International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Translating Sanskrit to Hindi Language using Recurrent Neural Network (RNN)-L2 Regularization

Prashanth Kammar*[1], Parashuram Baraki[2], Sunil Kumar Ganganayaka[3], Manjunath Swamy Byranahalli Eraiah[4], Kolakaluri Lakshman Arun Kumar[5]

Submitted: 29/01/2024 **Revised**: 07/03/2024 **Accepted**: 15/03/2024

Abstract: Machine Translation (MT) is a subfield of computer linguistics that focuses on the automatic translation from one natural language into another without any human involvement. There is a huge need for translating information between languages to send and communicate thoughts because native people interact in a variety of languages. However, Sanskrit is an ancient Indo-European language that requires essential processing to be explored in computer science and computational language analysis. In this paper, Recurrent Neural Network (RNN)-L2 regularization method is proposed for the Sanskrit to Hindi translation language. A neural machine translation system is trained using the linguistic information from the rule-based input. The proposed method is innovative and adaptable to any low-resource language with extensive morphology that covers multiple domains with minimal human involvement. The efficacy of the RNN-L2 regularization method is demonstrated by employing the dataset of Corpora. The existing methods such as machine translation systems, and hybrid machine translation systems are used to explain the efficacy of the RNN-L2 regularization method. The proposed RNN-L2 regularization method achieves better BLEU score, and METEOR of 76% and 72% compared with the existing methods such as machine translation systems, and hybrid machine translation systems.

Keywords: *Linguistic Feature Extraction, Machine Translation, Natural Language, Recurrent Neural Network, Rule-based system*

1. Introduction

Machine Translation (MT) systems have become enhanced through the use of an ongoing methodology known as Neural Machine Translation (NMT). Many online translation services, including Bing Systran, Google Translate, and e-Translation, use Neural Machine Translation (NMT) [1]. The NMT invention is a method that employs deep learning neural networks to map among several natural languages [2]. Dictionary-matching methods were initially employed for managing machine translation tasks and eventually upgraded to rule-based methods [3]. Systems for machine translation are particularly helpful since they greatly simplify human-to-human communications. Today, people use them for a variety of purposes including professional settings, traveling, reading books, and articles published in various languages. The accessibility, quickness, affordability, and user-friendliness of machine translation systems are major benefits [4]. The corpora dataset includes 1012 multilingual sentence pairs from mobile translation services, including daily chats, clear sentences, and travel sentences [5]. Rule-Based Machine Translation (RBMT) can be done using the free and open-source Apertium platform. The majority of the pipeline's modulus follows rules created by linguists and language developers and was built to employ the shallow transfer-based translation method [6].

In the structure of the encoder-decoder, the encoder compresses the sequences of input in NMT systems into a single vector representation. The decoder then employs the vector representation to create the sequence of outcomes [7]. There are numerous websites offering news articles in both English and Hindi, which are the two languages that are most commonly spoken around the world. Latent Dirichlet Allocation (LDA) has conducted numerous studies for undefined text classification. The most recent studies use a model of generative LDA to determine which texts are written in English [8]. Cross-Language Text Summarization (CLTS) is the procedure of analyzing the text in the source language to determine its meaning before creating a brief, accurate summary of that text in the target language. Similar to the Text Summarization (TS) field, the method created for CLTS can be categorized based on whether they are abstractive, compressive, or extractive [9]. There are

[1] *Department of Computer Science and Engineering, Proudhadevaraya institute of Technology, Hosapete, and Visvesvaraya Technological University, Belagavi, India*
ORCID ID : 0009-0006-6119-201X
[2] *Department of Computer Science & Engineering, Smt.Kamala and Sri Venkappa M Agadi College of Engineering and Technology, Laxmeshwar and Visvesvaraya Technological University, Belagavi, India*
ORCID ID : 0000-0003-4857-6079
[3] *Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bengaluru, India*
ORCID ID : 0009-0005-7888-6475
[4] *Department of Computer Science and Engineering, Don Bosco Institute of Technology, Bengaluru, and Visvesvaraya Technical University, Belagavi India*
ORCID ID : 0000-0002-3991-0031
[5] *Department of Computer Science and Engineering, KNS Institute of Technology, Bengaluru, and Visvesvaraya Technical University, Belagavi, India*
ORCID ID : 0000-0002-5131-9404
* *Corresponding Author Email: prashanthkogali@gmail.com*

numerous regional sign languages within a nation. By creating new and upgrading existing assistive technology for disabled people, including the deaf population, developed countries have been attempting to establish inclusive communities [10]. Quechua is a language of indigenous with limited parallel resources, classifying it as a Low-Resource Language (LRL) [11]. The widespread use of holes punched in the text for binding purposes is a distinctive feature of manuscripts from South-East Asia and India. Text lines have difficult gaps because of these holes. The manuscript's physical size is often smaller than that of other historical documents, leading to a layout with a dense amount of text [12]. Sanskrit requires significant processing to be explored in computer science and computational language analysis. To overcome this issue, the RNN-L2 regularization method is proposed for Sanskrit to Hindi translation language. In the following, the primary contribution of this paper is summarized:

- To construct a neural model with the architecture of encoder-decoder and attention mechanism, a parallel corpus of data from several sources was obtained, and the remaining data was manually constructed.
- The output of the linguistic tools was combined with NMT embedding feature metrics to determine the multiple meanings of a word translation in various contexts. Additionally, evaluated its capacity to effectively tokenize data and reduce the sparsity of data.
- Sanskrit to Hindi MT sentences were assessed based on precision, F1-Score, accuracy, and Recall.

