

# Neural machine translation for Sanskrit to Malayalam using morphology and evolutionary word sense disambiguation

Rahul Chingamtattil, Rajamma Gopikakumari

Division of Electronics Engineering, School of Engineering, Cochin University of Science and Technology, Kochi, India

## Article Info

### Article history:

Received Mar 25, 2022

Revised Sep 3, 2022

Accepted Sep 14, 2022

### Keywords:

Morph analyzer

Neural machine translation

POS tagger

Recurrent neural network

Word sense disambiguation

## ABSTRACT

Neural machine translation (NMT) is a fast-evolving MT paradigm and showed good results, particularly in large training data circumstances, for several language pairs. In this paper, we have utilized Sanskrit to Malayalam language pair neural machines translation. The attention-based mechanism for the development of the machine translation system was particularly exploited. Word sense disambiguation (WSD) is a phenomenon for disambiguating the text to let the machine infer the proper definition of the particular word. Sequential deep learning approaches such as a recurrent neural network (RNN), a gated recurrent unit (GRU), a long short term memory (LSTM), and a bi-directional LSTM (BLSTM) were used to analyze the tagged data. By adding morphological elements and evolutionary word sense disambiguation, the suggested common character-word embedding-based NMT model gives a BLEU score of 38.58 which was higher than the others.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Rahul Chingamtattil

Division of Electronics Engineering, School of Engineering, Cochin University of Science and Technology

Kochi, India

Email: rahulshreys@gmail.com

## 1. INTRODUCTION

Machine translation (MT) is an automatic translation, in which the text is translated from one natural language using computer software [1], [2]. The translation is a challenge for people and not difficult for computers, as it addresses natural languages. High-quality translation involves a detailed concern of the source text and it enables effective and excellent language skills. Neural machine translation (NMT) is a new machine translation methodology [3]-[5] that helps to gain significantly in the evaluation of humans compared to statistical machine translation (SMT) systems. It's a whole end-to-end system designing one (or more) layers of machine translation into one huge neural network. NMT is a popular machine translation technology, as well as an effective strategy for several associated natural language processing (NLP) tasks, including dialogue, synthesis, and parsing [6].

Sanskrit is a language of the world's greatest study. It was evaluated both in India and overseas in many grammar schools [7], [8]. The number of grammatical categories in Sanskrit is traditionally varied between one and five. To establish the summarized, and to some extent the somaticized role of, the language depends extensively on morphologic characters. Sanskrit texts were made easily available in the public domain through digitalizing efforts. But there are still limited accessibilities for such digitized manuscripts. Due to the linguistic characteristics of the language, numerous technical issues occur while indexing and recovering such resources within a digital repository. In the context of the effective processing of Sanskrit text, Word division in Sanskrit is an important yet nontrivial requirement [9], [10].

The language used by the Kerala people in India is Malayalam [11]. Between Sanskrit and Malayalam, there are many similarities. Malayalam, which was brought into Kerala by Brahmins, is altered by Prakrit and Sanskrit. A unique blend of Sanskrit and Kerala's native languages, known as "manipravalam" [12], served as a literary expression in the eleventh century. The Sanskrit language absorbed Malayalam not only at the lexical level, but also at the morphemic level, phonemic level, and grammatical levels of language. The development of the morphological analyzer of Malayalam is initially followed rule-based approaches. But recent developments in machine training have altered the paradigm into the development of morphological analyzer computational models [13].

The reduction of the meanings of a word occurs when the context in which the term is employed is considered [14], [15]. The use of morphological tests, parts of speech [16] and syntactic parsing relates linguistically to word sense disambiguation. A challenge is the disambiguation of word senses by the machine. A wide range of language processing systems is pre-processing using prosedur operasional standar (POS) tagging [17], which greatly increases their accuracy and recall. Applications to speech recognition and analysis, data gathering, and some other NLP tasks are mostly found in the POS tagged (annotated) language corpora. Including machine translation, natural language processing requires the disambiguation of word sense [18]. The linguist differentiates linguistic ambiguity from structural. If a word's senses give rise to ambiguity, it is called lexical ambiguity. When a sentence or phrase is interpreted in distinct syntactic terms as ambiguity, structural ambiguity is named. The emphasis here is on linguistic ambiguity. Looking to resolve ambiguity with a deep learning method in Malayalam [19]. Deep learning approaches improve accuracy and effective performance for various tasks in which the purposes of the neural network itself need to be developed [20]. The main contributions of this paper can be summarized as follows:

- Pre-processing stage is used for sentence alignment, followed by a deep morph analyzer module that performs the morphological analysis of Sanskrit and Malayalam sentences.
- Deep learning-based POS tagger is developed for both Sanskrit and Malayalam languages.
- After word sense disambiguation module resolves the ambiguity in translated output sentences and is given to the corresponding deep learning-based morph generator to combine morphologically segmented outputs.

The rest of the paper is organized as follows: The rest of the paper organized as follows; Section 2 proposed NMT system for the translation between Sanskrit and Malayalam languages is explained. The experimental results and discussion in section 3 and the conclusion part is presented in section 4. Table 1 show the review of the existing works.

