BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 69(6), 2021, Article number: e138820
DOI: 10.24425/bpasts.2021.138820



# Deep Learning based Tamil Parts of Speech (POS) Tagger

**S. ANBUKKARASI[1]* and S. VARADHAGANAPATHY[2]**

[1] Department of Computer Science and Engineering, Kongu Engineering College, India
[2] Department of Information Technology, Kongu Engineering College, India

**Abstract.** This paper addresses the problem of part of speech (POS) tagging for the Tamil language, which is low resourced and agglutinative. POS tagging is the process of assigning syntactic categories for the words in a sentence. This is the preliminary step for many of the Natural Language Processing (NLP) tasks. For this work, various sequential deep learning models such as recurrent neural network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Bi-directional Long Short-Term Memory (Bi-LSTM) were used at the word level. For evaluating the model, the performance metrics such as precision, recall, F1-score and accuracy were used. Further, a tag set of 32 tags and 225 000 tagged Tamil words was utilized for training. To find the appropriate hidden state, the hidden states were varied as 4, 16, 32 and 64, and the models were trained. The experiments indicated that the increase in hidden state improves the performance of the model. Among all the combinations, Bi-LSTM with 64 hidden states displayed the best accuracy (94%). For Tamil POS tagging, this is the initial attempt to be carried out using a deep learning model.

**Key words:** POS tagging; deep learning model; natural language processing; Bi-LSTM.

## 1. INTRODUCTION

Part of speech (POS) tagging is the preliminary task for many natural language processing (NLP) applications. It is the process of recognizing the words in a text as a noun, verb, adverb, adjective, etc. Although the research work on POS tagging emerged a few decades ago, it is still considered a thriving research area in the field of NLP. Results of taggers can be used in applications such as named entity recognition, machine translation and grammar checking [1–4]. For example, from tagged sentences, an entity can be specified as name, place or thing. Many works have been carried out for automating POS tagging tasks in various languages. POS tagging techniques are classified into rule-based, statistical-based and hybrid approaches. Of late, neural network-based models have given the best results for most NLP-related problems, which can also be applied to POS tasks. Earlier works used rule-based techniques, where the scholars of a particular language would derive the rules for tagging. Rules can even be learned from the input, and rulebased taggers can be enriched with new rules [5]. The primary difficulty in rule-based technique is that it requires deep linguistic knowledge; moreover, enormous hand-crafted rules have to be generated to align the language's grammar, which requires language proficiency.

Statistical or stochastic taggers work based on probability, statistics or frequency. N-gram, hidden Markov model (HMM) and maximum likelihood estimation (MSE) are statistical approaches used in POS taggers. These techniques require only partial linguistic knowledge, unlike rule-based techniques, which necessitate thorough knowledge of grammar. However, the drawback of the system is that, at times, the grammar gets violated, and the text is tagged based on the probability it calculates from the previous and following words [6, 7]. The hybrid approach combines the rule-based as well as the stochasticbased techniques to derive the POS tagger. For agglutinative languages, morphologically rich, the hybrid technique can be used, as these kinds of languages require an extensive set of tags [8].

While many European languages have POS taggers that are implemented with a variety of techniques, Indian languages are still striving for good POS taggers. India is a linguistically diversified country where people from different states speak different languages. Many NLP tasks such as machine translation, question answering and grammar checking are carried out in various Indian languages [9, 10]. POS tagging is one such task as it is the preliminary technique for most of the aforementioned NLP tasks. This paper presents a deep learning-based POS tagger for one such Indian Language – Tamil.

The paper is organized as follows: Section 2 presents a literature survey of Indian as well as other languages; Section 3 presents details about Tamil language; Section 4 deals with various approaches to POS tagging; Section 5 deals with materials and methods used; Section 6 presents experimental setup; Section 7 discusses results and evaluation; and Section 8 concludes the paper.