The rest of the paper is organized as follows: Literature survey presented in section 2. Section 3 discusses the proposed methodology. Section 4 discusses the results. Section 5 describes the conclusion.

2. Literature Survey

S. Thara & Poornachandran [13] implemented a Word-Level Language Identification (WLLI) for Malayalam-English code-mixed data from social media websites like Youtube. This method was focused on the model of transformer BERT and its derivatives, DistilBERT, and CamemBERT for naturally perceiving word-level language. The WLLI method provides six labels to the code-mixed Malayalam-English data set: Malayalam (mal), undefined (undef), acronyms (acr), mixed (mix), English (eng), and universal (univ). The proposed method effectively captures the linguistic patterns and features identified in the code-mixed text, providing precise language identification. However, WLLI produces inaccurate language predictions when dealing with complex phrase structures or linguistic differences.

Surbhi Bhatia et al. [14] presented a Genetic Algorithm (GA) for the Hindi Word Sense Disambiguation (WSD). The ambiguous phrases left and right are employed, along with the dynamic configuration window function. Two context windows have been established while one context window was dynamic and contained the neighbor ambiguous words, the other context window was static and only contained the ambiguous word. The intricacy, instability, and vast search spaces associated with Hindi Word Sense Disambiguation were successfully handled by a genetic algorithm. However, manual tuning of several parameters such as selection criteria, population size, mutation, and crossover rates was required for the genetic algorithm.

Muskaan Singh et al. [15] introduced an MT system for Sanskrit-to-Hindi translation. The method develops a neural machine translation system using linguistic data from a rule-based feed. The method was innovative and suitable to any language of low resource with a rich morphology and covered multiple domains with minimum human involvement. The method achieves high performance by using both human and automatic measures and also generates effectively in terms of accuracy, response time, and speed. However, machine translation systems commonly operate on a sentence-by-sentences basis and struggle to incorporate contextual data effectively.

Jani Dugonik et al. [16] implemented a Hybrid Machine Translation (HMT) system that combines Statistical Machine Translation (SMT) and NMT to enhance NMT'S quality. For the Slovenian-English language pair, two NMT and SMT systems were established each for translation in one direction. The original sentences and translations were placed in the space of the same vector using a multilingual language model. HMT provides a higher-quality translation by utilizing the best characteristics of each system. However, multiple machine translation techniques were acquired for establishing and maintaining an HMT system.

Sahinur Rahman Laskar et al. [17] presented different kinds of negation effects for English-to-Assamese and Assamese-to-English translation by examining machine translation models. A rule-based method was provided for the step of data preprocessing to solve modal-verb negation difficulties that demonstrate significant improvement in terms of manual and automatic assessment scores. Machine translation models were trained and evaluated using language-specific resources which enhance their performance particularly when handling the sentences of negation. However, the method was challenging to translate negated statements effectively, which resulted in information loss or translation errors.

Shubham Dewangan et al. [18] introduced a Neural Machine Translation (NMT) for the Indian languages to enhance the translation. The effectiveness of a relatively limited number of Byte Pair Encoding (BPE) combined

operations in low-resource contexts, particularly for related languages was demonstrated. An effective training data augmentation method was introduced namely, phrase table injection, which combines NMT and SMT. The introduced method captures and utilizes contextual information efficiently. However, to attain the best translation performance, NMT models need a significant amount of high-quality data training.

Sitender & Seema Bawa [19] implemented a Sanskrit-to-English MT system by employing a hybridized form of rule-based and direct machine translation method. This method includes the language difference between Sanskrit and English, as well as a potential remedy to manage the differences. The Elasticsearch method has improved the machine translation system's ability to obtain information from multiple data dictionaries and rule bases utilized in system development. The implemented method achieves a fluency score, BLEU score, and adequacy score using natural language processing. However, high-quality parallel training data for Sanskrit-to-English translation was inadequate.

Md. Adnanul Islam et al. [20] presented a Corpus-based machine translator NMT and SMT for the translation of Bengali to English. Each corpus-based machine translation system incorporates the rule-based translator separately using various methods. The effectiveness of each integrating method was assessed using standard performance measures. The presented method provides continual improvements in translation quality by retaining or updating the models with new data. However, for optimal performance, the presented method requires a substantial amount of parallel training data.

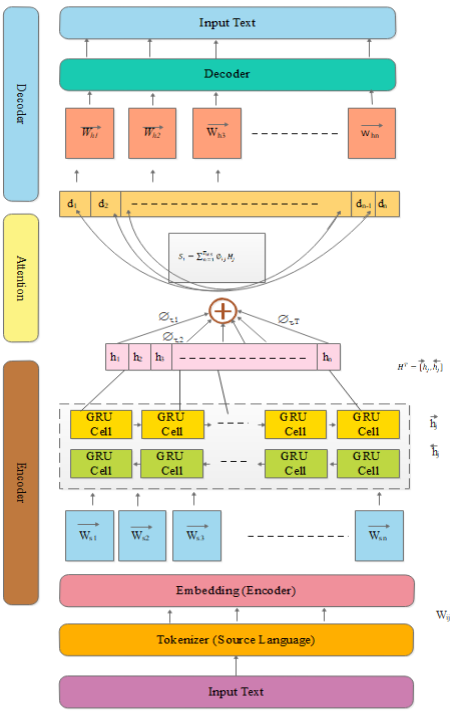

Fig. 1. Block diagram for the proposed method

3. Proposed Methodology

The Recurrent Neural Network (RNN)-L2 regularization method is proposed for Sanskrit to Hindi translation language. It includes a dataset, data pre-processing, adding linguistic components, a vector encoder that embeds input source sentences, a decoder that converts learning vectors into target sentences, and a web interface for providing users with access to the translation as a service. The overview of the proposed method is represented in Fig. 1.

