S. Kumar*, M. Anand Kumar and K.P. Soman

Deep Learning Based Part-of-Speech Tagging for Malayalam Twitter Data (Special Issue: Deep Learning Techniques for Natural Language Processing)

https://doi.org/10.1515/jisys-2017-0520
Received September 28, 2017; previously published online June 6, 2018.

Abstract: The paper addresses the problem of part-of-speech (POS) tagging for Malayalam tweets. The conversational style of posts/tweets/text in social media data poses a challenge in using general POS tagset for tagging the text. For the current work, a tagset was designed that contains 17 coarse tags and 9915 tweets were tagged manually for experiment and evaluation. The tagged data were evaluated using sequential deep learning methods like recurrent neural network (RNN), gated recurrent units (GRU), long short-term memory (LSTM), and bidirectional LSTM (BLSTM). The training of the model was performed on the tagged tweets, at word level and character level. The experiments were evaluated using measures like precision, recall, f1-measure, and accuracy. During the experiment, it was found that the GRU-based deep learning sequential model at word level gave the highest f1-measure of 0.9254; at character-level, the BLSTM-based deep learning sequential model gave the highest f1-measure of 0.8739. To choose the suitable number of hidden states, we varied it as 4, 16, 32, and 64, and performed training for each. It was observed that the increase in hidden states improved the tagger model. This is an initial work to perform Malayalam Twitter data POS tagging using deep learning sequential models.

Keywords: Part-of-speech tagging, deep learning, recurrent neural network, long short-term memory, gated recurrent unit, bidirectional LSTM.

1 Introduction

Every day, enormous volume of web content is being created and published in the Internet through blogs, microblogs, social media websites (Facebook, YouTube), social networking services (Twitter), search queries, multimedia content, opinion sites, advertisements, news, discussion forums, etc. These data contain information related to news (politics, sports, products, etc.), reviews or opinions (movies, products, etc.), stock data, technical contents, individual's views on events, etc., and they are commonly known as social media data. Social media has a huge audience generating and accessing huge volume of data every day, providing simplex mode communication that can instantly reach people in geographically separated locations [41]. Microblogging has emerged as a single point to obtain different types of information, as it had become a platform where people share their opinions or put post or real-time messages. Twitter [55] is one of the popular social media services. It is the single largest microblog where gigabytes of information are shared. Twitter data analysis finds its importance when mining it can provide the opinion change over time, identifying the current trend and shift, current topic of discussion, sentiment analysis of tweets and re-tweets, visual analytics of network, text summarization, etc. The characteristics of text in Twitter format are short messages (known as "tweets" – 280-character-long messages), conversation in thread format where URLs can be included, topic markers, username mentions, hashtags, re-tweets, emoticons, unorthodox capitalization, abbreviations, etc. [2, 17, 18, 64]. The challenge posed by the tweet is to interpret text in noisy form, that is grammatically incorrect, that irregularly uses capital letters, that contains many whitespaces/punctuations,

***Corresponding author: S. Kumar,** Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India, e-mail: sachinnme@gmail.com
M. Anand Kumar and K.P. Soman: Center for Computational Engineering and Networking (CEN), Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India

etc. These challenges motivate researchers in identifying methods to analyze tweets properly. In this direction, machine-learning-based mining of social media content is gaining large attention from the research community [50, 56].

The aforementioned task becomes more challenging when the tweets are in non-English language, such as tweets in Malayalam language (a low-resource language), considered for study in the present paper. Tasks like part-of-speech (POS) tagging, named entity recognition, parsing, etc., will fail in understanding the social media text as it differs in orthographic, syntactic, and lexical patterns followed in natural language text. As POS tagging for tweets in Malayalam is not available, the current paper proposes a POS tagging for Malayalam tweets using deep learning sequential models. Malayalam is a low-resource language in terms of tools for understanding natural language text with its applications and social media text. It is a language largely spoken in the state of Kerala, India, and there are millions of texts in social media in Malayalam [66]. Hence, this paper also derives its motivation as an initial attempt to come up with a basic POS tagging tool for Malayalam Twitter data. The initial popular work on Twitter POS tagging for English Twitter data was done by Gimpel et al. [25]. A coarse POS tagset was designed for Twitter data in English to capture the main parts of speech like verb, noun, etc., then the data were manually tagged and evaluated. In order to include the commonly seen URL formats, hashtags, "@," etc., specific tags are devised as in Ref. [25]. Toward the direction of POS tagging of social media data in Indian language, the research work started for Hindi, Bengali, and Telugu [30, 32, 65]. The experiments were conducted on both coarse- and fine-grained tagsets. The coarse-grained tagset combined the Gimpel et al. tagset and the Google universal tagset [25, 49]. For fine-grained tagset along with Gimpel et al.'s, the authors combined it with Indian language tagsets such as IL-POST, LDCIL, and TDIL tags [5, 40]. The present paper adopted the coarse tagset mentioned in Refs. [30, 32, 65] for manually tagging 9915 tweets in Malayalam language. The POS tagset used for annotation of Twitter data in Malayalam language is shown in Table 1. The examples for each tag are shown in romanized form for corresponding words in Malayalam. Table 2 shows the number of tokens. Deep learning based methods have shown their effectiveness for natural language processing (NLP) tasks [13, 14, 26]. Traditional methods require features mainly based on linguistic knowledge, with few trial-and-errors for selecting the features. Preparing complex features also increase the computational cost. Deep learning methods provide a different approach where the features are prepared by the neural network itself. The current paper provides an experimental evaluation performed using deep learning methods such as recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU), and bidirectional LSTM (BLSTM). The major contributions made in this paper are as follows:

– A coarse-grained POS tagset for Malayalam Twitter data was designed.
– A total of 9915 tweets (85,404 tokens) were manually annotated.
– An annotated dataset was evaluated using deep learning sequential methods.

The paper is organized as follows: Section 2 provides an extensive survey on POS tagging based on literature for Indian and other languages. Section 3 describes the materials and methods used in this work. Section 4 presents the experimental evaluations and results. Section 5 concludes this paper.

