The CRF layer can incorporate some additional constraints to the predicted probabilities from the Bi-LSTM layers. These constraints are learned by the CRF layer automatically from the training dataset during the training process. These constraints could be like the first tag in the sentence can't be a 'datenum', the last tag in the sentence can't be a 'organization', 'locations' do not follow 'numbers' and so forth. Hence CRF layer ensures the final predicted POS label sequences are valid. The loss function for CRF layer considers two types of scores namely-emission score and transition score. Emission score is the probability values coming from the Bi-LSTM layers and the transition score is taken from a probability matrix stored in CRF layer, which indicates the transition probability among different tags. These two scores are used in the calculation of path scores of different sentences. The calculation of path score for a complete sentence of length 5 is shown in Equations (10)–(12).

$$\begin{aligned} \text{EmissionScore} = & x_{0,START} + x_{1,name} + x_{2,other} \\ & + x_{3,location} + x_{4,other} + x_{5,other} + x_{6,END} \end{aligned} \quad (10)$$

$$\begin{aligned} \text{TransitionScore} = & t_{START \rightarrow name} + t_{name \rightarrow other} + \\ & t_{other \rightarrow location} + t_{location \rightarrow other} \\ & + t_{other \rightarrow other} + t_{other \rightarrow END} \end{aligned} \quad (11)$$

$$\text{PathScore} = \text{EmissionScore} + \text{TransitionScore} \quad (12)$$

Here ' x'_0 ' and ' x'_6 ' are the start and end markers which will be considered for all the sentences. The loss function calculates the real path score and the total score for all possible paths in the label sequences. The real path score is the score of the correct label sequence and the total score is the sum of all possible path scores for a particular sequence. During the training process, the parameters of the BiLSTM-CRF model will be updated again and again to minimize the following measurement.

$$\text{LossFunction} = \frac{P_{\text{RealPath}}}{P_1 + P_2 + \dots + P_N} \quad (13)$$

The entire Bi-LSTM-CRF network is constituted of three modules. The first module is the input module, which receives the vector representations of words as a concatenation of different features. A set of hidden layers constitutes the second module. The output of each hidden layer is provided to the successive hidden layer and finally to the output layer. ‘Tanh’ activation is used in the hidden layers. The calculation of ‘Tanh’ value for a particular input ‘x’ is shown in Equation (14). The output layer is a CRF layer which generates the maximum probable tag sequence corresponding to the word sequence. Dropout is used to prevent overfitting [62]. The network is coded in python using Tensorflow in the backend.

$$\begin{aligned} \tanh(x) &= (2\sigma(2x) - 1), \text{ where} \\ \sigma(2x) &= \frac{e^{2x}}{(1 + e^{2x})} \end{aligned} \quad (14)$$

5. Experiments and Results

Experiments were conducted to assess the effectiveness of different word representation models. Simply adding the pre-trained word vector to the target feature vector (word representation vector) during training was not promising. One solution suggested for this problem is adding character-based word embedding and affix embedding to the pre-trained word embedding. We adopt this solution by concatenating 150-dimensional character-based word embeddings and 60-dimensional affix embeddings to 300-dimensional pre-trained word vectors. To evaluate the proposed word representation model, we have employed the standard NER datasets provided by ARNEKT solutions. Training data for each language contains 80% of the total data. Evaluation is performed using the metric accuracy. Different word representation models and existing methods that we have used in our comparison are given below.

- Baseline: Since one of the best results in the shared data competition is reported by CRF, we considered it as the baseline.
- Pre-trained + Bi-LSTM-CRF: The method using pre-trained word embedding on Bi-LSTM-CRF model.
- Pre-trained + char_Convnet + Bi-LSTM-CRF: The technique using the concatenation of pre-trained word embedding and character-based word composition vector on Bi-LSTM-CRF model.
- Pre-trained + char_Convnet + affix embedding + Bi-LSTM-CRF: The method using the concatenation of pre-trained word embedding, character-based word embedding and affix embedding on Bi-LSTM-CRF model.
- Deep learning in the competition: The best reported deep learning methods (based on the results) from the competition.