**Table 1. Machine translation system based on their language pair with its corresponding BLEU and accuracy**

Author	Year	Technique	Language	Parameters	
				BLEU	Accuracy
Khan and Usman [21]	2019	Artificial Neural Network	English to Urdu and Hindi	NA	81.70
Kavirajan <i>et al.</i> [22]	2017	Rule-Based Machine Translation System	English to Tamil	NA	71.80
Sreedeepta and Idicula [23]	2017	Interlingua based machine translation	Sanskrit to English	NA	NA
Muskaan <i>et al.</i> [24]	2019	Hybrid MTS	Sanskrit to Hindi	13.445	NA
Koul and Manvi [25]	2021	Recurrent neural networks	Sanskrit to English	NA	85.33

## 2. METHOD

This work aims to generate a bidirectional translation between Sanskrit and Malayalam languages. So, an automatic language detection model is developed for analysing the input sentences. NMT performs the translation from Sanskrit to Malayalam or Malayalam to Sanskrit according to the output from the language identification module. A pre-processing stage is used for sentence alignment, followed by a deep morph analyzer module that performs the morphological analysis of Sanskrit and Malayalam sentences. The output of the morph analyzer is given to an NMT module after joint character-word embedding. The output of the NMT is given as input of the POS tagger module to perform tagging. A deep learning-based POS tagger is developed for both Sanskrit and Malayalam languages. A word sense disambiguation module resolves the ambiguity in translated output sentences and is given to the corresponding deep learning-based morph generator to combine morphologically segmented outputs. Figure 1 shows the proposed architecture.

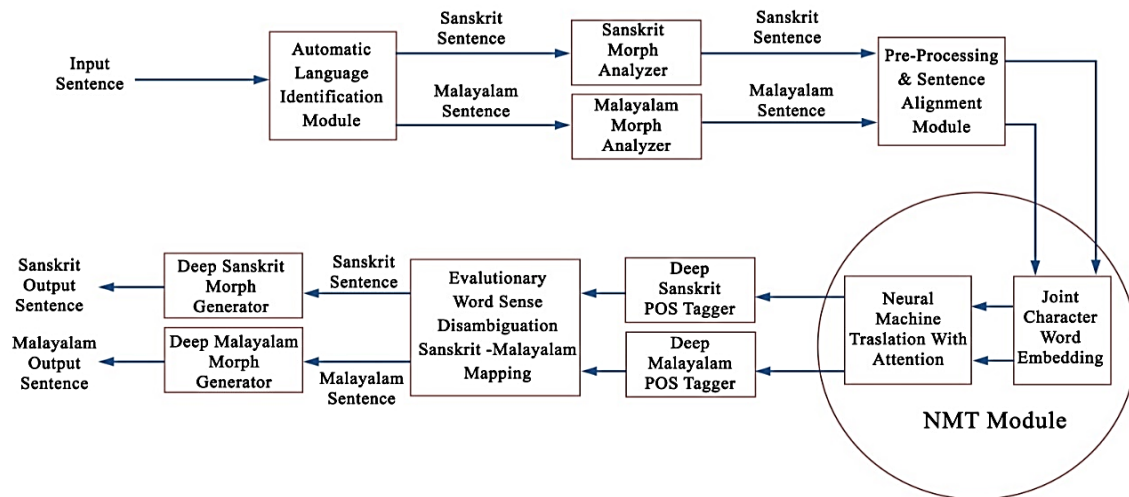


Figure 1. Proposed architecture

**2.1. Automatic language identification module**

The Sanskrit and Malayalam languages are identified using an automatic language detection model. For building a language classifier, a deep neural network technique is used. The input character should be automatically identified by this model as Sanskrit or Malayalam. For this goal, the possibility is explored for the use of recurrent neural networks, long short term memory (LSTM) networks, and gated recurrent units (GRU). These models' input sequences are characters for a certain language. Languages like Sanskrit and Malayalam are comparable and they contain many common words. A tokenizer for dividing the body into distinct characters is therefore implemented.

**2.2. Morphological analyzer**

The morphological analysis consists of the segmentation of a word into its component morphemes and typically the grammatical categories of information, as well as the assignment of lexical information to a particular lexeme or lemma. Morphological analysis is the identification of the word portions or, more formally, the word components. The design and deployment of the morphology analyzer and generator appear promising for NLP applications in Malayalam.

**2.2.1. Sanskrit morph analyzer**

The morphology analyzer is used with four parallel deep learning models for Subanta, Tinanta, Taddhita, and Samasa. The corresponding words are trained in each model. Inputs of all models (Subanta, Tinanta, Taddithas, and Samasa) are labeled as Romanized characters and outputs as morpheme separated words. The Word2Vec technique was used to embed characters. Its outputs are routed into a bidirectional recurrent layer with LSTM units in Figure 2. The built-in size is 128 and the hidden layer is 32. The model is equipped with a category cross-entropy loss, greedy decoding, and Adam optimizer of 2. Logistic regression is carried out for the final label identification. Morph analyzer is for all types and is explained in algorithm 1.

**2.2.2. Malayalam morph analyzer**

It is a demanding technique to identify all morphological changes in Malayalam. The main obstacles to creating a Malayalam morphological analyzer are influences, many suffixes, and word compounding. Malayalam grammatical categories are noun, pronoun, verb, and adverb. The big issues in the development of a Malayalam morph analyzer are the formulation of Orthographic (Sandhi) rules, the design of verb morphology, and the design of noun morphology. This study presents an architecture based on deep learning for the implementation of the morphological analyzer in Malayalam. The network formulates a problem for long short-term memory (LSTM) since it is regarded as the efficient architecture for sequential labeling problems at the character level.