2 Related Works

POS tagging is a well-known NLP task that has been investigated for decades. For text in natural language, several POS tagging models were proposed in the past using rules, Markov methods, kernel methods, hybrid methods, etc. [4, 8–10, 15, 20, 22, 24, 26, 43, 54, 58, 59]. In Ref. [59], the authors proposed a trigram hidden Markov model (HMM)-based POS tagger for Indian languages such as Bengali, Hindi, Marathi, and Telugu. The approach finds the POS tag corresponding to the raw text data. The authors introduced a word-type and prefix analysis method with suffix analysis to find probable tags. In Ref. [58], the authors presented a conditional random field (CRF)-based POS tagging for code-mixed Indian social media text as part of shared task on data such as Hindi-English, Bengali-English, and Telugu-English. Previous studies have also discussed deep-learning-based POS tagging. In Ref. [53], the authors discussed RNN-based POS tagging on Chinese Buddhist

Table 1: Tagset for Tweets in Malayalam with Examples.

Sl. no	Tag name	Description	Examples
1	CC (conjunction)	Conjunction	eNKiluM, pakshe, ennAl (if so, but, though)
2	DM (demonstrative)	Demonstrative	itANU, I, atuM (this, this, that also)
		Demonstrative Wh-word	ss entO (what)
3	JJ (adjective)	Adjective	ponne, kaZinja (dear, over)
4	N_NN (noun)	Noun	divasam, lokkESaNRe, kAryaM (day, location's, thing)
		Proper noun	kOLEjU, bhAratatte (college, about India)
		Verbal noun	samipaM, saMagamaM (near, meet)
		Nloc	toTTappuRatte (nearby)
5	PR (pronoun)	Pronoun reciprocal	tammilttammil (among)
		Pronoun reflexive	ennetanne (myself)
		Pronoun relative	Arokke, maRRuLLaNRe (who are all, others)
		Pronoun personal	njANKanTU, eNRe, njAn (I saw, mine, I)
		Pronoun Wh-word	eviTe, entANU (where, what)
6	PSP (postposition)	Postposition	enna, pOluM, tanne (that, like-wise, self)
7	QT (quantifiers)	Quantifiers cardinal	raNTU, oru, pathineTTU (two, one, eighteen)
		Quantifiers general	vERE, anEka, anavaDi (many others, many)
		Quantifiers ordinal	irupattiyonnAmatU (twenty-first)
8	RB (adverb)	Adverb	angane, engane (like-wise, like-this)
9	RD (residuals)	Echo words	vEgaM, vEgaM (fast)
		Punctuation	:,.
		Foreign word	Non-Malayalam word
		Symbol	(,)
		Unknown	?, ?
10	RP (particles)	Interjection	illallO, allE, ayyO (nothing, is it not, oh)
		Intensifier	mikaCa, ERRavuM (excellent, most)
		Negation	alla, mAtRamalla, illAtta (not, not only, not having)
11	V_AUX	Verb auxiliary	ANU (is)
12	V (verb main)	Verb main finite	karanju, vannAluM, tOnnaM (cried, if comes, may be)
		Verb main non-finite	naraCa, munnERunna (grayish, going forward)
		Verb main infinitive	pokAn, pirikkAn (to go, to split)
		Verb main gerund	kANunnatU, nalkiyatU, viLiCatU (that which is seen, that which is given that which is called)
13	HT	Twitter-specific tags	#
14	@		@
15	URL		http or https
16	&		&
17	Emoticons		:], :), :-), :p

Table 2: Number of Tokens Corresponding to Each Tag.

Tag name	N_NN	V_VM	V_AUX	CC	JJ	DM	PR	PSP	QT	RB	RD	RP	U	@	#	&	em
Count	38,014	18,356	17	304	4690	5012	3635	1757	2418	174	1367	2113	61	3437	105	3930	14
%	0.445	0.2149	0.0002	0.0035	0.055	0.0586	0.0425	0.0189	0.0283	0.002	0.016	0.0247	0.0007	0.0402	0.0012	0.046	0.0002

text. The model was shown to be more effective than the HMM model. A bidirectional LSTM-based approach is discussed in Ref. [52], which explores the effectiveness of BLSTM representation and an auxiliary loss for sequence modeling tasks such as POS tagging. The model was evaluated on 22 different languages, at word, character, and byte levels. In Ref. [42], the authors focused on creating a word representation through character composition using BLSTM. The authors experimentally evaluated the model on POS tagging for several languages. In Ref. [57], a deep neural network model combining word- and character-level representation for POS tagging is discussed. A convolutional layer prepares the character representation. The model proved effective for POS tagging English and Portuguese languages. In Refs. [11, 35], the authors discussed character-level deep learning representation from the raw text [23] and a byte-level LSTM-based POS tagging task. The most popular paper and initial work related to the Twitter POS tagger is Ref. [25], in which a coarse POS tagset was designed for Twitter data in English. To capture the main POS like verb, noun, etc., the data were manually tagged and evaluated. In order to include the commonly seen URL formats, hashtags, and "@," specific tags were devised. The "@" and hashtags will come along with words. The modeling was done using CRF with several features like capitalization, suffix, presence of digit, etc. Regular expressions were used for detecting "@," "#," word with capitals, etc. In Ref. [46], a CRF-based POS tagger for a French social media POS tagger is discussed. The paper reports an accuracy of 91% on various data collected from Facebook, forums, Twitter, etc. The authors proposed a tagset with 28 tags. Another paper [47] presents an unsupervised word cluster to obtained improved accuracy than that reported in Ref. [25]. The authors have released another dataset in English with annotations.