*e-mail: anbu.1318@gmail.com

Manuscript submitted 2021-04-15, revised 2021-08-03, initially accepted for publication 2021-08-31, published in December 2021

## 2. RELATED WORK

Several POS tagging models such as the Markov method, rule-based methods, Kernel method and hybrid method have been

developed for various natural languages [11–13]. In [14], the anchor hidden Markov model (AHMM), a restricted HMM class, was developed to create unsupervised POS tagging. The research introduced AHMMs that learn from unlabelled data by creating an estimation method. Using anchor tags along with features, an accuracy of 71% was derived. In [15], tri-gram-based HMMs were introduced for Indian languages, namely Bengali, Telugu, Marathi and Hindi, using second-order HMM. In the research, unknown words were handled using the prefix analysis method and word-type analysis method. In [16], based on the context of both sides of the given word, HMM tagger was developed. The research concluded that sequential unsupervised training of tag sequence and lexical probabilities in an HMM improves accuracy over simultaneous training for certain models.

In [17], a combination of bi-directional long short-term memory (Bi-LSTM) and conditional random field (CRF) model was used for sequence tagging; past and future features of input of Bi-LSTM and sentence-level tagging information of CRF for tagging purpose were used. The authors claim that without word embedding, their tagger achieved good accuracy. In [18], a generative graphic model called deep neural network (DNN) was utilized for training the model. The authors employed the tag set developed by Microsoft Research India (IL-POST). They used various features such as length of a word, suffix and prefix of a word, POS information of the previous words and a predefined dictionary for tag classification. In [19], an auxiliary-loss-based novel multi-task Bi-LSTM was proposed. This auxiliary loss helps improve the accuracy of the rare words to be tagged. The research presents neural network-based POS tagging for 22 languages, and character embedding is combined with word embedding in a hierarchical network to represent the words better.

In [20], the naive Bayes algorithm, one of the supervised learning algorithms, was used to classify the text documents related to women health issues in the Tamil language. The research applied 6549 words for training the model and obtained an F1-score of 80%. In [21], supervised and unsupervised methods were used with multilingual parallel corpora for Tamil POS tagging. In the research, various techniques such as HMM, support vector machine (SVM) and CRF were implemented, and the SVM model was found to give the maximum accuracy (61.29%). In [22], an SVM-based POS tagger was developed for the Tamil language with a tag set, with the corpus containing 225 000 words collected from newspapers, online articles and short stories. The tagged corpus has been used for developing chunking corpus.

Akhil *et al*. [23] used a deep learning-based approach for POS tagging in the Malayalam language. They made use of the publicly available Malayalam dataset with a total of 287 588 tagged words and the tag set of 36 tags from the Bureau of Indian Standards (BIS). They experimented with different deep learning models such as long short-term memory (LSTM), gated recurrent unit (GRU) and Bi-LSTM. The experiments were conducted with 4, 16, 32 and 64 hidden layers. It was claimed that the Bi-LSTM model with 64 hidden layers achieved an f-measure of 98%.

From the literature survey, it is clear that very few research studies have carried out POS tagging for the Tamil language; moreover, these studies have only utilized traditional methods. Hence in this paper, a deep learning-based POS tagger for the Tamil language is proposed. Further, various neural network models are created, and the comparison is presented.

## 3. TAMIL LANGUAGE

Tamil is a Dravidian language spoken in the regions such as Tamil Nadu (India), Sri Lanka and Malaysia. Tamil has an extremely rich morphology and agglutinative grammar. It uses suffixes to mark the case, number and class of a noun. For a Tamil word, a lexicon root can be attached with one or more affixes. An affix can be considered as a suffix that can be either derivational or inflectional. The Tamil language has 12 vowels, 18 consonants, 216 compound characters and one unique character (*Ayudha Ezhuthu*). Thus, in total, there are 247 letters in the Tamil language.

Further, Tamil vowels are divided into five short vowels, five long vowels and two diphthongs. Consonants are classified into three categories with six consonants each: *vallinam* (hard), *mellinam* (soft) and *idaiyinam* (medium). Tamil is a head-final language, i.e., the verb comes at the end of the clause, which follows subject–object–verb (SVO) order. Table 1 depicts the sample tagged Tamil sentence.

