HiLT@IECSIL-FIRE-2018: A Named Entity Recognition System for Indian Languages

Sagar, Srinivas P Y K L, Rusheel Koushik Gollakota, and Amitava Das

Indian Institute of Information Technology, Sri City, Andhra Pradesh 517646, India.
{sagar.r16,srinivas.p, rusheelkoushik.g16}@iiits.in
amitava.das@mechyd.ac.in

Abstract. In this paper, we describe our submission for FIRE 2018 Shared Task on Named Entity Recognition in Indian Languages[3]. Identification of Named Entities is important in several higher language technology systems such as Information Extraction systems, Machine Translation systems, and Cross-Lingual Information Access systems. The system makes use of the different contextual information of the words along with the Word-Level and Character-Level features that are helpful in predicting the various Named Entity Classes. This model is an end to end Language Independent Deep Learning Model for Named Entity Recognition to support all Indian Languages. Our model uses no domain-specific knowledge and no handcrafted rules of any sort to attain good results. The model solely relies on semantic information in form of word level embedding and character level embedding learned by unsupervised learning algorithms so that this model can be replicated across other languages.We have used 2CNN+LSTM model consisting of CNN as character-level encoder, CNN as word-level encoder and BiLSTM as the Tag Decoder.

Keywords: Named Entity Recognition · Deep Learning · Embeddings

1 Introduction

Named Entity Recognition (NER) is an important tool in almost all Natural Language Processing applications. Proper identification and classification of Named Entities are very crucial and pose a very big challenge to the NLP researchers. This work focuses on NLP approaches for identifying Named Entities such as Name, Number, Date, Event, Location, Things , Organization and Occupation. NER is an important step to various NLP Applications like Machine Translation, Question Answering , Topic Modelling , Information Extraction and many more. Further there is no concept of capitalization in Indian Languages like we have in English and thus this makes NER task more difficult and challenging as the rules prepared for English cannot be applied directly to Indian Languages , so we have come up with a Language Independent Deep Learning Model for Named Entity Recognition. In this paper, we use a simple 2CNN+LSTM model to predict the tag of a word in a sentence built using character embeddings and

word embeddings in order to extract various features at the Character-Level and Word-Level respectively.

The rest of the paper is organized as follows : In Section 2 we discuss about the dataset for our task. In Section 3 we discuss about our methodology and describe our model for Named Entity Recognition. In Section 4 we discuss about various Experiments and Results. In Section 5 we finally conclude our paper.

2 Related Work

NER task has been extensively studied in the literature of Natural Language Processing. Previous approaches in NER can be classified into Rule Based Approaches and Learning based approaches. Further Learning Based approaches can be divided into Machine Learning Based Approaches and Deep Learning Based Approaches. Another model was built for NER using large lists of names of people, location etc. in 1996[8]. Rule based system had a big disadvantage that a huge list was needed to be made for all the named entities before determining their Named Entity Class. They didn't had the capability of detecting a new Named Entity and determining their class if they are not already in the dictionary. Then comes the learning based approaches where people use feature learning based approach using Hand Crafted Rules like Capitalization ,etc. These features were given to machine learning based classifiers like SVM, Naive Bayes,etc. which were then used to classify into different classes[7]. But this is a problem with Indian language itself. Indian languages show frequent variation in nature, so very big dictionaries cannot be made. Further there is a problem in the language based approaches, since these language do not have capitalization and other things which can be used to easily classify. Further it was also treated as sequence labelling problem because the context is very important in determining the entities. HMM and CRF were used to solve such problems[4]. Further lately, deep active learning was used in order to gather more details, where character level and word level encoders were also used for named entity recognition in English Language. Deep learning approaches have also been applied using LSTM to predict the Named Entities and their class in Indian Languages[1].

3 Data

The labelled training for Hindi, Telugu, Tamil, Malayalam and Kannada was provided by FIRE 2018 where every word in the sentence was tagged with their corresponding NER Tag [2]. Further for pre-evaluation as well as final-evaluation the unlabelled data was also provided for all the five Indian Languages.