There have been significant amounts of work related to general POS tagging tasks in Indian languages – Hindi, Oriya, Marathi, Punjabi, Bengali, Kannada, Malayalam, Tamil, Telugu, etc. [1, 3, 16, 19, 29, 37, 38, 45, 61, 62, 67]. In Ref. [65], the authors designed a POS tagger for code-mixed data in English-Hindi (bilingual) collected from forums, Facebook, etc. The authors annotated data and addressed problems like transliteration, language identification, non-standard spelling, and normalization. The corpus had 6983 posts. This is the initial work that focused on transliteration in social media data in Indian language context. The authors concluded that transliteration and normalization are challenging in code-mixed data. POS tagging for Hindi and Bengali tweets are presented in Ref. [30]. The authors proposed a POS tagset with 38 fine-grained and 12 coarse-grained tags for Hindi, standardized by LDC-IL. The corpus contains tweets of news and collected a total of 3488 tweets. The features used were current word, previous three words and the tags, and first four characters of a word. They trained with 1300 tweets and provided an evaluation using different methods like naive Bayes, support vector machine (SVM) and random forest (RF). In Ref. [32], the authors discussed code-mixed bilingual POS tagging, English-Hindi, in social media data collected from Facebook and Twitter posts. The tagging was performed using both fine-grained and coarse-grained tags. Their corpus contained 1236 Facebook posts and 4435 tweets. The tagsets used were a mixed group from Gimpel et al. [25], IL-POST tags [5], and TDIL [39]. For coarse-grained tags, the authors used Gimpel et al.'s tagset [25] and the universal tagset from Google [49]. The authors provided an evaluation and comparison of different POS tagger models created using machine learning algorithms like CRF, naive Bayes, RF, and sequential minimal optimization. The features used were current word, prefix, suffix, next and previous word, presence of digit, and language of current word. An RNN-based POS tagging on code-mixed data were proposed in Ref. [48] on corpus provided as part of a shared task [31]. Another paper [51] presented a POS tagging on code-mixed social media data in Hindi, Telugu, and Bengali. The Stanford POS tagger was used for tagging. The features used were current word, previous word, next word, POS tags of previous two words, and word position in the sentence. Evaluations were performed using Weka tool, which obtained a comparatively better accuracy for the J48 decision tree method. In Ref. [36], the authors proposed a POS tagging for Tamil data using BLSTM. A BLSTM-based character-to-word embedding model was used for representing word. The approach obtained an accuracy of 86.45% in testing.

The present work derives its motivation from similar works on social media data in Indian languages. The lack of corpus in Malayalam social media data encouraged creating an annotated corpus with 9915 tweets. The paper is the first of its kind to report on Malayalam social media Twitter data tagging. The studies in this area are mostly in English, as discussed previously. However, the different language structures make each research work important. As the Malayalam language contains many inflections and conjugations, the current work aims to give a start in this direction.

3 Materials and Methods

Deep learning currently is the buzzword in machine learning. It is a data-hungry approach to generate better features than hand-designed features as in traditional methods (decision tree, SVM, etc.). In case of text data, deep learning sequential models are better than feed-forward methods, as they suffer from fixed context length [7, 14, 33, 44, 60]. The current paper uses deep learning sequential methods such as RNN, LSTM,

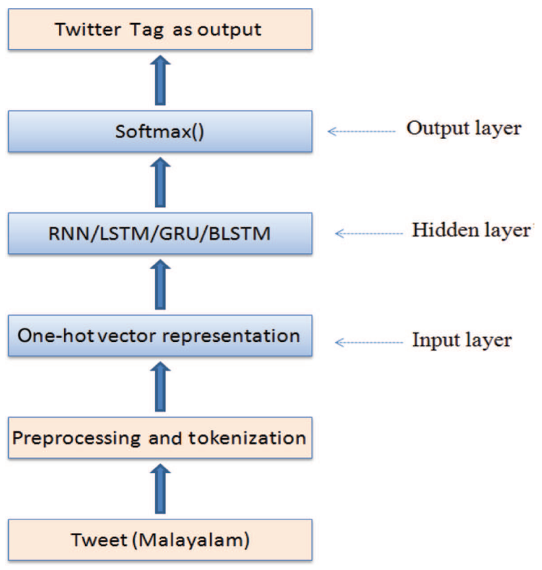

Figure 1: Deep Sequential Model-Based POS Tagging.

GRU, and BLSTM to evaluate the manually tagged Malayalam POS tagged tweets. Figure 1 shows the simple architecture for the proposed deep learning based approach for Malayalam Twitter data POS tagging. Initially, the tweet in Malayalam is passed to the preprocessing and tokenization stage. All the URLs, @-mentions, and #-mentions are replaced with U, @, and #. The emoticons are kept as such using regular expressions (basic level preprocessing). All repeating symbols, like !!!!, ?????, and ..... are reduced to !, ?, and .. The left and right context of a token is considered to find its POS tag. A vector for these three tokens is prepared by taking the one-hot vector of each token. In this manner, data are prepared and fed to the next training stage with RNN, LSTM, GRU, and BLSTM. Softmax is chosen as the classifier. The realization of each network architecture for the experiment is done using Keras [34].

3.1 RNN

RNN is an extension of feed-forward network (standard multilayer perceptron) that can manage sequences of variable length via hidden recurrent states connected in a loop (connected to themselves for feedback) [21]. It provides a simple way to handle data in sequential form $x_1, x_2, x_3 \ldots x_n$. In sequential data, the length or number of elements in the sequence (tokens, numerical values, etc.) will not be same always. In RNN, the output of one state is fed back as input to it, forming a loop or feedback. This can be seen as passage through time and acts as a small memory of things that happened till that time. The hidden state h_t is calculated based on the current input x_t and the previous hidden state h_{t-1}. That is,

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

$$h_t = f(Ax_t + Wh_{t-1}). \tag{2}$$

The function f is generally a non-linear function (also known as an activation function) like tanh(), rectified linear units, etc. Despite the success of RNN, it encounters vanishing gradient issues (gradients decay over time) when the sequence to learn is long [6]. In order to address this issue, gated RNNs were introduced. In the gated RNN method, different gates are introduced to control the existence and flow of information.