**Table 1**
Example Tagged Tamil Sentence

| Sentence | Tag | Translation |
|---|---|---|
| எங்கள் | <PRP> | We forgot to send our son's marriage invitation to him; We don't know the address too. |
| மகனின் | <NN> | |
| திருமணப் | <NN> | |
| பத்திரிகையை | <NN> | |
| அவருக்கு | <VINT> | |
| அனுப்ப | <PRP> | |
| மறந்து | <VNAV> | |
| போனோம் | <VF> | |
| முகவரியும் | <NN> | |
| தெரியாது | <VAX> | |

## 4. POS FOR TAMIL LANGUAGE USING DEEP LEARNING APPROACHES

Lately, deep learning models have become the predominant technology in machine learning. This data-starving methodology uses data-derived features rather than the manually designed features used in earlier methods (KNN, naive Bayes, etc.). For solving image analysis problems, convolutional neural networks (CNN) work better [24]. As far as text data is concerned, it is

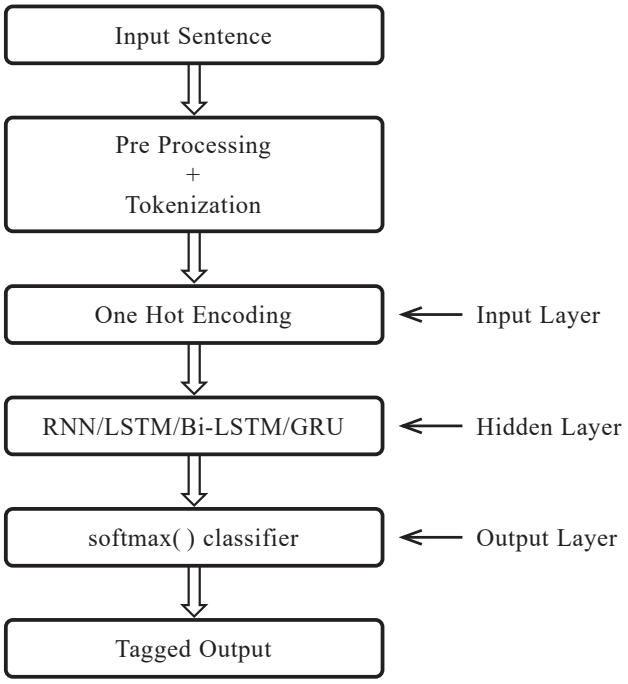


**Fig. 1.** POS Tagging with Deep Sequential Model

better to have deep learning-based sequential models than feed-forward models since they struggle with fixed context length [25, 26]. This paper presents various deep learning sequential models, such as recurrent neural network (RNN), LSTM, Bi-LSTM and GRU, for evaluating tagged Tamil text. Figure 1 presents the basic architecture of the proposed sequential deep learning-based approach for POS tagging of Tamil text.

At first, the text is passed to pre-processing and tokenization steps. Next, all the punctuation marks except the dot are removed from the text. Then, sentences are tokenized by considering the left and right context of the token for determining its POS tag. All these tokens are vectorized using the one-hot encoding technique. After tokenization, all the texts are converted into sequences, and these sequences are padded. In this way, data pre-processing is implemented, and resulting data are further fed to the training stage with RNN, LSTM, Bi-LSTM and GRU. Softmax is chosen as a classifier since the proposed work is considered a multi-class problem [27]. The network architecture implementation is done with Python's Keras package. As far as the Tamil language is considered, till date, very few NLP works have been carried out since it is a low-resource and agglutinative language. Further, most of the existing works in NLP for Tamil employ only traditional approaches. This work is one of the first of its kind to create POS tagging for the Tamil language using deep learning approaches.

## 5. MATERIALS AND METHODS

### 5.1. RNN

RNN is the extended feedforward network (FFN), which can handle variable-length sequences. It handles the data in a sequential form $(a_1, a_2 \ldots a_n)$. In sequential data, the length of the sequence or the number of elements is not always the same. It gets the output from the previous step and passes this as input to the current step, which forms a looping structure. RNN contains hidden states that accumulate some additional information about the sequence. The current state is calculated using,

$$h_t = f(a_t h_{t-1}), \tag{1}$$

$h_t$ $\rightarrow$ current State
$h_{t-1}$ $\rightarrow$ previous State
$a_t$ $\rightarrow$ input state.