3.1 Data Format

Every line in the training file had one word and their corresponding tag separated by a tab space and a sentence ended with a "newline" and no tag corresponding

to it. The testing set data format was almost the same as training set, i.e. every line had a word but it didn't had a corresponding tag. And in the testing set, a sentence also ended with the same "newline".

3.2 Data Statistics

Table 1 describes the count of various tags in all five different languages. Here "other" is not a NER Tag , it is used to mark any word which is not a Named Entity. Since most of the words in a sentence, including "," and even "." , are not Named Entities so that's why other has the maximum count of tags on all the languages.

Table 2 describes the number of sentences and the corresponding count of Tagged Words in sentences for all the five languages.

Tag Distribution	Hindi	Telugu	Tamil	Kanadda	Malayalam
datenum	2672	2521	15482	1046	1609
event	2932	729	5112	523	837
location	166547	95756	134262	10473	29371
name	88887	60499	118021	15110	59422
number	37754	28618	77310	3948	30553
occupation	15732	8437	16507	3081	8037
organization	12254	2431	9999	746	4841
things	4048	1069	6183	242	1999
other	1141207	577625	1109354	262651	701664

Table 1: Tag Distribution in different Languages for the training data

Data-Set	Hindi	Telugu	Tamil	Kannada	Malayalam
Sentences	76537	63223	134030	20536	65188
Tagged Words	330826	200060	382876	35169	136669

Table 2: Sentences and Tagged Words in the training data

4 NER Model Description

We have built a 2CNN+LSTM model for this task of NER. For the task we have used word level and character level features of a word. We have used CNN to determine the Character Level Feature Vector and Word Level Feature Vector. For this task LSTM can also be used in place of CNN but since [6], LSTM are computationally much more expensive, so CNN was used. Fig. 1 describes the complete workflow through the model. Our model consists of 3 components:

4.1 Character Level encoder

Initially, Character embeddings[5] are created for all characters in the words in the training data. The length of the word varies in the training set so, each word

is made to a length of double the average length of the words in order to get uniform length of each word. Words smaller than the given length are padded and words longer are cut down . Further, Character tokenization is also done in order to rank characters according to their frequency in all words. Only top 75% of the characters are retained and all other characters are replaced by unknown <UNK_CHAR> tag.

The Character Level Encoder, extracts the character level features of a word. A sentence is taken as input and every word of the sentence is mapped to the corresponding characters in that word. Every character in a word is further mapped to corresponding character embedding . A word matrix is formed by combining all the character embedding to form a matrix for every word. Then CNN is applied over every word matrix formed by combining the character vectors of every word. Further a Max-Pooling layer is applied over every word matrix to get a Character Level Feature Vector.

4.2 Word Level Encoder

Word embeddings[5] are created for all the words using word2vec. We need to bring the sentences to uniform length, so sentences are made to a length double the average length of sentences in the training set. Sentences smaller than the given length are padded and sentences longer and cut down to this length. All the sentences are then tokenized and the words are ranked according to their frequency in the document . Further, in order to handle the case of unknown words the top 75 % of the word are retained and the remaining words are replaced by unknown <UNK_WORD> tag.

Word level encoder extracts the word level features from the surrounding words in a sentence. A sentence is taken as input and every word of the sentence is mapped to its word embeddings. Word embedding of a word is concatenated with the character level feature vector of that particular word. A matrix is formed by combining all the vectors of words to form a sentence-matrix. Then CNN is applied over the sentence-matrix. Max pooling layer is not applied because we need the same number of vectors as output as the length of sentence in order to determine the NER Tag of every word in the sentence.

4.3 Tag decoder

The Tag Decoder induces a probability distribution [1] over a sequence of tags in a sentence. A sentence is taken as input and every word of the sentence is mapped to its word embedding. Now the output which we got from the Word-Level CNN was concatenated with the corresponding word embeddings. Now this is given as the input to the Bidirectional LSTM. Further the Bi-LSTM determines the output tag of all the words in the sentence. Output of every word is passed through 3 projection layers and Softmax is applied on the last projection layer to determine the corresponding tag.