**2.3. Neural machine translation**

NMT is a deep learning method for automatically translating from one language to another. It is an extension of models used for neural language sequence by sequence technique [26]. The NMT model contains an encoder and a decoder that is trained together. These can be implemented with recurrent neural network (RNN), LSTM, or GRU. The NMT attention-based architecture has an encoder-decoder design: the

full source sentence,  $X = [x_1, x_2, \dots, x_{N_x}]$  is encoded in a neural network-based decoder is used to decode a sequence of vectors, which generates the target translation  $y = [y_1, y_2, \dots, y_{N_y}]$   $Y = y_1, y_2, \dots, y_{N_y}$ .

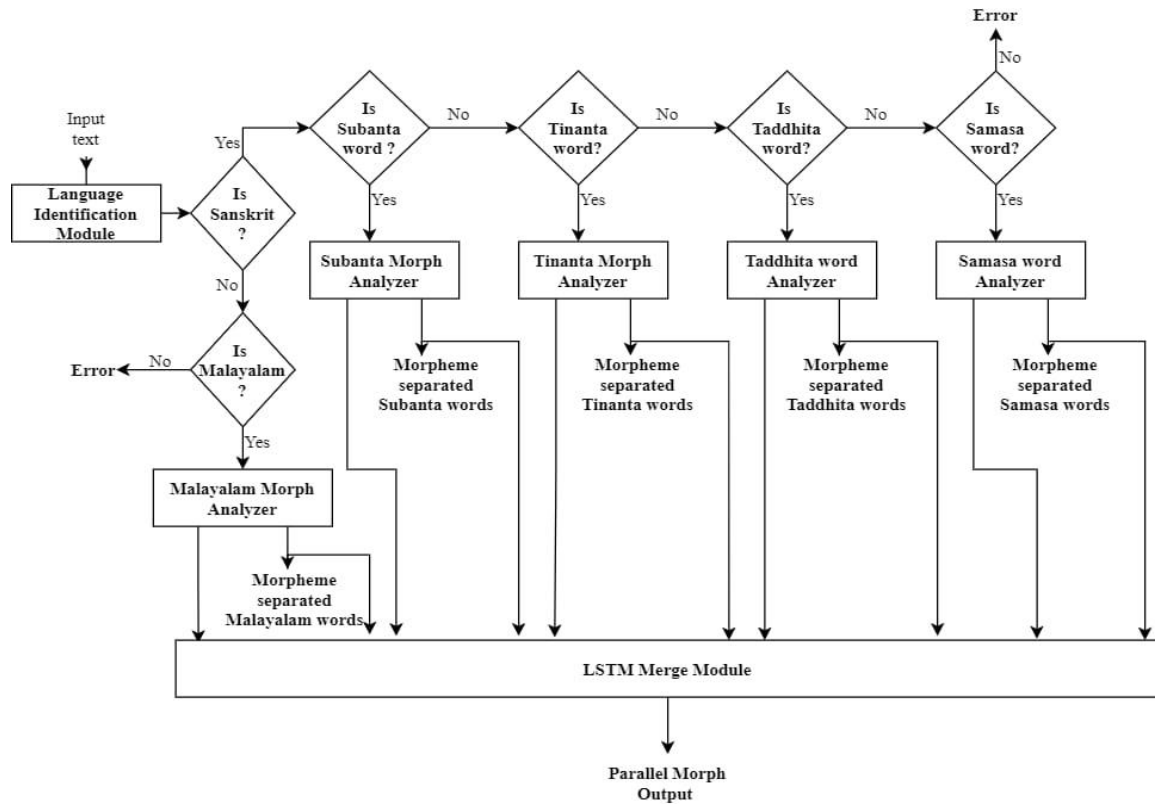


Figure 2. Flowchart of the designed framework

### Algorithm 1: Morph Analyzer

**Input:** {Romanized, C-V corresponding labels}

**Output:** { Morpheme separated words }

**Begin**

**Step 1:** Find the Bow depiction of the character.

**Step 2:** Set the embedding size to 128 and the concealed size to 32.

**Step 3:** Describe the network's first layer as a fully connected layer.

**Step 4:** Describe an LSTM/RNN/GRU network with hidden size 32.

**Step 5:** Select Softmax as the activation function.

**Step 6:** Loss function = categorical cross entropy  
Optimizer = Adam , Batch size = 3

**Step 7:** Using logistic regression, find the label of each syllable.

**End**

- a) Encoder: The encoder is usually a bidirectional RNN that encodes data in both directions of the source sentence using forward and reverse RNNs: the forward one uses the source text as input and computes a forward hidden state  $\vec{h} = [\vec{h}_1, \vec{h}_2, \dots, \vec{h}_{N_x}]$  from left to right. The reverse one calculates backward hidden states  $\overleftarrow{h} = [\overleftarrow{h}_1, \overleftarrow{h}_2, \dots, \overleftarrow{h}_{N_x}]$  using the source text  $x$  as input from right to left. Then, to generate annotations of the source sentence  $h = [h_1, h_2, \dots, h_{N_x}]$  forward hidden states, and backward hidden states are combined.