The function $f$ is known as a non-linear function or activation function such as tanh, ReLU etc. The output is calculated using the given formula: $y_t$ output and $W_{hy}$ weight is at a particular layer.

$$y_t = W_{hy} h_t \,. \tag{2}$$

When the sequence length is long, it leads to vanishing gradient issues. To avoid this problem, gated RNNs can be used, where various gates are used to control the flow of the sequence.

### 5.2. LSTM

LSTM is the variant of RNN that uses a gating mechanism for capturing long dependencies. It consists of a memory cell, which acts as long short-term memory with three gates, which remember information for a longer time. LSTM performs three tasks: 1) decides how much past data it must remember; 2) decides how many units from the current state must be added, and 3) decides which part of the current state will make the output. Three gates of LSTM can be given as,

$$i_t = \sigma\left(w_i[h_{t-1}x_t] + b_i\right), \tag{3}$$

$$f_t = \sigma\left(w_f[h_{t-1}x_t] + b_f\right), \tag{4}$$

$$o_t = \sigma\left(w_o[h_{t-1}x_t] + b_o\right), \tag{5}$$

$i_t$ – input gate
$f_t$ – forget gate
$o_t$ – output gate
$i_t$ – input at current timestamp
$b_x$ – bias for the respective gate
$w_x$ – weight for the respective gate neurons.

### 5.3. Bi LSTM

In LSTM, information is passed in one direction and forms a loop. Conversely, in Bi-LSTM, the data sequence is passed in both forward and backward directions; hence, the model remembers the information for a longer time. It suits many NLP applications, such as machine translation and question answering, as it processes the data sequence in both directions. Experiments are performed with two LSTMS to process the data in two directions, and finally, results are concatenated before applying classifiers.

### 5.4. GRU

GRU is the improved version of RNN. It has only two gates, namely the update gate and reset gate, to solve the vanishing gradient problem. These are two vectors that decide what infor-

mation should be passed to the output unit. It can remember the information from the long past. The following equation gives the update gate:

$$(6)U_t = \sigma\left(W(u)x_t + Z(u)h_{t-1}\right). \tag{6}$$

The reset gate is given by,

$$R_t = \sigma\left(W(r)x_t + Z(r)h_{t-1}\right). \tag{7}$$

Current memory content is given by,

$$h'_t = \tanh\left(Wx_t + rt \odot Zh_{t-1}\right). \tag{8}$$

### 5.5. Training and classifier

Each architecture described above must learn the parameters called weight matrices. Cost function determines the values of the parameters. The error of the cost function is reduced through the gradient-descent method. Based on the error's gradient of cost function reduction $\left(-\frac{\partial E}{\partial W}\right)$, the weights of the parameters are updated. The negative sign in the error function denotes the direction of the error detection. The amount of error is affected based on the weight parameter. The loss function is the estimation of error between the predicted label and the true label. The weights continue to get adjusted until the error value becomes low. The output layer in a neural network is placed with the softmax function. It assigns a probability for the tags of the given words. The tag estimation for the given word $w_i$ is denoted as

$$\hat{y}_i = \underset{k\in 1,2,\ldots n}{\arg\max}\, P_i\left(k\middle|w_1, w_2, w_3, \ldots, w_s\right). \tag{9}$$

## 6. EXPERIMENTAL SETUP

This section specifies the experimental setup used in the proposed deep learning based-POS tagger for the Tamil language. The experiments were conducted on a machine with a configuration of i3-5005U @ 2 GHZ with 4 GB RAM. The experiment was carried out using Keras, which a deep learning application programming interface (API) is written in Python language [28]. This package ran on top of the machine learning platform TensorFlow. The dataset used for our work contained 225 000 Tamil words, which were tagged in [22]. The data were collected from various sources such as newspapers, online articles and stories. This is one of the very few available datasets for POS tagging of the Tamil language. Although it is not publicly available, it could be obtained on a request basis. A snapshot of the dataset is given in Fig. 2. The current work is the first of its kind to implement POS tagging for the Tamil language with deep learning models. For experimentation purposes, all the models were trained with 4, 16, 32 and 64 hidden states, and the number of epochs taken was 10. The model showed overfitting, and the validation loss started increasing after the tenth epoch. Therefore, an epoch is kept as 10. The learning rate was fixed as 0.01. The loss function used was cross-entropy, the optimizer was Adam, the activation function was softmax and batch size was chosen as 128. Precision, recall, accuracy and F1-score were used as evaluation metrics for the experiment. Out of 17 389 sentences, 1253 were used for training purposes and the rest for testing and validation. The highest metric for the models was obtained when the hidden state was set as 64. Section 7 presents the results of this work in detail.