3.1. Corpora dataset

A Corpora dataset is a kind of structured learning data that includes texts from a variety of sources including Wikipedia, the news, literature, tourism, judicial, healthcare, and the general domain. It has two types: Monolingual and parallel corpus. Corpora consists of a total of 8.8 billion tokens from news crawls across all 11 languages along with Indian English. The Bhagwad-Geeta which consists of 700 slokas and has been converted into Hindi, was also manually developed. Additionally, the Indian Languages Corpora Initiative (ILCI) project made 50,000 Sanskrit-Hindi corpus. The algorithm was trained using the whole parallel corpus of 162,760 parallel sentences.

3.2. Data Pre-processing

The data of the corpus is prepared for the NMT application and there are two steps to it; clean text and split text. Text is separated into sentences is an essential process in text cleaning. Then, the remaining non-numeric or non-alphabetical tokens are removed, along with any punctuation marks, and non-printable characters. Unicode characters are converted to ASCII value and all uppercase letters become lowercase. For each pair of imported datasets, these operations are performed on each sentence. The splitting operations are then applied to cleaned data. Different computation graphs were created because the dataset contained sentence pairs of varying lengths. Then, sentences of a similar length are divided into smaller batches after sorting sentences in a batch according to the length of sentence pairs. The training corpus is shuffled periodically by splitting the corpus into maximum batches and then splitting the corpus again into mini-batches. Applying a gradient for the parameter update completes the processing.

3.3. Rule-based machine translation system; extraction of linguistic features

The pre-processing rule-based MT system is performed based on main semantic, morphological, and syntactic regularities of the source and target language, which are mostly retrieved from dictionaries and grammar. The Sanskrit Consortium Project supported by MIT, used Anu-saaraka to divide various tools into several modules. The rule-based pipeline design for the translation of Sanskrit to

Hindi consists of 11 modules. To more effectively train the system, each module outputs a unique set of linguistic information to the neural-based encoder-decoder.

1. Pre-processing of user input: It enables the user's input, cleans it up, normalizes it, and translates the notations of input into WX notation. It then calls the system of MT, which executes the computation, and displays the outcomes.
2. Tokenizer: A character flow is inputted into the tokenizer, which separates the character into tokens (markers, words, and punctuation). It eliminates the data formatting and inserts a sentence tag. Linguistics referred to as morphology implies the study of words, including their inherent structure and meaning word. It has word streams, which are tokenized to give those words meaning.
3. Sandhi splitter: When the words of Sanskrit sandhi appear in the input text, it is called sandhi splitter. Both of these words and compound words are divided by it.
4. Morphological Analyzer: Words are divided into their roots and grammatical suffixes using the morphological analyzer. There are various units, and each one serves grammatical and semantic purposes. Additionally, it offers inflectional analysis, reduces the response, handles unrecognized words using the analysis of local morph, and generates a derived roots derivational analysis.
5. Parsing: For simple translation from one language to another, a parser is employed as an interpreter or compiler to break data into smaller components. Word or token sequences serve as the input for parsers. These inputs are transformed into a parse tree format. The tree structure containing labels for verbs, nouns, and their corresponding properties, transforms the source into the language of the target. In addition to karaka analysis, morph analysis according to context is carried out. The relationship between the verb and its participants is named and identified in computational Paninian grammar.
6. Shallow Parsing: When the parser detects an input failure, it occurs a minimal amount of phrase parsing and generates cleaned morph analysis for the subsequent layer.
7. Word Sense Disambiguation (WSD): The modules use the input phrase terms vibhakti and lakara to execute WSD. It displays a word's proper Sanskrit meaning.
8. Parts of Speech Tag (POS): It includes tags for each word's components of speech such as nouns, verbs, or adjectives.
9. Chunker: A minimum word grouping such as noun, verb, or adjective is carried out during this step. A suitable chunk tag is given to it on a rule basis.
10. Hindi Lexical Transfer: The Sanskrit Lexicon is translated into Hindi by employing the dictionary to find the roots of the words. The format of the output is determined by the Generator of Hindi, which produces the Hindi language output as it relates to Sanskrit. If translation fails, this module executes transliteration.
11. Hindi Generator: An agreement check among a noun, verb, and adjective in the language of the target is performed by a sentence-level generator during this step. The 'ne' marks for vibhakti are added, while 'ko' markers are dropped at the appropriate locations. The process of final generation includes grammatical properties, their associated suffixes, and the root words that occur alone with them.

As a result, linguistic principles and techniques are used to translate each Sanskrit word into its equivalent Hindi word. Additionally, the subsequent phase receives this data. The input that is passed on to the following stage is transformed into Comma-Separated Values (CSV), which is ideal for the development of the model, training, and fitting values for the architecture of neural-based encoder-decoder for determining the Sanskrit to Hindi translation words. The outcomes of this linguistic tool are included as features for source sentence input encoding.