$$h_i = [\vec{h}_i, \overleftarrow{h}_i] \quad (1)$$

- b) Decoder: The decoder is a conditional recurrent language model that breaks down the target sentence into its parts:

$$\log p(Y|x) = \sum_{s=1}^{N_y} \log p(y_s | Y_{<s}, x) \quad (2)$$

Following the generation of the partial sequence  $Y_{<y} = y_1, y_2, \dots, y_{s-1}$ , the decoder generates the word  $y_s$ . The current hidden state  $d_s$ , the previously created target word  $y_{s-1}$ , and the current context vector  $c_s$  are used to determine the likelihood of  $y_s$ :

$$p(y_s | Y_{<y}, x) = \text{softmax}(f(d_s = g(d_{s-1}, y_{s-1}, C_s))) \quad (3)$$

where  $f(\cdot)$  are a non-linear activation function and the current decoding hidden state  $d_s$  is calculated using the former decoding hidden state  $d_{s-1}$ , the previously formed target word  $y_{s-1}$ , and the current context vector  $c_s$  as (4).

$$d_s = g(d_{s-1}, y_{s-1}, C_s) \quad (4)$$

Where  $g(\cdot)$  as a recurrent activation function, such as GRU or LSTM.  $c_s$  is a context representation that changes over time, which is determined as a weighted sum of source annotations:

$$C_s = \sum_{i=1}^{N_x} \alpha_{s,i} h_i \quad (5)$$

where the weight  $\alpha_{s,i}$  of the source annotation  $h_i$  is calculated as (6).

$$\alpha_{s,i} = a(d_{s-1}, h_i) \quad (6)$$

It is an attention model that determines how closely the target word  $y_s$  and the source word annotation  $h_i$  match.

### 2.3.1. Attention-getting neural machine translation

To focus on exact words in a source sentence, vectors for each word in the sentence are kept in the place of a single vector for the entire word in a sentence [6]. The number of vectors in a source sentence  $(h_{s_1}, \dots, h_{s_x})$  is based on several words. These words (or their related vectors) can be combined to produce a matrix as.

$$H_s = [h_{s1}; h_{s2}; \dots; h_{sx}] \quad (7)$$

Each column  $H_s$  resembles a word from the source sentence. The number of columns in the matrix will be determined by the number of words in the source sentence. To provide it, we create an attention vector  $(a_x)$ , which converts the matrix  $H_s$  to a vector.

### 2.3.2. Training

The training purpose of the NMT system is the same as that of a recurrent language model, with the exception that the target sentence is conditional on the source sentence. For neural machine translation training over the parallel corpus T, the loss function is (8).

$$L = \sum_{(x,y) \in T} -\log p\left(\frac{y}{x}\right) \quad (8)$$

The neural machine translation architecture's forward propagation is essentially identical to that of typical recurrent neural networks. Back propagation methods continue similarly through the encoder side, but with no prediction loss.

### 2.3.3. Testing

Test the NMT system for new source sentences after it has been trained. Greedy decoding and Beam search decoding are the two main techniques. At each time step, the greedy decoding method generates predictions. The beam-search algorithm employed in phrase-based statistical machine translation systems is more complicated than the NMT beam-search technique. The decoder selects the most likely word in each time stamp and uses it as an input to the following timestamp until the end-of-sentence marker, according to a greedy algorithm.

### 2.3.4. Character-level word embedding

Character embedding follows the same steps as CBOW's Word2Vec technique to word embedding. Feed both the word embedding and the character embedding of the current word into the CBOW's input. The word and character embedding matrices are both updated when the error is back propagated through the neural network. The following formula can be used to establish an approximate relationship between a word embedding and a co-occurrence matrix:

$$\log(X_{ij}) = w_i^T w_j + p_i + p_j \quad (9)$$

where  $\vec{w}_i$  and  $\vec{w}_j$  are the corresponding embedding of  $w_i$  and  $w_j$ , and  $p_i$  and  $p_j$  their offset parameters.

### 2.4. Part of speech tagging module

The construction of an automatic word sense disambiguation system can be aided by an efficient POS tagger. Because connections are perfectly related with root words to describe their syntactic features, POS is very straightforward to discern. The problem is supposed to be a character-level sequence labelling problem. Character embedding minimizes the vocabulary size and removes the appearance of out-of-vocabulary terms during the testing phase. The Sanskrit POS tagger is implemented using a bidirectional gated recurrent unit (GRU) to track the framework of backward and forward direction.

### 2.5. Evolutionary word sense disambiguation

The problem of word sense disambiguation (WSD) is regarded as AI-complete. Knowledge is a crucial component in word sense disambiguation. Data is provided by knowledge resources to relate words to their meanings. Supervised WSD and unsupervised WSD are the two basic techniques to WSD that can be identified. To determine the link between neighboring words, the Lesk relatedness metric is utilized.

### 2.6. Deep morphological generator

The goal of morphological generation is to create the inflected form of a word based on the Feature Structure's features and values. It is also important to utilize the language resources that were generated for the analysis function. The root word would be the input to the morphological generator, which would then inflect the word to the morphology of the respective language and output the target forms of the word. The morphological generator's general format is as follows:

$$\frac{\text{Stem}}{\text{root}} + \text{suffixes} \leftarrow \text{Word} \quad (10)$$

The algorithm 2 of the Sanskrit and Malayalam morph generator is explained as follows.