| Word | POS Tag |
|---|---|
| உங்கள் | <PRP> |
| ஆசிரியர் | <NN> |
| ஒன்றாக | <ADV> |
| செயலிழக்கச் | <VINT> |
| செய்து | <VNAV> |
| பேசினார் | <VF> |
| கசப்பான | <ADJ> |
| எனவே | <CNJ> |
| மன்னிப்பது | <VBG> |
| கேட்கும் | <VNAJ> |

**Fig. 2.** A snapshot of the training dataset

## 7. RESULTS AND EVALUATION

This section discusses in detail the results of the experiment as well as their analysis.

In this study, evaluation was performed at the word level. Each word and the immediate words to the left and right of the words were considered as context to create the vector. In other words, a sequence was considered with three tokens. The vocabulary size determined the dimension of the vector for the word $w_i$. This was then trained and evaluated using various deep learning approaches such as RNN, LSTM, Bi-LSTM and GRU. For a multi-class classification problem, the cross-entropy loss function performed well.

Although all the models showed good metrics, the highest metric was obtained for Bi-LSTM with 64 hidden units. Evaluation measures for various states of the Bi-LSTM model are depicted in Table 2. Since the dataset consisted of unequal classes, it would be better to consider the F1-score than the accuracy metric. RNN, LSTM and GRU models were compared with the Bi-LSTM model, and the result is depicted in Table 3.

The second highest score was obtained for the GRU model. The metric loss specified how well the model was learning. It gradually decreased as the model improved the accuracy with fixed parameters. Figures 3 and 4 show the accuracy vs epoch and loss vs epoch, respectively, for all the models at the word level, when the models obtained high accuracy with the related hidden state value.

From the figures, it is evident that the Bi-LSTM model with hidden state parameter 64 works better than other models for the proposed work. It could be improved further by fine-tuning parameters and using an improved corpus.

**Table 2**
Evaluation measures of various hidden states for Bi-LSTM at the word level

| Methods | Precision | Recall | Accuracy | F1-Score |
|---|---|---|---|---|
| BiLSTM/4 | 0.9989 | 0.8695 | 0.9003 | 0.9297 |
| BiLSTM/16 | 0.9979 | 0.8795 | 0.9129 | 0.9349 |
| BiLSTM/32 | 0.9969 | 0.8866 | 0.9193 | 0.9385 |
| BiLSTM/64 | 0.9909 | 0.9132 | 0.9443 | 0.9504 |

**Table 3**
Comparison of various models

| Methods | Loss | Accuracy | F1-Score |
|---|---|---|---|
| RNN | 0.3026 | 0.9004 | 0.9001 |
| LSTM | 0.3108 | 0.9123 | 0.9102 |
| GRU | 0.2953 | 0.9140 | 0.9121 |
| Bi-LSTM | 0.9909 | 0.9443 | 0.9504 |