To generate the pre-trained word embeddings, we have used FastText (<https://fasttext.cc/docs/en/crawl-vectors.html> (accessed on 8 May 2012)) embeddings corresponding to each language. To construct the character-based word composition vector, we fix the input size as 32 for each word as in Reference [63]. Six convolutional filters with kernel sizes of 1, 2, 3, 4, 5 and 7 were used. ‘ReLU’ was used to bring nonlinearity between the input vectors and response variables. ‘ReLU’ stands for Rectified Linear Units found to be the best activation in our case because they can alleviate the vanishing gradient problem, a matter in which lower layers of the network train very slowly due to the exponential diminishing of gradient through the lower layers. Each convolutional filter provides an output vector of 25 dimensions. Hence, the total size of the character-based word compositional vector is 150.

To generate the affix embeddings, we randomly initialized the prefix and suffix embedding matrix. Both prefix and suffix embedding sizes are fixed as 30 and allowed to learn during the Bi-LSTM-CRF training process. The remaining hyper-parameter settings are given as follows:

- Bi-LSTM hidden size = 300
- Maximum number of epochs = 100
- early stopping = 30
- drop_out = 0.5
- Pre-trained word embedding size = 300
- Optimizer = Adam
- Batch size = 100
- Initial learning = 10^{-4}
- Total word representation size = 510.

Keras functional API is used to build the seq2seq network since they can build complex models. We evaluated our model on all languages available in the training set. Our system achieved state-of-the-art performance on all languages, outperforming Bhattu et al. by 0.76% on overall accuracy [53]. Table 3 shows the comparison of performance between the proposed word representation model (in bold style) over the existing methods (methods from the competition). This includes the results from the deep learning based systems reported in the competition as well as the baseline CRF method. Combining the character-based and affix-based embeddings to the pre-trained embedding vector seems to be beneficial since they can capture different components of word similarities. The baseline CRF system reported an overall accuracy of 97.44% without any dictionary features. The proposed system could improve the overall accuracy by 0.77% over the baseline system [64]. This demonstrates the power of combined word representation over the individual ones. Figure 6 depicts the class-wise accuracy scores for all the languages in the testing data. The accuracy rate of ‘number’ class for all the languages is near to 100% due to the digit recognizing capability of the network. On the other hand, the ‘datenum’ class for most of the languages get a lot of misclassification due to the same reason, where most of them get classified into ‘number’ class. Hence, a suitable strategy should be adapted to recognize the ‘datenum’ class.