3.4. RNN encoder-decoder with attention mechanism embedding extracted features

RNN-L2 regularization is proposed for Sanskrit to Hindi translation. It can be used with any language of low-resource because of minimal parallel data sources. GRU cells and enhanced NMT with an attention mechanism were utilized for the computation. With an attention mechanism for both the encoder and decoder, the implementation makes use of stacked bi-directional RNN layers. The source sentence $W_s = W_{s1,\ldots}W_{sn}$ into sequence variable of context vectors $S = h_1, h_2, h_3, \ldots., h_n$. The decoder constructs the target phrase by decoding the context vector S_i. By increasing the probability of the target word given the previously created word h_{i-1}, the hidden state decoder d_{si} and vector context $s_i P(h_i \backslash d_{si,} h_{i-1}, s_i)$ can be produced.

3.4.1. Encoder

The Sanskrit language source sentence is given in (1)

$$W_s = W_{s1,}W_{s2,}W_{s3}, \ldots W_{sz}, S_i \in R^{K_s} \quad (1)$$

Hindi target sentences: from the parallel corpus provided in (2)

$$W_h = W_{h1}, W_{h2}, W_{h3,\ldots..}W_{hi,}h_x \in R^{k_h} \quad (2)$$

Where

z, x – length of the input and output sentences

k_s, k_h- vocabulary size

The model initially tokenizes W_s to provide an input representation where sequences probability of $T(W_{s1}, W_{s2,\ldots.}W_{sn})$ is represented as $P_1(W_{s1,\ldots\ldots.}, W_{st})$. Instead of considering all previous words, it is typically dependent on a word window. Because the input document's places affect the number of words before the previous word (W_1) in (3)

$$P_1\left(W_{s1}, W_{s2\ldots,}W_{s_{tz}}\right) = \prod_{i=1}^{t} P(W_{s1,\ldots\ldots.}, W_{s_{i-1}})$$

$$\approx \prod_{i=1}^{t} P\left(W_{s_i}\middle|W_{s_1\ldots.}W_{s_{z-1}}\right)\ldots.W_{s_{z-1}}$$

(3)

Since text data cannot be directly applied to neural networks. By embedding layers, text is transformed into integer tokens or numbers, which are then transformed into vectors. The tokenizer is employed for source and target language by defining the maximum number of vocabulary terms. After being transformed into a series of integer tokens, the dataset is then padded, trimmed, and saved as NumPy arrays. The encoder computes embedded vectors $(W_{s1,}W_{s2,}W_{s3,\ldots.W_{sz,}})$ for the computation of the hidden layer using the tokenizer's output as arrays. These vectors, which range in value from 1 to -1, correspond to words with similar semantic meanings. Forward RNN computes the hidden states $h_{1,}h_{2,}h_{3,\ldots.}h_{\tau_j}$ by processing the input sentence from beginning to end f. The backward RNN analyzes the text in reverse order to determine the hidden states $h_{1,}h_2, \ldots h_{\tau_j}$. An annotation vector $H_i = [h_j^T;\ h_j^T]$ is created by combining these hidden states, i.e., backward and forward. For each input word s_z, the conventional encoder includes an embedding lookup and steps for mapping via hidden states in (4)

$$H_j = f(h_i - 1, \bar{E}W_{sn}) \quad (4)$$

The encoder calculations are deeply layered as shown in (5), (6) in the manner described below. For each input word, the traditional encoder contains an embedding lookup.

$$h_{t,1} = f_1(h_{t-1}, 1, W_s t) \quad (5)$$

For $i > 1$

$$h_{t,i} = f_{h_{t-1,i,}h_{t,i-1}} \quad (6)$$

Where

$h_{t-1,i}$ – value of the previous timestamp

$h_{t,i-1}$ – value for the preceding layer in the sequence

The input sentence is calculated by processing backward and forward RNNs, and is contained in the context vector s_i. For the encoder and decoder function, Gated Recurrent Unit (GRU) is employed. To facilitate the detection of long-term dependence by RNN, GRU is built to have the longer-lasting memory. GRU employs input W_{st} and the previous hidden state h_{t-1} to produce the next hidden state h_t.

Equations (7), (8), (9), (10), (11), and (12) shows the update gate, the reset, the new memory and the state of hidden, for all I words in a phrase

$$up_i = \sigma(W_{up}\overline{E_{sı}} + O_{up}h_{i-1}) \quad (7)$$

$$res_i = \sigma(W_{res}\overline{E_{sı}} + O_{res}h_{i-1}) \quad (8)$$

$$h_i = \tanh\left(W\overline{E_{sı}} + O[res_i \odot h_{i-1}\right] \quad (9)$$

$$h_i = (1 - up_i) \odot h_{i-1} + up_i \odot h_i) \quad (10)$$

Where

d – word embedding dimensionality

u – number of hidden units $\bar{E} \in R^{dxks}$

σ – function of the logistic sigmoid

$$W, W_{up,}W_{res} \in R^{uxd} \quad (11)$$

$$O, O_{up,}O_{res} \in R^{uxu} \quad (12)$$

A bidirectional recurrent neural networks backward states are calculated identically for the update gate in (13), the reset gate in (14), the new memory in (15), and the state of hidden, for all I words in an (16)

$$up_i = \sigma(W_{up}\overline{E_{sı}} + O_{up}h_{i-1}) \quad (13)$$

$$res_i = \sigma(W_{res}\overline{E_{sı}} + O_{res}h_{i-1}) \quad (14)$$

$$h_i = \tanh\left(W\overline{E_{sı}} + O[res_i \odot h - i - 1\right] \quad (15)$$

$$h_i = (1 - up_i) \odot h_{i-1} + up_i \odot h_i) \quad (16)$$

The combined forward and reverse states are represented as $H_i = [h_j^T;\ h_j^T]$.

