Named Entity Recognition in Tamil Language Using Recurrent Based Sequence Model

V. Hariharan, M. Anand Kumar and K. P. Soman

Abstract Information extraction is a key task in natural language processing which helps in knowledge discovery by extracting facts from the semi-structured text like natural language. Named entity recognition is one of the subtask under information extraction. In this work, we use recurrent based sequence models called Long Short-Time Memory (LSTM) for named entities recognition in Tamil language and word representation for words is done through a distributed representation of words. For this work, we have created a Tamil named entities recognition corpus by crawling Wikipedia and we have also used openly available FIRE-2018 Information Extractor for Conversational Systems in Indian Languages (IECSIL) shared task corpus.

Keywords Information extraction · Named entity recognition · Recurrent neural network · Long short time memory

1 Introduction

The rise of the internet has seen a huge growth in the amount of information available online in the form of text and as long as we continue to log our daily inferences and other works on the platform such as social media, blogs, email, etc., amount of information will be continuously increasing. Elective methods to process this information on a large scale without the need for human intervention will be an indispensable tool for the understanding this large amount information, this process is called as Information Extraction. One of the crucial subtasks of information extraction is Named

V. Hariharan (✉) · M. Anand Kumar · K. P. Soman
Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: hari03@live.in

M. Anand Kumar
e-mail: m_anandkumar@cb.amrita.edu

K. P. Soman
e-mail: kp_soman@amrita.edu

H. S. Saini et al. (eds.), *Innovations in Computer Science and Engineering*, Lecture Notes in Networks and Systems 74,
https://doi.org/10.1007/978-981-13-7082-3_12

Entity Recognition (NER). This task helps in the classifying the words in the text into predened categories like a person, location, and organization, etc. By knowing the named entities in the text helps us to categorize the text where these named entities occur. This NER subtask helps in creating a structured database from semi-structured text corpus [1]. In this work, supervised learning methods are used for solving the task of named entity recognition. A recurrent based sequence model such called LSTM is trained via supervised learning [2] approach to predict the named entities of the words in the corpus.

2 Literature Survey

Named entity recognition is a key task in ending the factual information from the semi-structured natural language. This has done through unsupervised methods by employing clustering and ranking techniques [3]. In semi-supervised (weakly supervised) methods, techniques like bootstrapping [4] are very popular approach, i.e., having a small sample of good annotation for training. In earlier supervised methods, support vector machines and conditional random fields are used for training [5]. All these earlier methods used features such as co-occurrences, word substrings, punctuation, and handcrafted linguistic patterns. These features are crafted specially according to the domain and the language, so these features cannot easily be scaled. The recent development in the distributed representation proposed by [6] produces as dense vector representation for the word based on the entire corpus this forms a good representation for the word. In this work, we employ a class of neural network architecture called recurrent based sequence models for the learning of the named entities in the text.

2.1 Distributed Representation

Word representation is one of the crucial components in any natural language processing system. The distributed representation framework like word2vec, Glove and fastest gives a dense vector representation for each word in the corpus. The dense vector representation for the word is formed by taking softmax for the word over the context window through an iterative process and for updating the information of the word not in the context window, negative sampling is done, i.e. (randomly selecting words that are not in the context window instead of iterating through the all the words in the corpus) [7]. This iteration over the whole corpus is continued until a good analogy between the word vectors is obtained like in the Eq. 1.

$$\text{'king'} - \text{'man'} + \text{'woman'} = \text{'queen'} \tag{1}$$

or a good performance of the system where these word vectors are employed [8]. This dense vector representation formed by the distributed representation framework aims to capture the syntactic and semantics of the word with respect to all words in the corpus. One of the major drawbacks of the word2vec and Glove model is it does not consider the internal structure of the words so it fails the capture the morphology present in the language, fastText overcomes this drawback by considering the different character n-grams of the word along with the word itself to jointly learn the vector representation for word. By considering the character n-grams of words, the morphology of the language is learned to some extent.