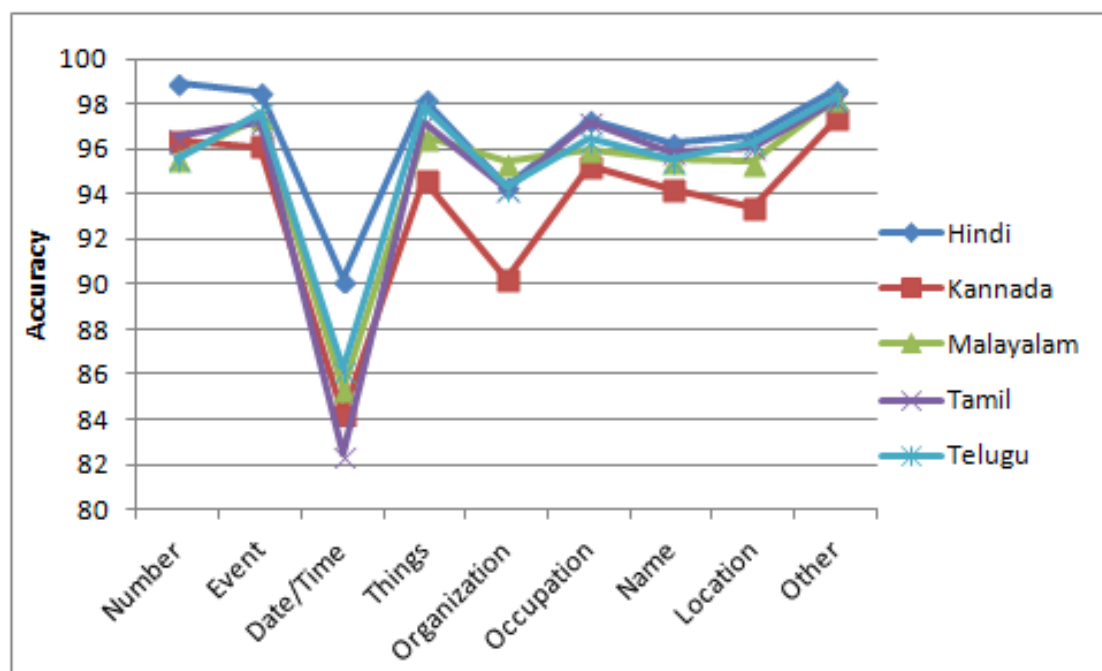


Figure 6. Class-wise accuracy scores for all the languages present in the shared data.

Table 3. Performance (Accuracy %) of our model in comparison with the deep learning results from the competition.

Model	Hindi	Kannada	Malayalam	Tamil	Telugu	Average
Baseline (CRF) [64]	97.67	97.03	97.44	97.36	97.72	97.44
Bhattu et al. [53]	97.82	97.04	97.46	97.41	97.54	97.45
Khushleen Kaur [12]	96.84	96.38	96.64	96.15	96.63	96.53
Thenmozhi et al. [13]	96.73	95.63	95.87	95.55	96.77	96.11
Sagar et al. [14]	94.44	92.94	92.92	92.48	92.42	93.04
Gupta et al. [15]	91.52	92.14	90.27	87.72	90.02	90.33
Our model	98.44	97.62	98.25	98.35	98.41	98.21

5.1. Impact of Character-Based Word Embedding

In order to assess the effectiveness of the proposed combined word representations, we compared the same system on different individual word representations. Combining the character-level word composition vector to pre-trained word vector seems to capture character-level features of the word in addition to word-level features. Table 4 presents the impact of different word representations on Bi-LSTM-CRF tagger. This includes the performance of the tagging model on individual word representations and combined word representations (in bold style). The character-based word representations could improve the accuracy of word-based models by 1.12%. This indicates the significance of character-based word representations.

Table 4. Impact of different word representations on BiLSTM-CRF tagger Accuracy(%).

Representation	Hindi	Kannada	Malayalam	Tamil	Telugu	Average
FastText	96.87	96.41	96.68	96.22	96.66	96.57
FastText+char_ConvNet	97.98	97.30	97.76	97.61	97.84	97.69
FastText+char_ConvNet+Affix	98.44	97.62	98.25	98.35	98.41	98.21

5.2. Impact of Affix Embeddings

Since Indian languages are morphologically rich, the use of affix-level features seems to be very effective in improving the overall performance of NER systems. Our analysis implies that appending affix-level features to the word-level features can capture the inflectional characteristics of agglutinative words. From Table 4, it is clear that affix-based features can improve the overall performance. When added to the word-level features, the affix-level features could improve the results by 0.52%. An example sentence tagged using the proposed Bi-LSTM-CRF model is shown in Figures 7 and 8. Here the correct tag for the second word is 'location', which is obtained only after incorporating the affix-level features to word-level features (Figure 8).

രാമൻ\name ആലുവയിൽ\other കലാണത്തിൻ\other പോയി\other .\other

Figure 7. Sample tagged text without incorporating the affix-level features.

രാമൻ\name ആലുവയിൽ\location കലാണത്തിൻ\other പോയി\other .\other

Figure 8. Sample tagged text by incorporating the affix-level features.

5.3. Impact of Training Data Size

Training data size has a considerable impact on improving the performance of NER models. Experiments were conducted with various sizes of training data on different word representations. We obtained different results with different sizes of the training data. The performance of the model gets increases with the increase in training data size. Table 5 illustrates the performance of various word

representations on different sizes of the training data. Here at first, the complete training data (80% of the total data) is used for training using different word representations. Afterwards, 20% reduction is made to each of the training data files and repeated the same experiments.

Table 5. Comparing the performance of different word representations on Indian Named Entity Recognition (NER) tasks by varying the training data size.