3.4.2. Adding linguistic features to the encoder

To train recurrent neural networks, the system incorporates linguistic elements that were taken from the rule-based pipeline design. A unique vector word embedding s_{zy} is present for each feature. Integrating all of these word vectors $E\epsilon R^{dyXky}$ with d_k as the sum of all embedding feature dimensions and ky as the K^{th} feature vocabulary size. The entire embedding size is later combined with these embeddings because their lengths are compatible. These linguistic features are retrieved and multiplexed onto the input embedded sentence vectors. All other model functionality and parameters remain the same, only the encoder change is made as in (17), resulting in an outstanding enhancement in the output fluency.

$h_l = \tanh\,(W \prod_y^F \overline{E_y} s_{zy} + Oh_{l-1})$
(17)

3.4.3. Attention Mechanism

The attention layer covers the gap between the decoder, which generates a s_i vector context at each time step t_i and the encoder, which produces a word sequence representation in the form of $h_j = (h_i, h_i)$. By computing the effect of word representation (h_i, h_i), it determines the connection between the input word W_s and the subsequent output word W_h. The context vector can be described as the weighted annotations sum h_i. For this, required to identify the model of alignment a_{ij}, or output position score to the input position, as shown in (18), (19), (20). It uses the input Sanskrit sentences j^{th} annotation and hidden state d_{i-1}.

$$a_{ij} = J_a^{\tau}\tanh\,(W_a d_{i-1} + O_a h_j) \quad (18)$$

$$\alpha_{ij} = \frac{\exp\,(a_{ij})}{\sum_{y=1}^{T_S} \exp\,(a_{iy})} \quad (19)$$

$$S_i = \sum_{j=1}^{ts} \alpha_{ij} h_j \quad (20)$$

Where S – feed-forward neural network

$W_a \in R^{n_1'}, O_a \in R^{n'Xn}, J_a \in R_{n'X2n}$ are matrics weight. Using the function of SoftMax activation, the computed value of scalar attention is normalized so that the sum of all input words is 1.

3.4.4. Decoder

The decoder outputs a new word prediction W_{hi} and new outcome hidden state decoder at each time step t by employing a previously hidden state phrase d_{i-1}, an input context representation s_i, and a previous word embedding outcome $E_{h_{i-1}}$. In (21), the initial hidden state is calculated.

$d_0 = f(w_d h_1)$
(21)

The hidden state d_i is calculated given the encoder's annotation in (22), updated in (23), and reset in (24)

$$d_i = tan\bar{h}\left(WEh_{y_{i-1}}\right) + O[res_i + d_{i-1}] + S_{s_i}) \quad (22)$$

$$up_i = \sigma(W_{up}E_{h_i-1} + O_{up}d_{i-1}S_{up}s_i) \quad (23)$$

$$res_i = \sigma(W_{res}Eh_{i-1} + O_{res}h_{i-1} + S_{res}S_i) \quad (24)$$

Where u – number of hidden units

d – word embedding dimension

E – embedded word matrix for the target language

Weight matrices are $W, W_{up,} W_{res} \in R^{uXd}, 0, O_{up,} O_{res} \in R^{uX2d}$. The decoder's hidden state d_{i-1}, input context s_i, and prior output word embedding h_{i-1} as in (25), provide the basis of the prediction vector P_i for an output word.

$$P_i = softmax(O_{ot_{d_i-1}} + V_{ot}E_{h_i-1} + S0s_i \quad (25)$$

Where, $V_{ot} \in R^{2lXd}, O_{ot}R^{2lXu}, C_oR \in 2lX2u$ is an embedding matrix of output word.

As a result of the encoder state progression from d_{i-1} to d_i being fragmented when utilizing d_{i-1} instead of d_i for the output word prediction P_i in (26), the $E_{W_{h_{i-1}}}$ condition is repeated

$$P_i = [maxP_i, 2\bar{j} - 1, i, \bar{2}j]_{j=1\ldots\ldots,l}^{\tau} \quad (26)$$

Even training is carried out according to the network's knowledge of the proper output, which is given a value of higher probability as in (27)

$$Prob(h_i|d_{i-1,}s_i)\alpha(h^{\tau}W_o pi) \quad (27)$$

Function of activation SoftMax is employed to transform a raw vector into a range of probabilities with a total value equal to one. Additionally, ReLU is utilized, which combines input to produce the next hidden state. The function of activation is supplied to the model to better predict the target variable and it also functions as a rectifier.

3.4.5. L2 Regularization

L2 regularization techniques operate by introducing a norm penalty parameter to the objective function as shown in (28), to restrict the model capacity.

$$\hat{J}(\theta) = J(\theta) + \lambda R(w) = J(\theta) + \lambda \sum_i |w_i^2| \quad (28)$$

Where the norm penalty term's $R(w)$ relative contribution to the common objective function $J(\theta)$is weighted. On the data training, this parameter reduces the original and objective function size J. A measurement of the parameter's size w_i when the regularized objective function is minimized $\hat{J}$ by the training procedure. The L2 regularization makes the learning algorithm "perceive" the input to have a high variance, which causes the feature weight whose covariance with the target output is smaller than this additional variance to be reduced.

4. Experimental Setup and Results

To obtain a processing speed of about 2500 words per second, the RNN-L2 regularization method is processed by employing a heavily configured core GPU with 32GB RAM. Normal systems cannot operate at this speed since it will take two hours to complete one epoch. So, NVIDIAGeforceGTX1050 and QuadroK6000 are used together with a GPU that is well configured. Both automated metrics and evaluations by humans were used to evaluate the proposed method's performance.