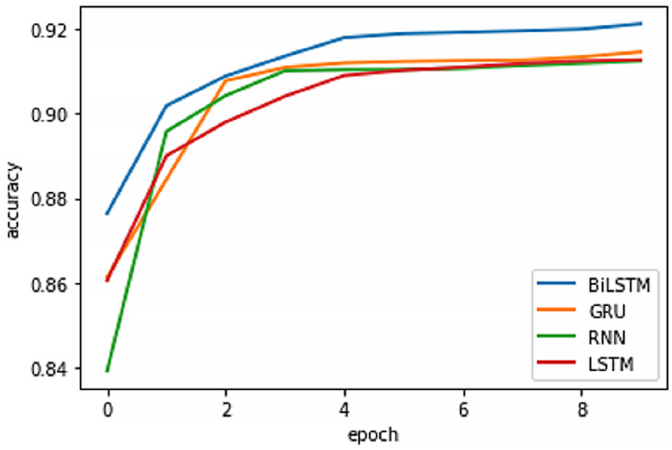


**Fig. 3.** Highest accuracy for all the models

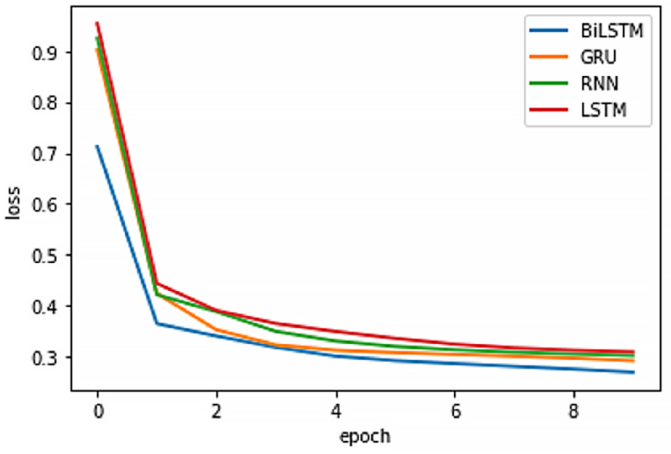


**Fig. 4.** Loss vs Epoch for the models

## 8. CONCLUSION

This study focused on POS tagging for the Tamil language using deep learning models. Although a few works have been carried out with other stochastic techniques, this work is the first of its kind as it uses deep learning models for Tamil POS tagging. For the experimentation purpose, various RNN models such as LSTM, GRU and Bi-LSTM were considered. Bi-LSTM with 64 hidden states yielded the best accuracy and F1-Score at the word level out of all the models. In the future, the corpus size could be increased, and tags can be further morphologically analyzed to improve accuracy based on the context. Moreover, attention-based transformer architecture could be implemented in future as it gives promising results for NLP-oriented tasks since it does not rely on any recurrent networks.

## REFERENCES

[1] R. Rajimol and V.S. Anoop, "A framework for named entity recognition for Malayalam – A Comparison of different deep learning architectures," *Nat. Lang. Process. Res.*, vol. 1, pp. 14–22, 2020.

[2] Y. Liu *et al.*, "Multilingual denoising pre-training for neural machine translation," *Trans. Assoc. Comput. Ling.*, vol. 8, pp. 726–742, 2020.

[3] K.S. Kalaivani and S. Kuppuswami, "Exploring the use of syntactic dependency features for document-level sentiment classification," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 67, pp. 339–347, 2019, doi: 10.24425/bpas.2019.128608.

[4] S. Anbukkarasi and S. Varadhaganapathy, "Machine Translation (MT) techniques for Indian Languages," *Int. J. Recent Technol. Eng.*, vol. 8, 86–90, 2019, doi: 10.35940/ijrte.B1015.0782S419.

[5] E. Brill, "A simple rule-based part of speech tagger," in *Proc. 3rd Conference on Applied Natural Language Processing*, *Association for Computational Linguistics*, 1992, pp. 152–155, doi: 10.3115/974499.974526.

[6] T. Berg-Kirkpatrick, A. Bouchard-Côté, J. DeNero, and D. Klein, "Painless unsupervised learning with features," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 582–590.

[7] N. Bölücü and B. Can, "Joint PoS tagging and stemming for agglutinative languages," in *Proc. of the International Conference on Computational Linguistics and Intelligent Text Processing*, 2017, pp. 110–122.

[8] P. Arulmozhi, T. Pattabhi R.K. Rao and L. Sobha, "A Hybrid POS Tagger for a Relatively Free Word Order Language," [Online]. Available https://www.academia.edu/23833233/A_Hybrid_POS_Tagger_for_a_Relatively_Free_Word_Order_Language (Accessed: Jan, 10, 2021)

[9] J. Singh, N. Joshi, and I. Mathur, "Development of Marathi part of speech tagger using statistical approach," in *Proc. of International Conference on Advances in Computing, Communications and Informatics*, 2013, pp. 1554–1559.

[10] M. Ramanathan, V. Chidambaram, and A. Patro, "An Attempt at Multilingual POS Tagging for Tamil," [Online]. Available http://pages.cs.wisc.edu/~madhurm/CS769_final_report.pdf (Accessed: Jan. 10. 2021).