2.2 Sequence Based Recurrent Models

All natural language can be best inferred when it is processed as a sequence or with respect to some context. So in order for the neural network to understand the sequential information of the language [9], a recurrent based neural network called recurrent neural network (RNN) which processes each word in the sentence at different time steps, so that the information of the previous timestep is passed on the next timestep. This is done by connecting the hidden state of the previous time step with current timestep. In RNN, the number of time steps depends on the upon the number of words in the sentence given in as input. So RNN can naturally handle variable length sequence.

An RNN unrolled across its time steps is shown in Fig. 1. At each time step $x_{0:t}$, the RNN takes one input in Fig. 1, x_t corresponds to the input at various time steps, h_t corresponds to the output at each time step and A corresponds to the hidden layer.

$$a_t = g(V[x_t : a_{t-1}] + c) \tag{2}$$

$$h_t = Wa_t + b \tag{3}$$

In Eq. 2, the hidden state of each time step a_t is calculated and g is the nonlinear activation function which is applied over the linear transformation between the weight matrix V and the input of the current time step x_t concatenated with a previous hidden

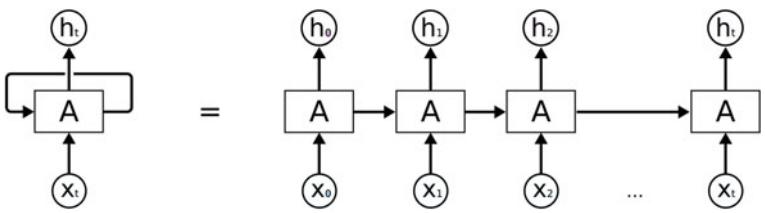

Fig. 1 A single RNN cell versus single RNN cell unrolled across its time steps

state a_{t-1}. In Eq. 3, linear transformations are done over the hidden state a_t and added with bias term b, to get the output of current time step.

2.3 Long Short Time Memory

Long Short Time Memory (LSTM) is slight modification over the vanilla RNN to overcome exploding and vanishing gradient problem [10]. In LSTM, there is an additional state called cell state, this state is often referred as the memory element of the LSTM, because in the cell state, addition operation is performed with the previous cell state, weighted by input and forget gate. Instead of multiplication with the previous hidden state in the vanilla RNN, this causes the gradient to ow across many time steps with decay. The mathematical equation of LSTM is given below

$$c_n = f_n c_{n-1} + i_n \tanh(V[x_n; h_{n-1}] + b_c) \tag{4}$$

$$h_n = o_n \tanh(W_h c_n + b_h) \tag{5}$$

The cell state c_n is calculated in the Eq. 4 and hidden state h_n is calculated in the Eq. 5. Where i_n, f_n and o_n is the input gate, forgot gate, and output gate, respectively. A single time step of LSTM cell is shown in Fig. 2.

The equation of the input gate i_n, forget gate f_n and output gate o_n is given in Eqs. 6, 7, and 8.

$$i_n = \sigma\,(W_i[x_n; h_{n-1}] + b_i) \tag{6}$$

$$f_n = \sigma\,(W_f[x_n; h_{n-1}] + b_f) \tag{7}$$

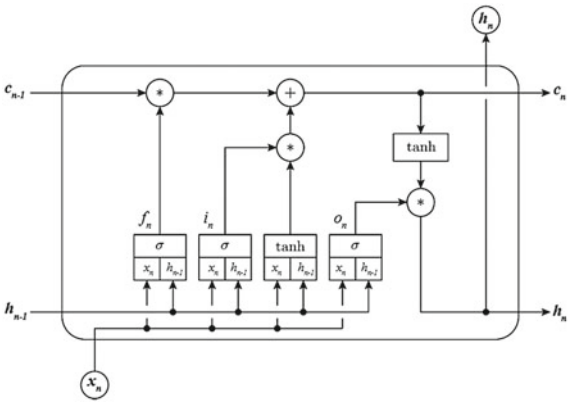

Fig. 2 A single time step of LSTM cell

$$o_n = \sigma\left(W_o[x_n; h_{n-1}] + b_o\right) \tag{8}$$

The output of these gates is a vector and value of each element in the vector is bounded between 0 and 1 which is due to the sigmoid nonlinearity. The input gate i_n and forget gate f_n grant the LSTM network the ability like how much the network can remember from past timesteps and how much to copy from current timestep. The input gate set to zero is like ignoring the current timestep and forget gate set to zero is like forgetting everything in the cell state (memory) therefore starting fresh. The output gate o_n gives a vector which controls how much amount of information is exposed by the current cell state to the network which is reading from it. If some elements in the output gate vector are zero, then it cancels out those corresponding dimension in the cell state, so it is not visible to the network which is reading from it.