4.1. Evaluation Metrics

- Bilingual Evaluation Understudy (BLEU) – It is an essential parameter for assessing sentence translation accuracy in comparison to human generated reference translation as shown in (29)

$$BLEU = \min\left(1, \frac{output_{length}}{Reference_{length}}\right)\left(\prod_{i=1}^{4} precision_i\right) \quad (29)$$

- Metric for Evaluation of Translation with Explicit ORdering (METEOR) - It is a metric to evaluate the output of machine translation. The metric is based on the harmonic mean of unigram recall and precision as shown in (30), and (31)

$$F_{mean} = \frac{10PR}{9+RP} \quad (30)$$

$$METEOR = F_{mean}(1-p) \quad (31)$$

4.2. Experimental Results

The model update is affected by the corpora's length sentence. Fig. 2 shows that as the length of sentences increases in the corpus training, the number of weight updates dramatically rises beyond a point, and later drops off following the sentence length reaches 20 words. Table 1. shows the Sentence Length affecting updates on different sentence lengths ranging from 0 to 50 respectively.

Table 1. Sentence Length affecting updates

Sentence Length	Updates
0	0
10	6
20	8.5
30	6.8
40	4.9
50	3

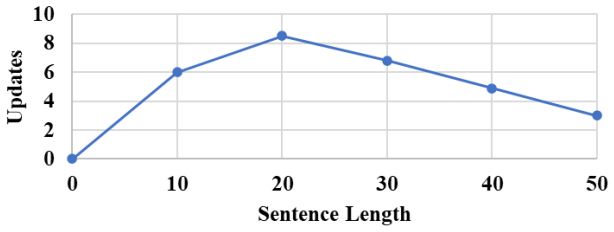

Fig. 2. Sentence Length affecting updates

Table 2. shows the Sentence Length affecting epochs on different sentence length ranges from 0 to 50 respectively. The effect of sentence length on the number of training set iterations, or epochs that are executed in Fig. 3. The graph makes it highly apparent that there are a number of epochs reduced following the point (20 sentences long).

Table 2. Sentence Length affecting Epochs

Sentence Length	Epochs
0	0
10	4.2
20	6.5
30	5
40	3.8
50	2.5

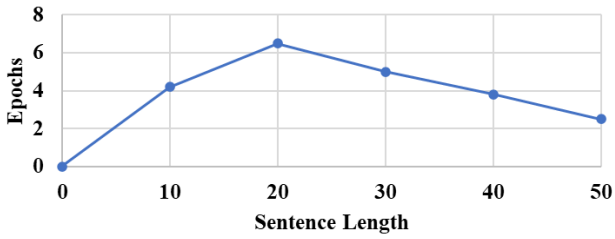

Fig. 3. Sentence Length Affecting Epochs

The sentence length affecting model time is shown in Fig. 4. It indicates that the period fluctuates significantly. The model training time is stable for sentences between 10 to 20, but increases rapidly for sentences between 20 to 30. Table 3 shows different sentence lengths ranging from 0 to 50 respectively. In conclusion, the updates, epochs, and time are limited by sentences that are no longer than 20 words. There is a decrease in the graphs when the corpus sentence length crosses this limit.

Table 3. Sentence Length Affecting Time (H)

Sentence Length	Time (H)
0	0
10	110
20	110
30	250
40	120
50	123

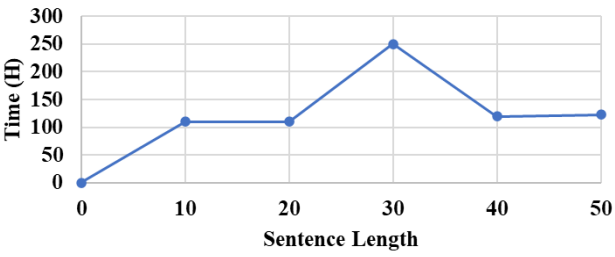

Fig. 4. Sentence Length Affecting Time (H)

The BLEU score fluctuates with the size of the beam as shown in Fig. 5. The inference process uses a beam search to identify the most probable word order for each translation. In comparison to conventional statistical machine translation beam sizes, the beam issue related to the translation of neural machines affects comparatively small beam sizes. The beam size 1 to 4 exhibits constant variation in the BLEU score, whereas the BLEUs increased between 5 to 10. The BLEU score decreases if there is a significant rise in beam size. Therefore, in training sentence length is normalized to limit the size of the beam. Table 4. shows that the BLEU varies with the size of the beam as below.

Table 4. BLEU varies with the size of the beam

Size of Beam	BLEU
1	39
2	42
3	43
4	44
5	55
6	58
7	61
8	61
9	57
10	53

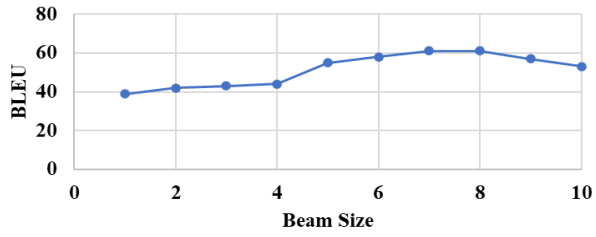

Fig. 5. BLEU varies with the size of the beam

Fig. 6. demonstrates the Development probability varies with Sentence Length. The plot shows that the development probability rises to 20 sentence length and then drops. As a result, dividing larger sentences or normalizing sentences longer than 20 words would be an appropriate technique. Table 5. shows the development probability varies with sentence length on different ranges from 0 to 50 respectively.