Training Data Size	Representation	Average Accuracy (%)
Complete (100%)	FastText	96.57
	FastText+char_ConvNet	97.69
	FastText+char_ConvNet+Affix	98.21
80%	FastText	95.83
	FastText+char_ConvNet	96.87
	FastText+char_ConvNet+Affix	97.51
60%	FastText	95.61
	FastText+char_ConvNet	96.70
	FastText+char_ConvNet+Affix	97.31
40%	FastText	95.29
	FastText+char_ConvNet	96.44
	FastText+char_ConvNet+Affix	96.98

5.4. Analysis

In this section, we present a qualitative analysis of the results from the NER system that use affix embedding features. We observed various instances where the affix-level features improving NER performance. Significant improvements were observed for Malayalam and Tamil, where the agglutination of words is more severe. By identifying the frequent affixes in a language, we are better approximating the true affixes of the language. This should benefit the tag disambiguation layer (Bi-LSTM-CRF) to learn the named entity characteristics effectively. Moreover, identifying the frequent affixes from an unannotated corpus (big) yields a better vocabulary of affixes in a particular language. Experiments were also conducted to determine whether the performance improvements were truly due to affix-level features. We changed the character-based word embedding size from 150 to 210 to match with the total word representation size in the absence of affix embedding. This model failed to score as much as the proposed combined word representation model.

Further, the effectiveness of filtering affixes from a general corpus rather than the training corpus is also verified. We ran our model with frequent affixes selected from the training set as well as those from a general corpus (unannotated big in size). The performance of the tagger without using the general corpus (97.98%) was even lower than the best-recorded results (98.21%) but still giving an improvement over baseline systems (97.44%). Hindi, which is the official language of India, gives the best results in all the approaches due to the availability of more training data as compared with other languages.

6. Conclusions

In this article, we have discussed a deep learning based Named Entity Recognition system for Indian languages. The exclusive feature of the proposed system is the use of affix embeddings towards the efficient representation of words as input features. The morphological features of the languages are effectively utilized in this study by identifying the frequent affixes as well as character-level features. Incorporating affix-level features in the entity extraction task appears to be helpful for Indian languages. Our experiments show that the affix-level features are complementary to character-level and word-level features in the NER task. The choice of threshold frequency and n-gram size are crucial, since including the complete set of affixes from the training data set did not improve the performance. Without using any form of external resources (like lexicon, POS feature, n-grams feature, etc.), our system was able to achieve state-of-art performance in Indian language NER. Since the proposed

word representation model can efficiently represent the syntactic, semantic and orthographic features of words, we hope to apply this method to represent words for other NLP tasks in low resource languages. While our model proposes a simple idea of extracting frequent affixes of a language as an approximate set of affixes of the language, we advice further improvements using some automated neural mechanisms. Even though our experiments were mainly focussed on Indian languages, this approach could be applied for any morphologically rich language.

Author Contributions: Formal analysis, M.K.; Investigation, M.K; Methodology, A.A.P.; Project administration, A.A.P. and S.M.I.; Software, A.A.P.; Supervision, S.M.I.; Validation, M.K; Writing—original draft, A.A.P. and S.M.I.

Funding: The first author of this paper would like to thank University Grants Commission for providing fellowship through UGC NET/JRF scheme.