3 Methodology

In this work, supervised learning methods are used for solving the named entity recognition problem. In the approach, a recurrent based neural network called LSTM is used to learn the named entities in the text sequence. LSTM is preferred over the recurrent based neural network since it captures the long-term dependencies in the text sequence without suffering from vanishing and exploding gradient problem. The architecture of the NER model is shown in Fig. 3.

For this work, we have used two corpora for named entity recognition in the Tamil language. The rst corpus is taken from the openly available FIRE-2018 Information Extractor for Conversational Systems in Indian Languages (IEC-SIL) shared task [11], in this corpus, there are about 1,34,030 sentences with 8 entity tag and an other tag. The second corpus is made by crawling Tamil Wikipedia and the annotation for the named entity tag is made by using property tag present in the Wikidata website. In this corpus, there are about 2,12,873 sentences with 20 entity tags and another

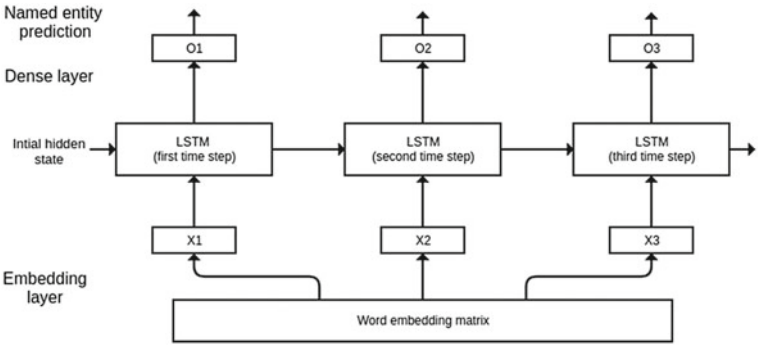

Fig. 3 Named entity recognition model

tag. The corpus statistics are shown in Tables 1 and 2. The word representations are done through distributed representation framework like Glove [12] and fastText [13]. Through distributed representation, each word in the corpus is given a dense vector of certain dimension. These word vectors are fed into the recurrent based neural network model called Long Short Time Memory (LSTM). The recurrent networks can naturally handle variable length sequences. So in this case, the number of words in the sentence determines the time step. To predict the named entity tag for each word in the sequence, dense layer is applied for each time step. The output of the dense layer corresponds to the number of classes, i.e. (number of named entity tags). In addition to initializing of word vectors through distributed word representation framework, we have also used randomly initialized the word embedding and feed into the LSTM network.

Table 1 IECSIL corpus statistics

Sl. no	Entity tag	Freq (%)
1	Datenum	1.03
2	Event	0.34
3	Location	8.99
4	Name	7.90
5	Number	5.18
6	Things	0.41
7	Occupation	1.10
8	Organization	0.67
9	Other	74.34

Table 2 Tamil Wiki corpus statistics

Sl. no	Entity tag	Freq	Sl. no	Entity tag	Freq
1	Date	2.88	12	Geometry	0.05
2	Work of art	6.33	13	Law	0.22
3	Property	0.07	14	Norp	5.95
4	Concept	0.17	15	Money	0.05
5	Time	5.11	16	Event	0.63
6	Org	1.8	17	Quantity	0.97
7	Gpe	10.7	18	Entity	5.9
8	Abstract	0.14	19	Language	2.57
9	Person	6.4	20	Facility	0.17
10	Location	3.38	21	Others	46.01
11	Product	0.38			

4 Result and Discussion

In this work, the task of named entity recognition is solved via supervised learning approach using recurrent based sequence network called Long Short-Time Memory (LSTM). This named entity recognition model is applied on FIRE-2018 IECSIL shared task corpus and Tamil Wiki corpus. On these two corpora, we have trained a distributed word representation network called fastText and Glove to give dense word vectors of different dimensions and word vectors which are randomly initialized is also used for evaluation. These word vectors are fed into the NER model. From the results of the two corpora, it is seen that word embedding obtained by the distributed word representation framework like Glove and fastText performs better than the randomly initialized word vectors. It is also seen that the fastText based distributed representation performs better than the Glove based distributed representation framework this is because in fastText based distributed representation, the word representations are jointly learned along with the different character n-grams of the word and it also forms a better word representation for the unknown words. Since Tamil is a morphologically rich language fastText based word embedding is seen to perform better than other distributed representation methods (Tables 3 and 4).