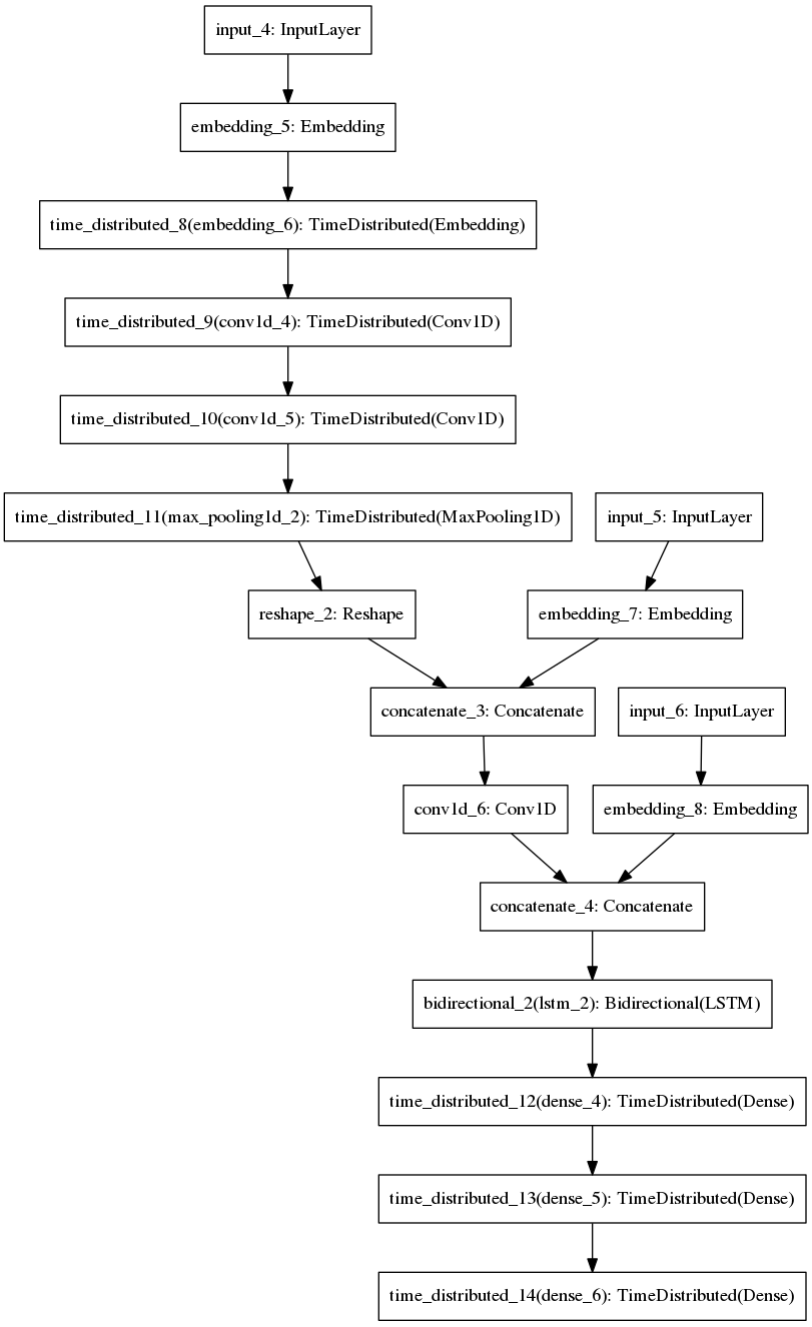

Fig. 1: Language Independent Named Entity Recognition Model

5 Experimental Results

Our model is a non-linear, and we need to tune a number of hyper parameters for an optimal performance. Hyper parameters are described as follows. The Character Level Embedding Size determines the length of the character embedding vector. We have used the embedding size to be 50 . The Word Embedding Size determines the length of Word Embedding Vector and we have set the embedding size to be 100 with window size of 5. Following this we selected top words and top characters after tokenization and marked all other words as UNK_WORD. We chose top 75% of the words for tokenization. We have used 2 CNN Layers with 20 and 11 filters respectively for extracting Character Level features in a word respectively. Then we have applied another CNN layer with 10 filters for extracting Word Level Features in a sentence. The number of layers of Bi-LSTM as a tag decoder and the number of hidden units in Bi-LSTM which are used to determine the tag of a sentence is also one of the important parameter. We have used just one layer of Bi-LSTM with hidden units equal to 300. Number of layers in the feed forward network and number of cells in each layer is also one of the hyper parameters. We have used 3 dense layers, the first and second dense layers have 100 cells and the third layer having the number of cells equal to number of classes we need to categorize.