---

#### Algorithm 2: Deep Morphological Generator

---

**Step 1:** Obtain the word to be examined.  
**Step 2:** Examine the Root Dictionary to see if the entered word exists.  
**Step 3:** If the word appears in the dictionary, Then stop;  
**Else**  
 Remove any suffixes from the right side of the sentence.  
**End If**  
**Step 4:** If any suffix is present in the word, Then  
 Examine the availability of the suffix in the dictionary.  
 Then eliminate the suffix present,  
 Then, without the suffix, re-initialize the term,  
**Go to Step 2.**  
**Step 5:** Repeat until the Dictionary identifies the root/stem term.  
**Step 6:** Save the Sanskrit root/stem word in a variable, then look for the Malayalam equivalent in the bilingual dictionary.  
**Step 7:** Check the Sanskrit word for all grammatical features, and then generate the Malayalam term with the same features.  
**Step 8:** Exit.

---

## 3. RESULTS AND DISCUSSION

Each module of the proposed model has been subjected to extensive analysis. To construct the final model, the best combination acquired in each is used. The following system requirements are used to carry out the experiments. CPU platform: Intel, Machine type: (8 vCPUs, 52 GB memory), NVIDIA Tesla K80, 2 GPUs, Standard persistent disk: 1 TB, OS: Debian GNU/Linux 9.9(stretch) (GNU/Linux 4.90-9amd64\*8664).

### 3.1. Data preparation

Experiments are carried out employing Sanskrit and Malayalam parallel corpora. The digital corpus of Sanskrit (DCS) and Computational Linguistics R&D at JNU-India provide Sanskrit corpora. They are pre-processed into different sentences, each with a limit of 10 words. There are 2, 54, 700 Sanskrit sentences in the resulting dataset. Manually created Malayalam sentences are based on consultations with linguistic specialists in both Sanskrit and Malayalam. The proposed Malayalam morphological analyzer, has an accuracy of 98.25% and the data preparation is explained in Table 2.

Table 2. Number of words and characters in Malayalam

	Malayalam noun	Malayalam verb	Total
Number of words	1,04,643	82,247	1,86,890
Number of characters	15,14,198	13,11,205	28,25,403

### 3.2. Performance measures of language identification module

The determination of the optimal hyper parameters is critical to the effectiveness of a deep learning algorithm for language identification. The studies use various combinations of loss functions, learning rate, optimizers, batch sizes, and several epochs. The learning rate is chosen from the range of 0:001; 0:01; 0:1; 1; it is found that with a learning rate of 0.01 the method produces good accuracy.

The model is trained with a single hidden layer with several neurons ranging from 1 to 400 at first. The model is then trained using two hidden layers, each with a different number of neurons ranging from one to 400. The accuracy, as well as a sample set of outcomes and the total running time, are calculated for each case. The LSTM architecture, which consists of three hidden layers with 360, 240, and 128 neurons, has a maximum accuracy of 97.623%.

From Table 3, it can be assumed that the language identification model based on the LSTM network performs the best. When it comes to categorizing Sanskrit and Malayalam literature, it has a 97.623% accuracy rate. The Sanskrit texts are sent to the Sanskrit morph analyst, while the Malayalam texts are sent to the Malayalam morph analyzer.

Table 3. Performance measures of Sanskrit-Malayalam Language identification model

Architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
RNN	97.491	0.98	0.95	0.97
LSTM	97.623	0.99	0.94	0.96
GRU	97.585	0.98	0.95	0.97

### 3.3. Performance measures of language identification module

#### 3.3.1. Sanskrit morph analyzer

Both Sanskrit and Malayalam morphological analysis can be performed using the parallel deep learning architecture. The Sanskrit morph analyzer generates four alternative results for the words Subanta, Tinanta, Tadditha, and Samasa. The morphology of Malayalam words can be performed with a Malayalam morph analyzer. For training, a 128-embedding size is used, and deep learning architectures like RNN and LSTM are used. Table 4 show the performance measures and total running time for all of the models.

Table 4. Performance measures of morph analyzer

Model	Architecture	Accuracy (%)	Precision	Recall	Time(Hrs)
Subanta Morph Analyzer	RNN	95.81	94.2	94.3	624
Subanta Morph Analyzer	LSTM	95.96	94.4	94.5	627
Tinanta Morph Analyzer	RNN	96.82	94.6	94.6	652
Tinanta Morph Analyzer	LSTM	96.93	94.8	94.9	656
Tadditha Morph Analyzer	RNN	95.61	94.2	94.4	612
Tadditha Morph Analyzer	LSTM	95.83	94.3	94.4	618
Samasa Morph Analyzer	RNN	95.44	94.1	94	610
Samasa Morph Analyzer	LSTM	95.67	94.3	94.1	612

#### 3.3.2. Malayalam morph analyzer

The deep learning approach is used to train Malayalam words. The model's input is Romanized characters, while the output is morpheme separated words. A single hidden layer is used to train the model, with the number of neurons ranging from 1 to 250. The size of the character embedding is also variable,

ranging from 1 to 200. The architecture with 32 neurons in a single hidden layer and a 128-embedding size has the best accuracy of 98.25%. Figure 3 illustrates the graph for the same. The proposed Malayalam morphological analyzer has a 98.25% accuracy rate. The accuracy of the Parallel Sanskrit-Malayalam morphological analyzer model is 96.2%.