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

References

1. Patil, N.; Patil, A.S.; Pawar, B. Survey of named entity recognition systems with respect to Indian and foreign languages. *Int. J. Comput. Appl.* **2016**, *134*, 21–26. [\[CrossRef\]](#)
2. Bindu, M.; Idicula, S.M. Named Entity Identifier for Malayalam Using Linguistic Principles Employing Statistical Methods. *Int. J. Comput. Sci. Issues* **2011**, *8*, 185.
3. Wu, D.; Zhang, Y.; Zhao, S.; Liu, T. Identification of web query intent based on query text and web knowledge. In Proceedings of the 2010 First International Conference on Pervasive Computing, Signal Processing and Applications, Harbin, China, 17–19 September 2010; pp. 128–131.
4. Etaiwi, W.; Awajan, A.; Suleiman, D. Statistical Arabic Name Entity Recognition Approaches: A Survey. *Procedia Comput. Sci.* **2017**, *113*, 57–64. [\[CrossRef\]](#)
5. Amato, F.; Colace, F.; Greco, L.; Moscato, V.; Picariello, A. Semantic processing of multimedia data for e-government applications. *J. Vis. Lang. Comput.* **2016**, *32*, 35–41. [\[CrossRef\]](#)
6. Fantacci, R.; Gei, F.; Marabissi, D.; Micciullo, L. The Use of Social Networks in Emergency Management. In *Wireless Public Safety Networks 2*; Elsevier: Amsterdam, The Netherlands, 2016; pp. 25–61.
7. Kokkinoginis, Z.; Filguieras, J.; Carvalho, S.; Sarmiento, L.; Rossetti, R.J. Mobility network evaluation in the user perspective: Real-time sensing of traffic information in twitter messages. In *Advances in Artificial Transportation Systems and Simulation*; Elsevier: Amsterdam, The Netherlands, 2015; pp. 219–234.
8. Barathi Ganesh, H.; Soman, K.; Reshma, U.; Mandar, K.; Prachi, M.; Gouri, K.; Anitha, K.; Anand Kumar, M. Overview of arnekt iecsil at fire-2018 track on information extraction for conversational systems in Indian languages. In Proceedings of the Proceedings of the 10th annual meeting of the Forum for Information Retrieval Evaluation, Gandhinagar, India, 6–9 December 2018; pp. 18–20.
9. Zamora, J. Rise of the chatbots: Finding a place for artificial intelligence in India and US. In Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion, Limassol, Cyprus, 13–16 March 2017; pp. 109–112.
10. Murthy, R.; Khapra, M.M.; Bhattacharyya, P. Improving NER Tagging Performance in Low-Resource Languages via Multilingual Learning. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2018**, *18*, 9. [\[CrossRef\]](#)
11. Murthy, V.R.; Bhattacharyya, P. A deep learning solution to Named Entity Recognition. In *International Conference on Intelligent Text Processing and Computational Linguistics*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 427–438.
12. Kaur, K. Khushleen@IECSIL-FIRE-2018: Indic Language Named Entity Recognition Using BidirectionalLSTMs with Subword Information. In Proceedings of the Proceedings of the 10th annual meeting of the Forum for Information Retrieval Evaluation, Gandhinagar, India, 6–9 December 2018.
13. Thenmozhi, D.; Kumar, B.S.; Aravindan, C. *SSN_NLP@ IECSIL-FIRE-2018: Deep Learning Approach to Named Entity Recognition and Relation Extraction for Conversational Systems in Indian Languages*; Department of CSE, SSN College of Engineering: Chennai, India, 2018.
14. Sagar, S.P.; Gollakota, R.K.; Das, A. *HiLT@ IECSIL-FIRE-2018: A Named Entity Recognition System for Indian Languages*; Indian Institute of Information Technology: Sri City, India, 2018.