[11] N. Bölücü, B. Can, "A Cascaded Unsupervised Model for PoS Tagging," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 20, pp. 1–23, Mar. 2021, doi: 10.1145/3447759.

[12] S. Adinarayanan and N.S. Ranjaniee, "Part-of speech tagger for sanskrit. A state of art survey," *Int. J. Appl. Eng. Res*., vol. 10, pp. 24173–24178, 2015. doi: 10.37200/IJPR/V23I1/PR190243.

[13] H. Ali, *Unsupervised Parts-of-Speech Tagger for the Bangla language,* Department of Computer Science. University of British Columbia, 2010. [Online]. Available: https://www.cs.ubc.ca/~carenini/TEACHING/CPSC503-09/FINAL-REPORTS-08/hammad-report1.1.pdf (Accessed: Jan. 10. 2021).

[14] K. Stratos, M. Collins, and D. Hsu, "Unsupervised part-of-speech tagging with anchor hidden markov models," *Trans. Assoc. Comput. Ling.*, vol. 4, pp. 245–257, 2016, doi: 10.1162/tacl_a_00096.

[15] K. Sarkar and V. Gayen, "A trigram HMM-based POS tagger for Indian languages," in *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, 2013, pp. 205–212.

[16] M. Banko and R.C. Moore, "Part of speech tagging in context," in *Proc. 20th International Conference on Computational Linguistics*, 2004, 556, doi: 10.3115/1220355.1220435.

[17] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," 2015. [Online]. Available: https://arxiv.org/abs/1508.01991 (Accessed: Jan. 10. 2021).

[18] M. Thayaparan, S. Ranathunga, and U. Thayasivam, "Graph Based Semi-Supervised Learning for Tamil POS Tagging." FIRE 2014, [Online]. Available: https://aclanthology.org/L18-1624.pdf (Accessed: Jan. 10. 2021).

[19] B. Plank, A. Søgaard, and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," in *Proc. 54th Annu. Association for Computational Linguistics*, 2016, pp. 412–418.

[20] M. Rajasekar and A. Udhayakumar, "POS Tagging Using Naive Bayes Algorithm For Tamil," *Int. J. Sci. Eng. Technol. Res.*, vol. 9, pp. 574–578, Feb. 2020.

[21] J. Singh, L. Singh Garcha, and S. Singh, "A Survey on Parts of Speech Tagging for Indian Languages," *Int. J. Adv. Res. Comput. Sci. Software Eng*., vol. 7, no. 4, Apr. 2017.

[22] V. Dhanalakshmi, A.M. Kumar, and K.P. Soman, and S. Rajendran, "POS Tagger and Chunker for Tamil Language," *Proceedings of the 8th Tamil Internet Conference*, Cologne, Germany, 2009.

[23] K.K. Akhil, R. Rajimol, and V.S. Anoop, "Parts-of-Speech tagging for Malayalam using deep learning techniques," *Int. J. Inf. Technol*., vol. 12, pp. 741–748, 2020, doi: 10.1007/s41870-020-00491-z.

[24] E. Lukasik *et al*., "Recognition of handwritten Latin characters with diacritics using CNN," *Bull. Pol. Acad. Sci. Tech. Sci*., vol. 69, no. 1, p. e136210, 2021, doi: 10.24425/bpasts.2020.136210.

[25] D. Andor *et al*., "Globally normalized transition-based neural networks," in *Proc. 54th Annu. Association for Computational Linguistics*, Berlin, Germany, 2016, pp. 2442–2452.

[26] M. Yan *et al*., "A deep cascade model for multi-document reading comprehension," in *Proc. of The Thirty-Third AAAI Conference on Artificial Intelligence*, 2018, pp. 7354–7361.

[27] P. Wang, Y. Qian, F.K. Soong, L. He, and Z. Hai, "Part-of-speech tagging with bidirectional long short-term memory recurrent neural network," [Online]. Available: https://arxiv.org/abs/1510.06168v1

[28] Keras, [Online] Available: https://keras.io/ (Accessed: 30.03.21).