Table 5. Development probability varies with sentence Length

Sentence Length	Dev-prob
0	0
10	45
20	55
30	35
40	48
50	40

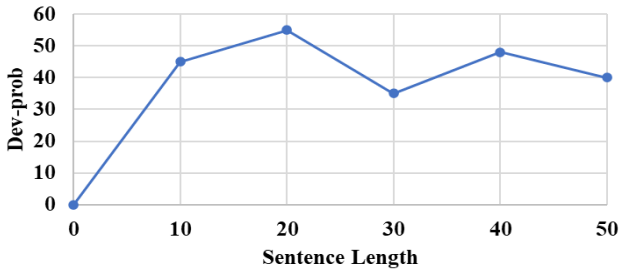

Fig. 6. Development probability varies with Sentence Length

Table 6 shows different sentence lengths ranging from 0 to 50 respectively. Fig. 7 simulates that the training probability varies with sentence length. As seen, the training probability drops gradually. Therefore, shorter sentences would be more effectively modeled to raise the training probability.

Table 6. Training probability varies with Sentence Length

Sentence Length	Training probability
0	0
10	45
20	30
30	38
40	30
50	42

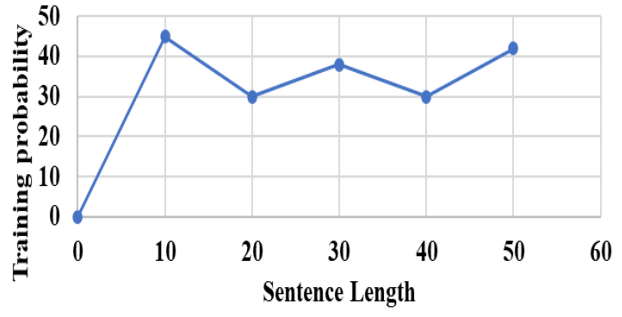

Fig. 7. Training probability varies with Sentence Length

4.3. Comparative Analysis

The comparative analysis includes methods, BLEU, and METEOR. Table 7. shows that the comparative analysis with the existing methods

Table 7. Comparative Analysis with Existing Methods

Author	Dataset	Methods	BLEU (%)	METEOR (%)
Muskaan Singh [15]	Corpora	MT system	75	61
Jani Dugonik [16]	Corpora	Hybrid MT system: English-Slovenian	42.9	61.5
		Slovenian-English	47.9	70.9
Proposed method	Corpora	RNN-L2 regularization	76	72

The existing method Sanskrit-Hindi MT system [15] has a BLEU score of 75% and METEOR has 61%. The hybrid MT system [16] has two techniques: English-Slovenian has 42.9%, 61.5% of BLEU and METEOR. Slovenian-English has 47.9 and 70.9 of BLEU and METEOR. When compared with the existing method, the RNN-L2 regularization achieves 76%, 72% of BLEU and METEOR.

4.4. Discussion

This section provides a discussion about the Recurrent Neural Network (RNN)-L2 regularization method and compares those results in the comparative analysis section 4.3. The major goal of this study is to translate the Sanskrit-Hindi language using RNN-L2 regularization. To construct a neural model with the architecture of encoder-decoder and attention mechanism, a parallel corpus of data from several sources was obtained, and the remaining data was manually constructed. A neural machine translation system is trained using the linguistic information from the rule-based input. The output of the linguistic tools from the conventional rule-based method was combined as NMT feature embedding matrices to determine the multiple meanings of a word translation in various contexts. Additionally, evaluated its capacity to effectively tokenize data and reduce the sparsity of data. L2 regularization techniques operated by introducing a norm penalty parameter to the objective function to restrict the model capacity. The efficacy of RNN-L2 regularization method is demonstrated by employing the dataset of Corpora. The data of the corpus are prepared for the NMT application. There are two steps to it; clean text and split text. Separating the text into sentences is an essential process in text cleaning. Then, the remaining non-numeric or non-alphabetical tokens are removed, along with any punctuation marks, and non-printable characters. When compared with the existing methods such as machine translation system, and hybrid machine translation system, the RNN-L2 regularization achieves 76%, 72% of BLEU and METEOR.

5. Conclusion

In this paper, the RNN-L2 regularization method is proposed for Sanskrit to Hindi translation. The proposed method is unique and appropriate to any language pair with limited resources and linguistic knowledge. The features are taken from the language rule and then used to train an RNN. A neural machine translation system is trained using the linguistic information from the rule-based input. The proposed method can be used with any morphologically rich low-resource language. High performance is achieved via performance evaluation using both manual and automated measures. When compared to the current rule-based systems, the RNN-L2 regularization is quicker and more effective. When there is no rule match, the rule-based model fails to generate outcomes, but RNN-L2 regularization returns the optimal answer. The RNN-L2 regularization achieves 76%, and 72% of BLEU and METEOR when compared to the existing methods of machine translation system, hybrid machine translation system. In the future, the term overfitting issues in training data will be reduced using the RNN-L2 regularization method.

Author contributions

Prashanth Kammar: Conceptualization, Methodology, Software, Writing-Original draft preparation, **Parashuram Baraki:** Visualization, Software, Writing-Reviewing and Editing, **Sunil Kumar Ganganayaka:** Visualization, Software, Investigation, Writing-Reviewing and Editing, **Manjunath Swamy Byranahalli Eraiah:** Visualization, Field study, Investigation, **Kolakaluri Lakshman Arun kumar:** Visualization, Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] L. Benkova, D. Munkova, L. Benko, and M. Munk, "Evaluation of English–Slovak neural and statistical machine translation," *Applied Sciences*, vol. 11, no. 7, p. 2948, Mar. 2021, https://doi.org/10.3390/app11072948.