5.1 Training

Adam optimizer is applied with "categorical_crosentropy" loss. We have splitted data into Training and Validation data validation split of 20% Now the model is trained for 25 epochs with an early stopping of patience 2 on validation loss. On this, most of the languages we got maximum accuracy in around 7 to 9 epochs and after that overfitting started.

5.2 Testing

For testing all the word not in the dictionary of the system are marked with unknown <UNK_WORD> and similarly for all the characters not in the list are marked as unknown <UNK_CHAR>. Further all the sentences are limited to double the average length of sentence which we had fixed earlier during training. Similarly all the words are limited to double the average length of words which we had fixed earlier during training. Results are predicted for the sentence for the length of double the average length of a sentence. For the remaining words the "other" tag is marked.

Table 3 describes our F1 Score which we achieved in all these five languages for both pre-evaluation and final-evaluation for the given five Indian Languages. Table 4 describes describes the F1 Score of all NER tags for all five Indian Languages.

F1-Scores	Hindi	Telugu	Tamil	Kannada	Malayalam
Pre-Evaluation	94.44	92.42	92.48	92.94	92.92
Final-Evaluation	94.35	92.47	91.79	93.17	92.1

Table 3: Our score for pre-evaluation and final-evaluation

F1 Scores	Hindi	Telugu	Tamil	Malayalam	Kannada
datenum	89	39	71	21	35
event	71	49	67	57	59
location	93	91	87	70	69
name	73	72	71	65	52
number	85	80	87	84	68
occupation	78	74	72	70	59
oragnisation	77	52	61	64	52
other	96	95	95	95	96
things	65	54	68	51	16

Table 4: Our Score for various tags of different languages

6 Conclusion

We have developed a named entity recognition system using deep learning for five different languages Hindi, Telugu, Tamil, Malayalam, Kannada. We have considered the Character Level and Word Level features. The model is a language independent model for Named Entity Recognition for Indian Languages which performs better than rule based approaches. This model can be extended further to many other Indian Languages as this approach does not require much of annotated data and also does not require an expert from that language to form the rules for Named Entity Recognition. Evaluation results show that the model performs slightly better for Hindi over other languages.

References

1. Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. Towards deep learning in hindi NER: an approach to tackle the labelled data sparsity. *CoRR*, abs/1610.09756, 2016.
2. H B Barathi Ganesh, K P Soman, U Reshma, Kale Mandar, Mankame Prachi, Kulkarni Gouri, Kale Anitha, and M Anand Kumar. Information extraction for conversational systems in indian languages - arnekt iecsil. In *Forum for Information Retrieval Evaluation*, 2018.
3. H B Barathi Ganesh, K P Soman, U Reshma, Kale Mandar, Mankame Prachi, Kulkarni Gouri, Kale Anitha, and M Anand Kumar. Overview of arnekt iecsil at fire-2018 track on information extraction for conversational systems in indian languages. In *FIRE (Working Notes)*, 2018.
4. Asif "Ekbal, Rejwanul Haque, Amitava Das, Venkateswarlu Poka, and Sivaji" Bandyopadhyay. "language independent named entity recognition in indian lan-

guages". In *"Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages"*, "2008".
5. Tomas Mikolov and Ilya Sutskever. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.
6. Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *CoRR*, abs/1707.05928, 2017.
7. Q Tri Tran, TX Thao Pham, Q Hung Ngo, Dien Dinh, and Nigel Collier. Named entity recognition in vietnamese documents. *Progress in Informatics Journal*, 5:14–17, 2007.
8. Takahiro Wakao, Robert Gaizauskas, and Yorick Wilks. Evaluation of an algorithm for the recognition and classification of proper names. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 418–423. Association for Computational Linguistics, 1996.