15. Gupta, A.; Ayyar, M.; Singh, A.K.; Shah, R.R. raiden11@ IECSIL-FIRE-2018: Named Entity Recognition For Indian Languages. In Proceedings of the Proceedings of the 10th annual meeting of the Forum for Information Retrieval Evaluation, Gandhinagar, India, 6–9 December 2018.
16. Segura Bedmar, I.; Martínez, P.; Herrero Zazo, M. Semeval-2013 Task 9: Extraction of Drug-Drug Interactions from Biomedical Texts (Ddiextraction 2013). In Proceedings of the Association for Computational Linguistics (ACL), Sofia, Bulgaria, 4–9 August 2013.
17. Bossy, R.; Golik, W.; Ratkovic, Z.; Bessièrès, P.; Nédellec, C. Bionlp shared task 2013—An overview of the bacteria biotope task. In Proceedings of the BioNLP Shared Task 2013 Workshop, Sofia, Bulgaria, 9 August 2013; pp. 161–169.
18. Uzuner, Ö.; South, B.R.; Shen, S.; DuVall, S.L. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *J. Am. Med. Inf. Assoc.* **2011**, *18*, 552–556. [CrossRef]
19. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
20. Ma, X.; Hovy, E. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv* **2016**, arXiv:1603.01354.
21. Santos, C.N.D.; Guimaraes, V. Boosting named entity recognition with neural character embeddings. *arXiv* **2015**, arXiv:1505.05008.
22. Bharadwaj, A.; Mortensen, D.; Dyer, C.; Carbonell, J. Phonologically aware neural model for named entity recognition in low resource transfer settings. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 1462–1472.
23. Santos, C.D.; Zadrozny, B. Learning character-level representations for part-of-speech tagging. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), Beijing, China, 21–26 June 2014; pp. 1818–1826.
24. Ling, W.; Luís, T.; Marujo, L.; Astudillo, R.F.; Amir, S.; Dyer, C.; Black, A.W.; Trancoso, I. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv* **2015**, arXiv:1508.02096.
25. Yadav, V.; Sharp, R.; Bethard, S. Deep affix features improve neural named entity recognizers. In Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, New Orleans, LA, USA, 5–6 June 2018; pp. 167–172.
26. Nair, R.S.S. A Grammar of Malayalam. Available online: <http://www.languageinindia.com/nov2012/ravisankarmalayalamgrammar.pdf> (accessed on 8 May 2012). (In India)
27. Hamada, A.; Nayel, H.L.S. Improvin NER for Clinical Texts by Ensemble Approach using Segment Representations. In Proceedings of ICON 2017(NLP AI), Calcutta, India, 18–21 December 2017; pp. 197–204.
28. Cohen, W.W.; Sarawagi, S. Exploiting dictionaries in named entity extraction: Combining semi-Markov extraction processes and data integration methods. In Proceedings of the Tenth Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22–25 August 2004; pp. 89–98.
29. Wang, X.; Jiang, X.; Liu, M.; He, T.; Hu, X. Bacterial named entity recognition based on dictionary and conditional random field. In Proceedings of the 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Kansas City, MO, USA, 13–16 November 2017; pp. 439–444.
30. Eftimov, T.; Seljak, B.K.; Korošec, P. A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. *PLoS ONE* **2017**, *12*, e0179488. [CrossRef]
31. Alfred, R.; Leong, L.C.; On, C.K.; Anthony, P.; Fun, T.S.; Razali, M.N.B.; Hijazi, M.H.A. A rule-based named-entity recognition for malay articles. In Proceedings of the International Conference on Advanced Data Mining and Applications, Hangzhou, China, 14–16 December 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 288–299.
32. Wu, Y.; Jiang, M.; Xu, J.; Zhi, D.; Xu, H. Clinical Named Entity Recognition Using Deep Learning Models. In Proceedings of the AMIA Annual Symposium Proceedings, Washington, DC, USA, 4–8 November 2017.
33. Salini, A.; Jeyapriya, U. Named Entity Recognition Using Machine Learning Approaches. *arXiv* **2003**, arXiv:cs/0306050.
34. Zhang, L.; Pan, Y.; Zhang, T. Focused named entity recognition using machine learning. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, Sheffield, UK, 25–29 July 2004; ACM: New York, NY, USA, 2004; pp. 281–288.
35. Sienčnik, S.K. Adapting word2vec to named entity recognition. In Proceedings of the 20th Nordic Conference of Computational Linguistics, Nodalida 2015, Vilnius, Lithuania, 11–13 May 2015; pp. 239–243.