3.2 LSTM

LSTM is a variant of RNN with a gating mechanism proposed in 1997 [27] to capture long dependencies. It can be viewed as a single block known as a memory cell with two inputs to it – previous hidden state and current input. Internally in the memory cell, it decides which information needs to persist or which does not. It is a well-explored method for various applications in several domains. The basic equations to compute the hidden state at time step t are same as in Eq. (2). LSTM computes using three gates, namely input gate, forget gate, and output gate, and cell state estimation. The basic equations are as follows:

$$i_t = \sigma(A^i x_t + W^i h_{t-1}). \tag{3}$$

$$fg_t = \sigma(A^{fg} x_t + W^{fg} h_{t-1}). \tag{4}$$

$$o_t = \sigma(A^o x_t + W^o h_{t-1}). \tag{5}$$

$$\hat{c}_t = \tanh(A^c x_t + W^c h_{t-1}). \tag{6}$$

$$c_t = \hat{c}_t \circ i_t + c_{t-1} \circ fg_t. \tag{7}$$

$$h_t = o_t \circ \tanh(c_t) \tag{8}$$

Here, i, fg, o, c, input gate, forget gate, output gate, and cell state are all vectors that have the same dimension as that of the hidden state vector.

3.3 BLSTM

In LSTM, the information flows forward, in a looped structure. In BLSTM, the network captures the structure of the sequenced data in both forward and backward directions to capture the context information for longer time steps [60]. It has shown promising results in NLP applications like language modeling and sequence labeling [30, 65]. For the experiment, the same LSTM architecture is used by passing the sequence in forward and reverse directions. The output of both LSTMs is merged or concatenated at the end before applying the $softmax()$ function.

3.4 GRU

GRU provides a simple gating function compared to LSTM [12]. It has two types of gates – update and reset – as shown in Eqs. (9) and (10). The reset gate takes input from the previous hidden state and current input, and computes a sigmoid function over it. The equations are as follows:

$$u_t = \sigma(A^u x_t + W^u h_{t-1}). \tag{9}$$

$$r_t = \sigma(A^r x_t + W^r h_{t-1}). \tag{10}$$

The effectiveness of GRU in capturing long dependencies is well explored in Ref. [16].

3.5 Training and Softmax Classifier

The parameters that need to be learned in the aforementioned architectures are generally known as weight matrices. The values of the parameters are decided using a cost function. The cost function's error is minimized (with respect to parameters) via a gradient-descent method. The weights are updated as the error's gradient, $-\frac{\partial E}{\partial w}$, of cost function reduces. That is, if the weight parameter is perturbed, the amount of error is affected. The negative sign denotes the direction in which error reduces. The learning occurs via propagating the gradients from error. Learned weights give better classification. For problems like sequence classification, cross-entropy loss function is commonly used and is defined as

$$H_{y}^{'}(y) := -\sum_{i} y_{i}^{'} \log(y_{i}), \tag{11}$$

where y_i is the probability distribution of the i^{th} predicted class and $y_i^{'}$ is the true probability distribution. The loss function is a measure about the error between predicted and true labels. The output layer of the neural network is given to $softmax()$. It provides the probability for tags corresponding to the word w_i given for testing. The estimate for the tag corresponding to the word w_i is found as

$$\hat{y}_i = \frac{\arg\max}{t \in 1, 2, ..n} P_i(t|w_1, w_2, w_3 ... w_m). \tag{12}$$

4 Experiment and Results

For the experiment and evaluations, the code was implemented using the Theano framework, a deep learning library [63]. The parameter optimization was performed using the Hyperas package [28]. The dataset used for the evaluation was manually annotated using the defined tagset, as shown in Table 1. A total of 9915 Malayalam tweets were tagged, and the distribution of dataset for the cross-validation experiment is given in Table 3 The average was obtained by dividing "tokens" with "count" values. The current work is the first of its kind (i) to perform a POS tagging task on tweets in Malayalam language and (ii) to perform deep learning based sequence modeling and its evaluations. A 10-fold cross-validation approach was used for the experiments with RNN, LSTM, GRU, and BLSTM.

The evaluations were performed at word and character levels. At word level, each word and the immediate left and right words as context (a sequence with three tokens) were considered to form the initial vector. The dimension of the vector for each w_i is the same as that of the vocabulary size $|V|$. In character level, the three-token sequence was converted into its corresponding characters forming a sequence of characters $c_1, c_2, c_3, \ldots, c_n$. In the character sequence, each c_i is represented as a one-hot vector where n denotes the number of characters in the three-word sequence. Character-level language modeling has generated huge interest recently [35]. The word- and character-level approaches are trained and evaluated using deep learning sequential models such as RNN, LSTM, GRU, and BLSTM. Throughout the experiment, a single-layer architecture was followed. The numbers of hidden states chosen were 4, 16, 32, and 64. In order to compute loss function, the most suitable for multi-class sequence classification is "cross-entropy." The number of epochs for the experiment was fixed as 100. The learning rate was 0.01, which gave the best result. In order to improve training, a dropout parameter was used. The dropout parameter introduces regularization to prevent model overfit. The model was evaluated using metrics such as precision, recall, f1-measure, and accuracy. From Table 2, it is known that the number of data per class was uneven. Hence, the f1-measure provides better understanding than accuracy about the classification, as the latter can only give information about correctly classified observations. When the number of data in each class is even, then the accuracy is a good metric.

4.1 Evaluation at Word Level

Even though LSTM, GRU, and BLSTM showed improved metrics, the highest metric was obtained for GRU with 64 hidden states. It obtained an f1-measure of 0.9254. The highest f1-measure for experiments using

Table 3: Tweets and Tokens in the Dataset Considered for Experiments.

Tweet	Count	Tokens	Average
Total	9915	85,404	8.6
Training	9000	77,860	8.6
Testing	915	7544	8.2

Table 4: Evaluation Measures at Word Level for Different Deep Learning Methods.