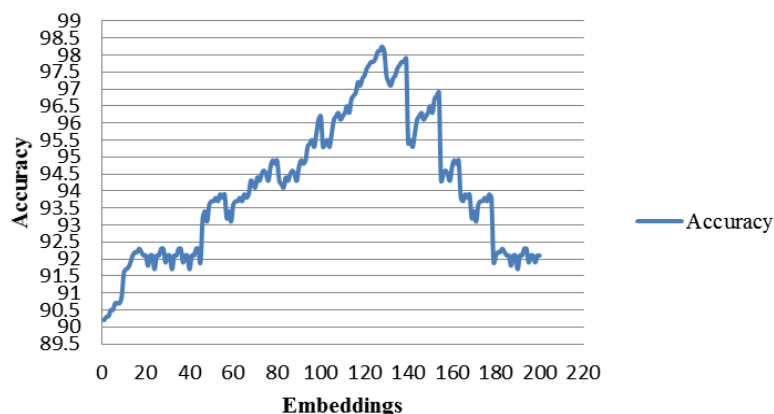


Figure 3. Accuracy of the Malayalam morph analyzer with 32 neurons in the hidden layer varies with the character embedding size

### 3.4. Performance measures of joint character-word embedding

The selection of the best hyper-parameters affects the performance of Joint Character-Word Embedding. The tests are carried out with various character and word embeddings, as well as varied combinations. The learning rate of the model is 0.0075, and the activation function is rectified linear unit (ReLU). To improve the training accuracy, a 0.1 dropout is utilized. Each case's Precision, Recall, and Accuracy is calculated, with a sample set of results provided in Table 5. With a character embedding size of 300 and a word embedding size of 300, the architectural character embedding size has a maximum accuracy of 97.4%.

Table 5. Performance measures of Joint Character-Word embedding model using Bidirectional GRU

Word-Embedding size	Character-Embedding size	Accuracy (%)	Precision	Recall
100	100	96.2	95.7	95.9
200	200	96.8	96.4	96.2
300	325	97.4	97.3	97.1

### 3.5. Performance of NMT

Several experiments based on Translation quality, Training Steps, Batch Size, Training Epoch, Computation Speed, and Training Throughput terminology, as well as a summary of the findings, are described in this section. The Transformer encoder-decoder design is used in all of the trials. The Transformer's default parameters are listed in Table 6.

Table 6. Parameters of transformer

Parameters	Value
Batch size	4096
Optimizer	Adam
Learning rate	2
Encoder, Decoder type	Transformer
Heads	8
RNN size	512
Word-Vector size	512
Attention dropout	0.1

#### 3.5.1. Computation speed and training throughput

Computational Speed and Training Throughput based on the number of GPUs, the batch size, and the model size show in Table 7. The default Transformer model has a batch size of 4096. It has been discovered that when batch sizes grow larger, calculation speed reduces while training throughput grows linearly.



Table 7. Computation speed and training throughput of the system

Batch Size	Computation Speed (Steps/hour)	Training Throughput (Sub words/Hour)
1000	28.6K	29.4M
1500	21.5K	33.7M
2000	17.4K	33.8M
2500	13.2K	35.4M
3000	11.3K	35.7M
3500	10.7K	36.2M
4096	8.5K	36.4M

### 3.5.2. Training data

There are 2, 54,700 Sanskrit-Malayalam sentence pairs in the training data. The proposed model's BLEU learning curve is shown in Figure 4. After 624 hours of training on the dataset, the BLEU score converges to 33.38.

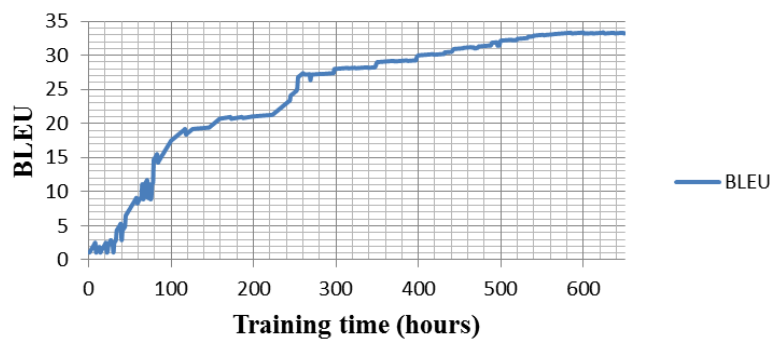


Figure 4. BLEU score on training data

### 3.6. Performance measures of POS tagger

Joint character-word embedding was used to perform the evaluations. The word2vec algorithm considers each word as well as its immediate left and right words to create word vectors. The character-level sequence is represented by CNN. In the proposed POS tagger module, character embedding is combined with Word2Vec using a convolution technique. RNN, LSTM, and Bidirectional GRU are used to implement the model. The loss function is computed using “Cross-entropy”, and the number of epochs is set to 100. The model's learning rate is 0.01 and the activation function is rectified linear unit (ReLU). To improve training accuracy 0.1 dropout is used. Following that, the model is trained using two hidden layers, each with a different number of neurons ranging from 1 to 32. The size of the word-character embedding is likewise modified, ranging from 50 to 350. In each scenario, Precision, Recall, and Accuracy are determined, and a sample set of findings is presented in Tables 8-10.