Table 3 Results for IECSIL@FIRE-2018 corpus

Sl. no	Dimension	Rand emb + LSTM (F1 macro)	Glove + LSTM (F1 macro)	fastText + LSTM (F1 macro)
1	50	85.44	90.26	93.1
2	100	86.03	90.62	93.55
3	150	87.11	92.3	93.73
4	200	87.43	93.03	94.32
5	250	87.01	93.01	94.3
6	300	87.23	93.06	94.54

Table 4 Results for Tamil Wikipedia corpus

Sl. no	Dimension	Rand emb + LSTM (F1 macro)	Glove + LSTM (F1 macro)	fastText + LSTM (F1 macro)
1	50	83.89	86.23	89.1
2	100	83.83	88.82	90.55
3	150	84.11	88.79	90.73
4	200	84.43	89.23	90.32
5	250	84.33	90.31	91.3
6	300	84.03	90.06	91.29

5 Conclusion and Future Work

The named entity recognition is a critical task in information extraction as it helps in identifying the entities like a person, location, etc. This helps in organizing the unstructured text in a structured format. In this paper, for identifying the named entities in the text, a recurrent based neural network called LSTM is applied on IECSIL@FIRE-2018 shared task corpus and Tamil Wiki corpus. The word representation is done by using a distributed word representation framework called fastText and Glove. In this work, word vectors of different dimension are used and it is seen that words vectors obtained through the fastText distributed representation perform better when compared to other embeddings. This is because fastText enriches the word vector with character n-gram information and Tamil is a morphologically rich language, fastText embeddings perform better for the named entity recognition task. In our next work, we are planning to make an end to end neural network for information extraction. Which does named entity recognition and relation extraction in a single stretch.

References

1. Remmiya Devi G, Veena PV, Anand Kumar M, Soman KP (2018) Entity extraction of Hindi–English and Tamil–English code-mixed social media text. In: Forum for information retrieval evaluation. Springer, pp 206–218
2. Lample G, Ballesteros G, Subramanian S, Kawakami K, Dyer C (2016) Neural architectures for named entity recognition. arXiv:1603.01360
3. Remmiya Devi G, Veena PV, Anand Kumar M, Soman KP (2016) Entity extraction for malayalam social media text using structured skip-gram based embedding features from unlabeled data. Proced Comput Sci 93:547–553
4. Mintz M, Bills S, Snow R, Jurafsky D (2009) Distant supervision for relation extraction without labeled data. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP, vol 2. Association for Computational Linguistics, pp 1003–1011
5. Abinaya N, Anand Kumar M, Soman KP (2015) Randomized kernel approach for named entity recognition in Tamil. Indian J Sci Technol 8(24)
6. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
7. Barathi Ganesh HB, Anand Kumar M, Soman KP (2018) From vector space models to vector space models of semantics. In: Forum for information retrieval evaluation. Springer, pp 50–60
8. Anand Kumar M, Soman KP, Barathi Ganesh HB (2016) Distributional semantic representation for text classification and information retrieval. In: CEUR workshop proceedings, vol 1737, pp 126–130
9. Sundermeyer M, Schlüter R, Ney H (2012) LSTM neural networks for language modeling. In: Thirteenth annual conference of the international speech communication association
10. Hochreiter S, Schmidhuber S (1997) Long short-term memory. Neural Comput 9(8):1735–1780

11. Barathi Ganesh HB (2018) Information extractor for conversational systems in Indian languages @ forum for information retrieval evaluation. http://iecsil.arnekt.com/!/home
12. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1532–1543
13. Bojanowski P, Grave E, Joulin A, Mikolov T (2016) Enriching word vectors with subword information. arXiv:1607.04606