Method	Precision	Recall	f1-Measure	Accuracy
RNN_4	0.6784	0.6504	0.6496	0.6504
RNN_16	0.7214	0.7061	0.7132	0.7061
RNN_32	0.8023	0.7743	0.7851	0.7743
RNN_64	0.9081	0.8911	0.8965	0.8911
LSTM_4	0.8572	0.8455	0.8446	0.8455
LSTM_16	0.9193	0.8691	0.8835	0.8691
LSTM_32	0.9242	0.9128	0.9162	0.9128
LSTM_64	0.9273	0.9255	0.9252	0.9255
GRU_4	0.8871	0.8252	0.8476	0.8252
GRU_16	0.9272	0.8745	0.8943	0.8745
GRU_32	0.9172	0.9147	0.9147	0.9147
GRU_64	0.9282	0.9258	0.9254	0.9258
BLSTM_4	0.8789	0.8580	0.8629	0.8580
BLSTM_16	0.9151	0.9021	0.9058	0.9021
BLSTM_32	0.9139	0.9019	0.9052	0.9019
BLSTM_64	0.9144	0.9126	0.9127	0.9126

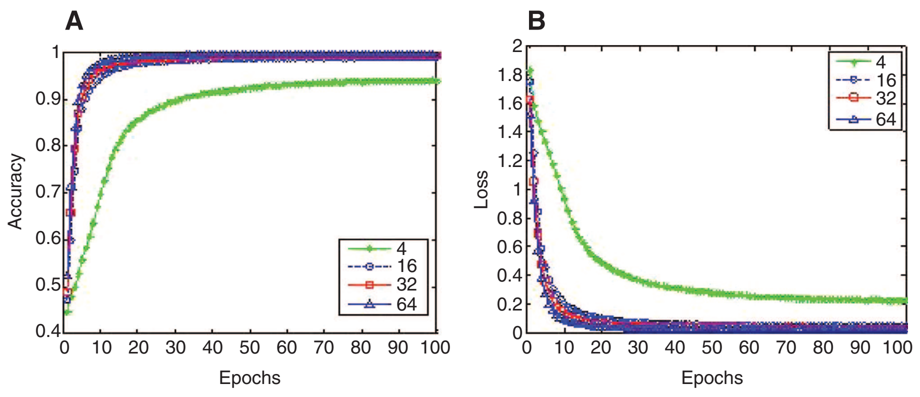

Figure 2: Performance of GRU with 4, 16, 32, and 64 Hidden States. (A) Accuracy vs. epochs; (B) loss vs. epochs.

LSTM was 0.9252 and for those using BLSTM was 0.9127, for 64 hidden states at word-level evaluation. This can be observed from Table 4. As GRU scored high in evaluation, its performance during training can be understood using Figure 2. It shows the change in accuracy and loss with respect to epochs during training of GRU with 4, 16, 32, and 64 hidden states. It can be observed that as the number of epochs increased, the accuracy during training also exhibited an increase and it crossed 0.9. From the plot of loss vs. epoch in Figure 2B, it can be observed that the slope decreased rapidly for 16, 32, and 64 hidden states. The decay in loss occurred faster. The magnitude of the loss decreased from 1.8413 to 0.0465. Loss value gives information on how well the model is learning. Decrease in loss shows improved learning by the model with all the fixed sets of parameters. It can be observed from Figure 3 that all methods showed improvement for hidden state number chosen as 64. The 10-fold cross-validation accuracy obtained for each method with 64 hidden states was >0.9. This shows that the word-level training did not learn an overfitted model.

4.2 Evaluation at Character Level

The precision, recall, f1-measure, and accuracy obtained for RNN, LSTM, GRU, and BLSTM of testing performed at character level are shown in Table 5. It can be observed from Table 5 that the error reduced as the number of hidden states increased, within each deep learning method. However, when compared to the word-level results, reduction in error was less during testing at character level. The highest evaluation score was obtained for BLSTM with 64 hidden states. It obtained an f1-measure of 0.8739. Figure 4 shows the change

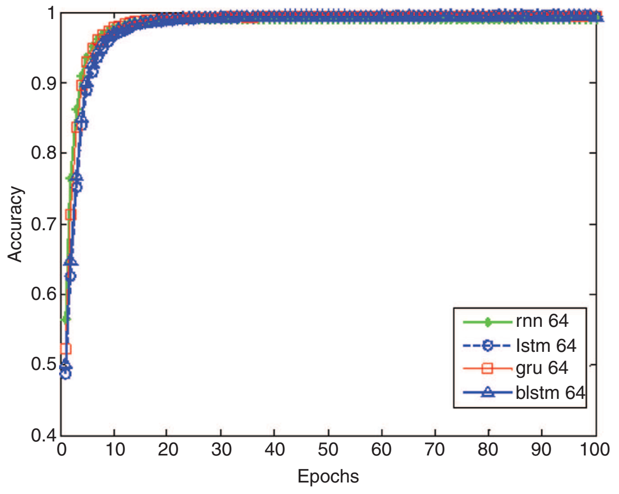

Figure 3: Highest Accuracy Obtained for Each Deep Learning Method at Word Level.

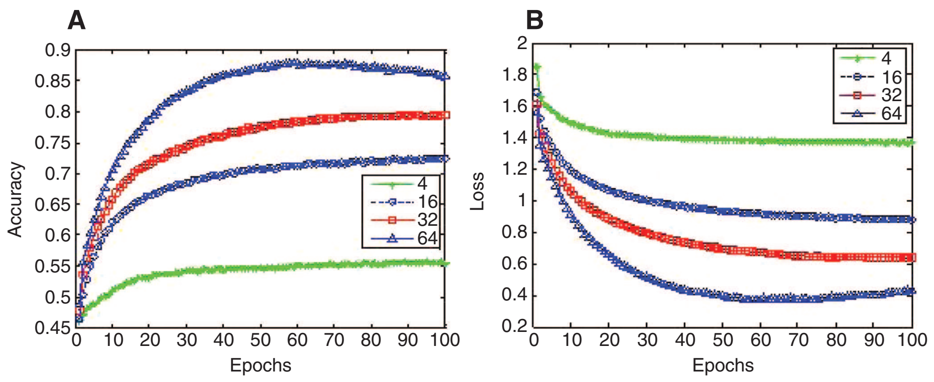

Figure 4: Performance of BLSTM with 4, 16, 32, and 64 Hidden States. (A) Accuracy vs. epochs; (B) loss vs. epochs.

in accuracy and loss with regard to epochs during training of BLSTM (as it obtained better scores in the evaluation) at character level with 4, 16, 32, and 64 hidden states. It can be observed that as the number of epochs increased, the accuracy during training also exhibited an increase. The plot of loss values corresponding to the BLSTM with 64 hidden states shows a better learning rate as the decay is gradual in nature. LSTM and GRU with 64 hidden states obtained f1-measures of 0.8632 and 0.8332. RNN showed poor evaluation metrics, as it failed to capture the context dependencies properly (sequence length will be more) even when the hidden state increased. Figure 5 shows the accuracy vs. epoch plot for all the models at character level, which obtained high accuracy with the corresponding hidden states. Even though character-level sequential modeling is a promising approach, in the current work, the results show that parameter tuning is required to improve the outcome. During testing of character-level models, LSTM, GRU, and BLSTM showed results >0.8 for 64 hidden states. In the current experiment with varying number of hidden states, comparing character-level and word-level approaches, the latter was found better in performing POS tagging for Malayalam Twitter data.