Table 8. Performance measures of Malayalam POS tagger using RNN

Number of neurons		Word-Embedding size	Character-Embedding size	Accuracy (%)	Precision	Recall
Hidden Layer1	Hidden Layer2					
16		100	100	89.3	89.1	88.6
16		200	200	89.1	89.3	88.8
16		300	300	88.6	89.8	89.2
32		100	100	90.4	90.1	89.5
32		200	200	90.2	90.3	89.7
32		300	300	89.8	90.1	90.4
10	10	100	100	92.3	92.4	91.5
10	10	200	200	92.6	92.8	92.1
10	10	300	300	93.1	93.5	92.4
16	16	100	100	93.7	92.8	93.1
16	16	200	200	93.5	93.2	93.7
16	32	290	325	95.3	95.6	94.8
32	32	100	100	94.2	93.8	94.3

Table 9. Performance measures of Malayalam POS tagger using LSTM

Number of neurons		Word-Embedding size	Character-Embedding size	Accuracy (%)	Precision	Recall
Hidden Layer1	Hidden Layer2					
16		100	100	89.9	89.7	88.6
16		200	200	88.6	89.1	88.9
16		300	300	89.7	89.6	89.9
32		100	100	91.6	90.9	89.4
32		200	200	90.2	91.2	90.7
32		300	300	90.5	90.4	90.1
10	10	100	100	92.8	92.1	91.8
10	10	200	200	92.7	92.5	92.3
10	10	300	300	93.6	93.3	93.4
16	16	100	100	93.9	93.2	93.5
16	16	200	200	94.3	94.1	94.2
16	32	290	325	95.9	95.4	95.2
32	32	100	100	94.8	94.1	94.4

Table 10. Performance measures of Malayalam POS tagger using Bidirectional GRU

Number of neurons		Word-Embedding size	Character-Embedding size	Accuracy (%)	Precision	Recall
Hidden Layer1	Hidden Layer2					
16		100	100	88.7	88.6	88.2
16		200	200	89.2	89.4	89.1
16		300	300	90.7	90.6	90.5
32		100	100	91.8	91.4	91.2
32		200	200	91.9	91.3	91.1
32		300	300	92.4	92.4	92.1
10	10	100	100	93.1	92.9	92.8
10	10	200	200	93.4	93.4	93.3
10	10	300	300	94.7	94.3	94.2
16	16	100	100	94.9	94.7	94.5
16	16	200	200	95.1	94.9	94.7
16	32	290	325	96.4	96.3	96.2
32	32	100	100	95.7	95.5	95.4

#### 4. CONCLUSION

By combining character-word embedding, morphology, and evolutionary word sense disambiguation, the proposed Sanskrit to Malayalam translation system is a modified version of Neural Machine Translation. The proposed system received a 38.58 BLEU score and a 3.84 fluency score. The proposed Malayalam morphological analyzer has a 98.25% accuracy rate, whereas the Deep POS tagger has a 98.32% accuracy rate. In comparison with earlier studies, the results show an increasing inefficiency. As a result, research in these areas can help to develop more effective translation systems. Other NLP applications could be built using the proposed deep POS tagger for Sanskrit, Deep Malayalam morph analyzer, and Sanskrit word sense disambiguation algorithm.




#### REFERENCES

- [1] S. K. Mahata, D. Das, and S. Bandyopadhyay, "Machine translation using recurrent neural network on statistical machine translation," *Journal of Intelligent Systems*, vol. 28, no. 3, pp. 447-453, 2019, doi: 10.1515/jisys-2018-0016.
- [2] R. Kumar, P. Jha, and V. Sahula, "An augmented translation technique for low resource language pair: Sanskrit to hindi translation," *In Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, pp. 377-383, Dec 2019, doi: 10.1145/3377713.3377774.
- [3] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," *arXiv preprint arXiv:1701.02810*, 2017, doi: 10.48550/arXiv.1701.02810.
- [4] M. Singh, R. Kumar, and I. Chana, "Machine translation systems for Indian languages: review of modelling techniques, challenges, open issues and future research directions," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2165-2193, 2021, doi: 10.1007/s11831-020-09449-7.
- [5] S. R. Laskar, A. F. U. R. Khilji, D. Kaushik, P. Pakray, and S. Bandyopadhyay, "Improved English to Hindi multimodal neural machine," doi: 10.48550/arXiv.2106.00250.
- [6] S. Singh, A. M. Kumar, and K. P. Soman, "Attention based English to Punjabi neural machine translation," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 551-1559, 2018, doi: 10.3233/JIFS-169450.
- [7] A. Gupta, A. Krishna, P. Goyal, and O. Hellwig, "Evaluating neural morphological taggers for sanskrit," *arXiv preprint arXiv:2005.10893*, 2020, doi: 10.48550/arXiv.2005.10893.
- [8] Sitender and S. Bawa, "Sansunl: a Sanskrit to UNL enconverter system," *IETE Journal of Research*, vol. 67, no. 1, pp. 117-128, 2021, doi: 10.1080/03772063.2018.1528187.
- [9] V. Reddy, A. Krishna, V. D. Sharma, P. Gupta, and P. Goyal, "Building a word segmenter for sanskrit overnight," *arXiv preprint arXiv:1802.06185*, 2018, doi: 10.48550/arXiv.1802.06185.