36. Nita Patil, Ajay S Patil, B.P. HYbrid Approach for Marathi Named Entity Recognition. In Proceedings of the ICON 2017(NLPAl), Calcutta, India, 18–21 December 2017; pp. 103–111.
37. Zhou, G.; Su, J. Named entity recognition using an HMM-based chunk tagger. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 473–480.
38. Malouf, R. Markov models for language-independent named entity recognition. In Proceedings of the COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002), Stroudsburg, PA, USA, 31 August 2002.
39. Carreras, X.; Màrquez, L.; Padró, L. Named entity extraction using adaboost. In Proceedings of the 6th Conference on Natural Language Learning 2002 (CoNLL-2002) 2002, Stroudsburg, PA, USA, 31 August 2002.
40. Li, Y.; Li, W.; Sun, F.; Li, S. Component-enhanced chinese character embeddings. *arXiv* **2015**, arXiv:1508.06669.
41. Yin, R.; Wang, Q.; Li, P.; Li, R.; Wang, B. Multi-granularity chinese word embedding. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 981–986.
42. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *arXiv* **2015**, arXiv:1508.01991.
43. Chalapathy, R.; Borzeshi, E.Z.; Piccardi, M. Bidirectional LSTM-CRF for clinical concept extraction. *arXiv* **2016**, arXiv:1611.08373.
44. Plank, B.; Søgaard, A.; Goldberg, Y. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv* **2016**, arXiv:1604.05529.
45. Xu, K.; Zhou, Z.; Hao, T.; Liu, W. A bidirectional LSTM and conditional random fields approach to medical named entity recognition. In Proceedings of the International Conference on Advanced Intelligent Systems and Informatics, Cairo, Egypt, 9–11 September 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 355–365.
46. Kim, Y.; Jernite, Y.; Sontag, D.; Rush, A.M. Character-Aware Neural Language Models. In Proceedings of the Thirtieth AAAI Conference (AAAI-16), Phoenix, AZ, USA, 12–17 February 2016.
47. 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: Berlin/Heidelberg, Germany, 2016; pp. 239–250.
48. Zhang, Y.; Yang, J. Chinese ner using lattice lstm. *arXiv* **2018**, arXiv:1805.02023.
49. Yang, J.; Zhang, Y.; Liang, S. Subword encoding in lattice lstm for chinese word segmentation. *arXiv* **2018**, arXiv:1810.12594.
50. Kuru, O.; Can, O.A.; Yuret, D. Charner: Character-level named entity recognition. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; pp. 911–921.
51. Limsopatham, N.; Collier, N.H. Bidirectional LSTM for named entity recognition in Twitter messages. In Proceedings of the 2nd Workshop on Noisy User-generated Text, Osaka, Japan, 11 December 2016.
52. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural architectures for named entity recognition. *arXiv* **2016**, arXiv:1603.01360.
53. Bhattu, S.N.; Krishna, N.S.; Somayajulu, D. idrbt-team-a@ IECSIL-FIRE-2018: Named Entity Recognition of Indian languages using Bi-LSTM. In Proceedings of Working Notes of FIRE 2018-Forum for Information Retrieval Evaluation, Gandhinagar, India, 6–9 December 2018.
54. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [[CrossRef](#)]
55. Barathi Ganesh, H.; Soman, K.; Reshma, U.; Mandar, K.; Prachi, M.; Gouri, K.; Anitha, K. Information Extraction for Conversational Systems in Indian Languages-Arnekt IECSIL; In Proceedings of the Forum for Information Retrieval Evaluation, Gandhinagar, India, 7–9 December 2018.
56. Forum for Information Retrieval Evaluation. Available online: <http://fire.irsi.res.in/fire/2019/home> (accessed on 2 February 2018).
57. Skymind. A Beginner’s Guide to Neural Networks and Deep Learning, 2017. Available online: <https://skymind.ai/wiki/neural-network> (accessed on 14 November 2018).

58. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the Twenty-eighth Conference on Neural Information Processing Systems, Montreal, Canada, 8–13 December 2014.
59. Na, S.H.; Kim, H.; Min, J.; Kim, K. Improving LSTM CRFs using character-based compositions for Korean named entity recognition. *Comput. Speech Lang.* **2019**, *54*, 106–121. [[CrossRef](#)]
60. Klein, D.; Smarr, J.; Nguyen, H.; Manning, C.D. Named entity recognition with character-level models. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4, Edmonton, Canada, 31 May–1 June 2003; pp. 180–183.
61. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning Word Vectors for 157 Languages. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018.
62. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
63. Yu, X.; Faleńska, A.; Vu, N.T. A general-purpose tagger with convolutional neural networks. *arXiv* **2017**, arXiv:1706.01723.
64. Ajees, A.; Idicula, S.M. CUSAT TEAM@ IECSIL-FIRE-2018: A Named Entity Recognition System for Indian Languages. In Proceedings of Working Notes of FIRE 2018 - Forum for Information Retrieval Evaluation, Gandhinagar, India, 6–9 December 2018.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).