It was observed from the experiment that the methods were able to identify the nouns [e.g. karNATakattil (in Karnataka), vyAzAzcca (Thursday), sUryanE (sun), kOLEju (college), peTrOL (petrol)] and verbs [kettaTu (heard), kiTTUla (wont get), tOnnunnu (think so), vannatu (came)]. The presence of unwanted symbols adds to the misclassification. A few examples that were commonly misclassified at both levels are toNTi (N_NN, JJ), grAnRAkkaNam (V, RB), marIccEnE (V, N), veTiyERRu (RB, V), ninnu (PSP, N), RERRu (DM, QT), etc. In the examples, brackets associated with each token contained the correct POS tag and the wrong POS tag obtained during the experiment. This could happen due to the absence of data with these words. A few examples of

Table 5: Evaluation Measures at Character Level for Different Deep Learning Methods.

Method	Precision	Recall	f1-Measure	Accuracy
RNN_4	0.2594	0.4328	0.3068	0.4328
RNN_16	0.4345	0.4689	0.4288	0.4689
RNN_32	0.4577	0.4734	0.3493	0.4734
RNN_64	0.2321	0.0678	0.0153	0.0678
LSTM_4	0.5171	0.5769	0.5204	0.5769
LSTM_16	0.7151	0.7209	0.6969	0.7209
LSTM_32	0.7866	0.7898	0.7804	0.7898
LSTM_64	0.8648	0.8662	0.8632	0.8662
GRU_4	0.5089	0.5585	0.5061	0.5585
GRU_16	0.7042	0.7071	0.6828	0.7071
GRU_32	0.7689	0.7727	0.7648	0.7727
GRU_64	0.8342	0.8354	0.8332	0.8354
BLSTM_4	0.5437	0.6013	0.5631	0.6013
BLSTM_16	0.7572	0.7572	0.7460	0.7572
BLSTM_32	0.8258	0.8259	0.8208	0.8259
BLSTM_64	0.8748	0.8757	0.8739	0.8757

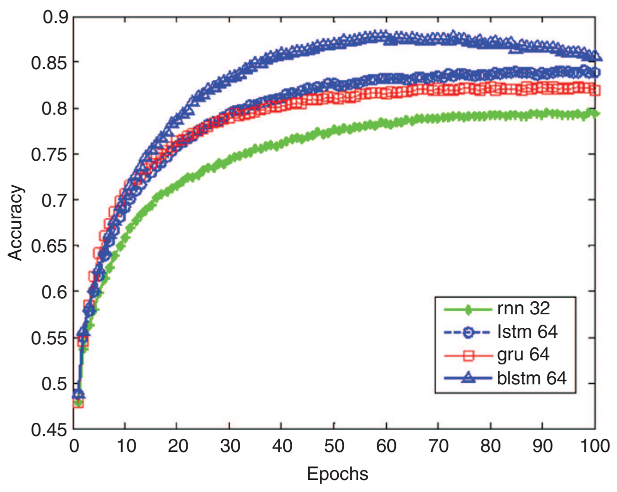

Figure 5: Highest Accuracy Obtained for Each Deep Learning Method at Character Level.

POS tagged outputs by the networks are shown in bullets below. The tags inside brackets are the wrong predicted tags. The bold marking indicates the networks that predicted wrong tags. The present paper intended to provide a start in this research direction. However, more interpretable analysis is considered to be included as future work, along with more data and fine tuning of networks.

ASayangaLkkum/N_NN(RB,V) cinthAgadikkum/N_NN(V) kOTTam/RB(N_NN) taTTAte/V(DM) sheyar/N_NN(V) ceyyu/V (GRU, **BLSTM, LSTM, RNN**).

- jInsu/N_NN parAmarSam/N_NN(V) mAdhyamangaLe/N_NN parihasiccu/V(RB) jOyimAtyu/N_NN http://t.co/E8yVbGdeaI/U (GRU, BLSTM, LSTM, **RNN**).
- kOLEju/N_NN innu/DM(V) sauhrdangaL/N_NN(RB) nalkiya/V comfarTTuleval/N_NN onnum/RP(QT) ini/N_NN eviTe/PR(RB) pOyAlum/V(RB) kiTTUla/V (**GRU, BLSTM, LSTM, RNN**).