- [10] S. Bawa, "Sanskrit to universal networking language EnConverter system based on deep learning and context-free grammar," *Multimedia Systems*, pp.1-17, 2020, doi: 10.1007/s00530-020-00692-3.
- [11] S. N. Mohan Raj, S. S. Kumar, S. Rajendran, and K. P. Soman, "Word sense disambiguation of Malayalam nouns," *In Recent Advances in Computational Intelligence*, pp. 291-314, 2019, doi: 10.1007/978-3-030-12500-4\_22.
- [12] M. Muhaseen, A. Kumar, B. Vinuraj, and R. P. Joseph, "An Archaeological Investigation into Shukasandesham," *Unmuneelilandesham and Mediaeval Malayalam Literary Works*.
- [13] B. Premjith, K. P. Soman, and M. A. Kumar, "A deep learning approach for Malayalam morphological analysis at character level," *Procedia Computer Science*, vol. 132, pp. 47-54, 2018, doi: 10.1016/j.procs.2018.05.058.
- [14] M. Abid, A. Habib, J. Ashraf, and A. Shahid, "Urdu word sense disambiguation using machine learning approach," *Cluster Computing*, vol. 21, no. 1, pp. 515-522, 2018, doi: 10.1007/s10586-017-0918-0.
- [15] S. Wang, W. Zhou, and C. Jiang, "A survey of word embeddings based on deep learning," *Computing*, vol. 102, no. 3, pp. 717-740, 2020, doi: 10.1007/s00607-019-00768-7.
- [16] X. Feng, Z. Feng, W. Zhao, B. Qin, and T. Liu, "Enhanced neural machine translation by joint decoding with word and POS-tagging sequences," *Mobile Networks and Applications*, vol. 25, no. 5, pp.1722-1728, 2020, doi: 10.1007/s11036-020-01582-8.
- [17] S. Kumar, M. A. 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)," *Journal of Intelligent Systems*, vol. 28, no. 3, pp. 423-435, 2019, doi: 10.1515/jisys-2017-0520.
- [18] W. Al-Saiagh, S. Tiun, A. Al-Saffar, S. Awang, and A. S. Al-Khaleefa, "Word sense disambiguation using hybrid swarm intelligence approach," *PloS one*, vol. 13, no. 12, p. e0208695, 2018, doi: 10.1371/journal.pone.0208695.
- [19] H. T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification?," In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [20] K. K. Akhil, R. Rajimol, and V. S. Anoop, "Parts-of-Speech tagging for Malayalam using deep learning techniques," *International Journal of Information Technology*, vol. 12, no. 3, pp. 741-748, 2020, doi: 10.1007/s41870-020-00491-z.
- [21] S. N. Khan, and I. Usman, "A model for english to urdu and hindi machine translation system using translation rules and artificial neural network," *Int. Arab J. Inf. Technol.*, vol. 16, no. 1, pp. 125-131, 2019.
- [22] B. Kavirajan, M. A. Kumar, K. P. Soman, S. Rajendran, and S. Vaithehi, "Improving the rule based machine translation system using sentence simplification (English to Tamil)," In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 957-963, 2017, doi: 10.1109/ICACCI.2017.8125965.
- [23] H. S. Sreedeepta, and S. M. Idicula, "Interlingua based Sanskrit-English machine translation," In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 2017, pp. 1-5, doi: 10.1109/CISCT46613.2019.9008136.
- [24] M. Singh, R. Kumar, and I. Chana, "Neuro-FGA based machine translation system for Sanskrit to Hindi language," In *2019 International Conference on Innovative Sustainable Computational Technologies (CISCT)*, pp. 1-6. IEEE, 2019, doi: 10.1109/CISCT46613.2019.9008136.
- [25] N. Koul, and S. S. Manvi, "A proposed model for neural machine translation of Sanskrit into English," *International Journal of Information Technology*, vol. 13, no. 1, pp. 375-381, 2021, doi: 10.1007/s41870-019-00340-8.
- [26] X. Wang, Z. Tu, and M. Zhang, "Incorporating statistical machine translation word knowledge into neural machine translation," *IEEE/ACM Transaction on Audio, Speech, and Language Processing*, vol. 26, no. 12, pp. 2255-2266, 2018, doi: 10.1109/TASLP.2018.2860287.

## BIOGRAPHIES OF AUTHORS



**Rahul Chingamtotattil**    received the bachelor's degree in computer engineering from Mahatma Gandhi University, Kerala in 2004, the master's degree in Computational Engineering and Networking from Amrita University in 2009 and pursuing doctorate degree in Electronics Engineering from Cochin University. He is currently working as an Assistant Professor at the Department of Computer Engineering, LBS College Of Engineering Kasaragod (Govt of Kerala Undertaking). His research areas include mobile security, deep learning, and social network analysis. He can be contacted at email: rahulshreys@gmail.com.



**Dr. Rajamma Gopikakumari**    received a bachelor's degree in Electronics and Communication Engineering from Cochin University, Kerala in 1982, a master's degree in Electronics and Communication Engineering from Cochin University in 1985, and a doctorate degree in Electronics Engineering from Cochin University in 1988. She is currently working as a Professor at the Department of Electronics and Communication Engineering at Cochin University, Kerala. Her research areas include signal processing, deep learning, and social network analysis. She has been serving as a reviewer for many highly-respected journals. She can be contacted at email: gopiikakumarii@gmail.com.