5 Conclusion and Future Work

The current study is the first one to perform Malayalam Twitter POS tagging. The present paper proposes a deep learning approach for Malayalam Twitter POS tagging and provides a comparison among the deep learning sequential models to find the suitable method for Malayalam Twitter POS tagging at word level and character level. The corpus consisted of 9915 tweets in Malayalam language. Gaining motivation from the works of Gimpel et al. [25] and Jamatia and Das [30] on Twitter data POS tagging, a tagset for Malayalam Twitter data was designed. To perform experiments, sequential model algorithms such as RNN and its variants – LSTM, GRU, and BLSTM – were used. The experiment was conducted at word level and character level for four different hidden state numbers: 4, 16, 32, and 64. For evaluating the experiments, the following metrics were used: precision, recall, f1-measure, and accuracy. From the evaluations at word level, it was found commonly that as the hidden state parameter increased, the metrics used for evaluation increased. Among the four sequential deep learning models, the highest metric was obtained for GRU with 64 hidden states at word level. It obtained an f1-measure of 0.9254; 0.9258 for accuracy, 0.9282 for precision, and 0.9258 for recall. The training at character level was performed by converting the word sequence into its corresponding character sequence. A similar experiment and evaluation performed for the word-level approach were repeated for the character-level approach. Among the four sequential models, BLSTM obtained the highest evaluation metrics. It obtained an f1-measure of 0.8739; 0.8757 for accuracy, 0.8748 for precision, and 0.8757 for recall. The experiment revealed that for word-level Malayalam Twitter POS tagging, GRU with 64 hidden states was found competing with the rest of the sequential deep learning models. For performing Malayalam Twitter POS tagging at character level, BLSTM with 64 hidden states showed competing results. The present work is a beginning in Malayalam POS tagging. As a future work, the proposed approaches can be utilized to perform an evaluation on a larger corpus. To explore the semantic role labeling problem, information extraction, the tagger can be improved by incorporating the morphological aspects of Malayalam tweets and designing a fine-grained tagset.

Bibliography

[1] A Part of Speech Tagger for Indian Languages (POS tagger), Tagset developed at IIIT-Hyderabad after consultations with several institutions through two workshops, 2007.

[2] A. Abbasi, H. Chen and A. Salem, Sentiment analysis in multiple languages: feature selection for opinion classification in web forums, *ACM Trans. Inform. Syst.* **26** (2008), 12–34.

[3] P. J. Antony and K. P. Soman, Kernel based part of speech tagger for Kannada, in: *Proceedings of International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 4, pp. 2139–2144, 2010.

[4] P. J. Antony, S. P. Mohan and K. P. Soman, SVM based part of speech tagger for Malayalam, in: *IEEE International Conference on Recent Trends in Information, Telecommunication and Computing (ITC)*, pp. 339–341, 2010.

[5] S. Baskaran, K. Bali, T. Bhattacharya, P. Bhattacharyya, M. Choudhury, K. V. Subbarao and G. N. Jha, A common parts-of-speech tagset framework for Indian languages, in: *Proceedings of the 6th International Conference on Language Resources and Evaluation*, pp. 1331–1337, 2008.

[6] Y. Bengio, P. Simard and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Netw.* **5** (1994), 157–166.

[7] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, A neural probabilistic language model, *J. Mach. Learn. Res.* **3** (2003), 1137–1155.

[8] Y. Bengio, A. Courville and P. Vincent, Representation learning: a review and new perspectives,*IEEE Trans. Pattern Anal. Mach. Intell.* **35** (2013), 1798–1828.

[9] E. Black, F. Jelinek, J. Lafferty, R. Mercer and S. Roukos, Decision tree models applied to the labeling of text with parts-of-speech, in: *Proceedings of the Workshop on Speech and Natural Language*, pp. 117–121, 1992.

[10] T. Brants, TnT: a statistical part-of-speech tagger, in: *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pp. 224–231, 2000.

[11] G. Chrupala, Text segmentation with character-level text embeddings, in: *Proceedings of the ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[12] J. Chung, C. Gilcehre, K. Cho and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555, 2014.

[13] R. Collobert and J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, in: *Proceedings of the 25th International Conference on Machine Learning*, pp. 160–167, 2008.

[14] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, Natural language processing (almost) from scratch, *J. Mach. Learn. Res.* **12** (2011), 2493–2537.

[15] W. Daelemans, J. Zavrel, P. Berck and S. Gillis, MBT: a memory-based part of speech tagger-generator, in: *Proceedings of the Fourth Workshop on Very Large Corpora*, pp. 14–27, 1996.

[16] A. Dalal, K. Nagaraj, U. Sawant, S. Shelke and P. Bhattacharyya, Building feature rich POS tagger for morphologically rich languages: Experiences in Hindi, in: *Proceedings of International Conference on Natural Language Processing (ICON)*, 2007.

[17] L. Derczynski, A. Ritter, S. Clark and K. Bontcheva, Twitter part-of-speech tagging for all: overcoming sparse and noisy data, in: *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pp. 198–206, 2013.

[18] L. Dey and S. M. Haque, Opinion mining from noisy text data, *Int. J. Doc. Anal. Recognit.* **12** (2009), 205–226.

[19] V. Dhanalakshmi, M. Anand Kumar, M. S. Vijaya, R. Loganathan, K. P. Soman and S. Rajendran, Tamil Part-of-Speech tagger based on SVMTool, in: *Proceedings of the COLIPS International Conference on Natural Language Processing (IALP)*, 2008.

[20] B. Eric, A simple rule-based part of speech tagger, in: *Proceedings of the Third Conference on Applied Natural Language Processing*, pp. 152–155, 1992.

[21] Y. Gal and Z. Ghahramani, A theoretically grounded application of dropout in recurrent neural networks, in *Advances in Neural Information Processing Systems*, pp. 1019–1027, 2016.

[22] R. Garside and N. Smith, A hybrid grammatical tagger: CLAWS4, in: *Corpus Annotation: Linguistic Information from Computer Text Corpora*, pp. 102–121, 1997.

[23] D. Gillick, C. Brunk, O. Vinyals and A. Subramanya, Multilingual language processing from bytes, arXiv:1512.00103, 2015.

[24] J. Gimenez and L. Marquez, *SVMTool: a general POS tagger generator based on support vector machines*, in: *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pp. 43–46, 2004.

[25] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan and N. A. Smith, Part-of-speech tagging for twitter: annotation, features, and experiments, in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, vol. 2, pp. 42–47, 2011.

[26] X. Glorot, A. Bordes and Y. Bengio, Domain adaptation for large-scale sentiment classification: a deep learning approach, in: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 513–520, 2011.

[27] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.* **9** (1997), 1735–1780.

[28] Hyperas, https://github.com/maxpumperla/hyperas, Accessed 3 February, 2017.

[29] M. Jagadeesh, M. A. Kumar and K. P. Soman, Deep belief network based part-of-speech tagger for Telugu language, in: *Proceedings of the Second International Conference on Computer and Communication Technologies*, pp. 75–84, 2016.

[30] A. Jamatia and A. Das, Part-of-speech tagging system for Indian social media text on Twitter, in: *Social-India 2014, First Workshop on Language Technologies for Indian Social Media Text, at the Eleventh International Conference on Natural Language Processing (ICON-2014)*, pp. 21–28, 2014.

[31] A. Jamatia and A. Das, Task report: Tool contest on POS tagging for codemixed Indian social media (Facebook, Twitter, and Whatsapp) Text@ icon 2016, in: *Proceedings of ICON*, 2016.

[32] A. Jamatia, B. Gamb and A. Das, Part-of-speech tagging for code-mixed English-Hindi Twitter and Facebook chat messages, in: *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pp. 239–248, 2015.

[33] G. Joshua, A bit of progress in language modeling, *Comput. Speech Lang.* **15** (2001), 403–434.

[34] Keras, https://keras.io/, Accessed 3 February, 2017.

[35] Y. Kim, Y. Jernite, D. Sontag and A. M. Rush, arXiv:1508.06615, 2015.

[36] K. S. G. Krishnan, A. Pooja, M. A. Kumar and K. P. Soman, Character based bidirectional LSTM for disambiguating tamil part-of-speech categories, *Int. J. Control Theory Appl.* (2017), 229–235.

[37] D. Kumar and G. S. Josan, Part of speech taggers for morphologically rich Indian languages: a survey, *Int. J. Comput. Appl.* **6** (2010), 32–41.

[38] S. S. Kumar, M. A. Kumar and K. P. Soman, Experimental analysis of Malayalam POS tagger using Epic framework in Scala, *ARPN J. Eng. Appl. Sci.* **11** (2016).

[39] LDCIL Tagset, www.ldcil.org/Download/Tagset/LDCIL/, Accessed 3 April, 2017.

[40] LDCIL, http://www.ldcil.org/standardsTextPOS.aspx, Accessed 5 February, 2017.

[41] J. Leskovec, Social media analytics: tracking, modeling and predicting the flow of information through networks, in: *Proceedings of the 20th International Conference Companion on World Wide Web*, pp. 277–278, 2011.

[42] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black and I. Trancoso, Finding function in form: compositional character models for open vocabulary word representation, arXiv:1508.02096, 2015.

[43] L. Marquez and L. Padro, A flexible POS tagger using an automatically acquired language model, in: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pp. 238–245, 1997.

[44] T. Mikolov, M. Karafiat, L. Burget, J. H. Cernock and S. Khudanpur, Recurrent neural network based language model, in: *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[45] N. Mishra and A. Mishra, Part of speech tagging for Hindi corpus, in: *International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 554–558, 2011.

[46] F. Nooralahzadeh, C. Brun and C. Roux, Part of speech tagging for French social media data, in: *Proceedings of the 25th International Conference on Computational Linguistics (COLING-2014)*, pp. 1764–1772, 2014.

[47] O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider and N. A. Smith, Improved part-of-speech tagging for online conversational text with word clusters, in: *The Association for Computational Linguistics in the Proceedings of Human Language Technologies*, pp. 380–390, 2013.
[48] R. N. Patel, P. B. Pimpale and M. Sasikumar, Recurrent neural network based part-of-speech tagger for code-mixed social media text, arXiv:1611.04989, 2016.
[49] S. Petrov, D. Das and R. McDonald, A universal part-of-speech tagset, arXiv preprint arXiv:1104.2086, Computer Language, 2011.
[50] G. Petz, M. Karpowicz, H. Fürschu, A. Auinger, V. Střítesky and A. Holzinger, Reprint of: Computational approaches for mining user's opinions on the Web 2.0, *Inform. Process. Manage.* **51** (2014), 510–519.
[51] P. B. Pimpale and R. N. Patel, Experiments with POS tagging code-mixed Indian social media text, arXiv:1610.09799, 2015.
[52] B. Plank, A. Søgaard and Y. Goldberg, Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss, arXiv:1604.05529, 2016.
[53] L. Qin, POS tagging of Chinese Buddhist texts using Recurrent Neural Networks, report, 2015.
[54] A. Ratnaparkhi, A maximum entropy model for part-of-speech tagging, in: *Conference on Empirical Methods in Natural Language Processing*, 1996.
[55] A. Ritter, C. Cherry and B. Dolan, Unsupervised modeling of twitter conversations, in: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 172–180, 2010.
[56] M. Rowe, M. Stankovic, A. Dadzie, B. Nunes and A. Cano, Making sense of microposts (#msm2013): big things come in small packages, in: *Proceedings of the WWW Conference-Workshops*, pp. 17, 2013.
[57] C. D. Santos and B. Zadrozny, Learning character-level representations for part-of-speech tagging, in: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1818–1826, 2014.
[58] K. Sarkar, A CRF based POS tagger for code-mixed Indian social media text, arXiv preprint arXiv:1612.07956, 2016.
[59] 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)*, pp. 205–212, 2013.
[60] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks, *IEEE Trans. Signal Process.* **45** (1997), 2673–2681.
[61] M. Shrivastava and P. Bhattacharyya, Hindi POS tagger using naive stemming: harnessing morphological information without extensive linguistic knowledge, in: *International Conference on NLP (ICON08)*, 2008.
[62] S. Singh, K. Gupta, M. Shrivastava and P. Bhattacharyya, Morphological richness offsets resource demand-experiences in constructing a POS tagger for Hindi, in: *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, pp. 779–786, 2006.
[63] Theano, http://deeplearning.net/software/theano/, Accessed 3 February, 2017.
[64] Twitter, https://blog.twitter.com/2012/twitter-turns-six, Accessed 19 November, 2016.
[65] Y. Vyas, S. Gella, J. Sharma, K. Bali and M. Choudhury, POS tagging of English-Hindi code-mixed social media content, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 974–979, 2014.
[66] WikiMalayalam, https://en.wikipedia.org/wiki/Malayalam, Accessed 5 May, 2017.
[67] M. M. Yoonus and S. Sinha, A hybrid POS tagger for Indian languages, *Lang. India* **11** (2011), 317–330.