[2] K. D. Garg, S. Shekhar, A. Kumar, V. Goyal, B. Sharma, R. Chengoden, and G. Srivastava, "Framework for Handling Rare Word Problems in Neural Machine Translation System Using Multi-Word Expressions," *Applied Sciences*, vol. 12, no. 21, p. 11038, Oct. 2022,

https://doi.org/10.3390/app122111038.

[3] S. Saini and V. Sahula, "Setting up a neural machine translation system for English to Indian languages," in: *Cognitive Informatics, Computer Modelling, and Cognitive Science*, Academic Press, 2020, pp. 195-212, https://doi.org/10.1016/B978-0-12-819443-0.00011-8.

[4] M. Brour, and A. Benabbou, "ATLASLang NMT: Arabic text language into Arabic sign language neural machine translation," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 9, pp. 1121-1131, Nov. 2021, https://doi.org/10.1016/j.jksuci.2019.07.006.

[5] J. X. Huang, K. S. Lee, and Y. K. Kim, "Hybrid translation with classification: Revisiting rule-based and neural machine translation," *Electronics*, vol. 9, no. 2, p. 201, Jan. 2020, https://doi.org/10.3390/electronics9020201.

[6] T. Khanna, J. N. Washington, F. M. Tyers, S. Bayatlı, D. G. Swanson, T. A. Pirinen, I. Tang, and H. Alòs i Font, "Recent advances in Apertium, a free/open-source rule-based machine translation platform for low-resource languages," *Mach. Transl.*, vol. 35, no. 4, pp. 475-502, Dec. 2021, https://doi.org/10.1007/s10590-021-09260-6.

[7] L. H. Baniata, I. K. Ampomah, and S. Park, "A transformer-based neural machine translation model for Arabic dialects that utilizes subword units," *Sensors*, vol. 21, no. 19, p. 6509, Sep. 2021, https://doi.org/10.3390/s21196509.

[8] A. Srivastav and S. Singh, "Proposed model for context topic identification of english and hindi news article through LDA approach with NLP technique," *J. Inst. Eng. India Ser. B*, pp. 1-7, Apr. 2022, https://doi.org/10.1007/s40031-021-00655-w.

[9] E. L. Pontes, S. Huet, J. M. Torres-Moreno, and A. C. Linhares, "Compressive approaches for cross-language multi-document summarization," *Data & Knowledge Engineering*, vol. 125, p. 101763, Jan. 2020, https://doi.org/10.1016/j.datak.2019.101763.

[10] N. S. Khan, A. Abid, and K. Abid, "A novel natural language processing (NLP)–based machine translation model for English to Pakistan sign language translation," *Cognit. Comput.*, vol. 12, pp. 748-765, Jul. 2020, https://doi.org/10.1007/s12559-020-09731-7.

[11] J.E. Ortega, R. Castro Mamani, and K. Cho, "Neural machine translation with a polysynthetic low resource language," *Mach. Transl.*, vol. 34, no. 4, pp. 325-346, Dec. 2020, https://doi.org/10.1007/s10590-020-09255-9.

[12] D. Banik, A. Ekbal, P. Bhattacharyya, and S. Bhattacharyya, "Assembling translations from multi-engine machine translation outputs," *Appl. Soft Comput.*, vol. 78, pp. 230-239, May 2019, https://doi.org/10.1016/j.asoc.2019.02.031.

[13] S. Thara and P. Poornachandran, "Transformer based language identification for Malayalam-english code-mixed text," *IEEE Access*, vol. 9, pp. 118837-118850, Aug. 2021, https://doi.org/10.1109/ACCESS.2021.3104106.

[14] S. Bhatia, A. Kumar, and M. M. Khan, "Role of genetic algorithm in optimization of Hindi word sense disambiguation," *IEEE Access*, vol. 10, pp. 75693-75707, Jul. 2022, https://doi.org/10.1109/ACCESS.2022.3190406.

[15] M. Singh, R. Kumar, and I. Chana, "Improving neural machine translation for low-resource Indian languages using rule-based feature extraction," *Neural Comput. Appl.*, vol. 33, pp. 1103-1122, Feb. 2021, https://doi.org/10.1007/s00521-020-04990-9.

[16] J. Dugonik, M. Sepesy Maučec, D. Verber, and J. Brest, "Reduction of Neural Machine Translation Failures by Incorporating Statistical Machine Translation," *Mathematics*, vol. 11, no. 11, p. 2484, May 2023, https://doi.org/10.3390/math11112484.

[17] S. R. Laskar, A. Gogoi, S. Dutta, P. K. Adhikary, P. Nath, P. Pakray, and S. Bandyopadhyay, "Investigation of negation effect for English–Assamese machine translation," *Sādhanā*, vol. 47, no. 4, p. 238, Nov. 2022, https://doi.org/10.1007/s12046-022-01965-5.

[18] S. Dewangan, S. Alva, N. Joshi, and P. Bhattacharyya, "Experience of neural machine translation between indian languages," *Mach. Transl.*, vol. 35, no. 1, pp. 71-99, Apr. 2021, https://doi.org/10.1007/s10590-021-09263-3.

[19] Sitender and S. Bawa, "A Sanskrit-to-English machine translation using hybridization of direct and rule-based approach", *Neural Comput. Appl.*, vol. 33, pp. 2819-2838, Apr. 2021, https://doi.org/10.1007/s00521-020-05156-3.

[20] M. A. Islam, M. S. H. Anik, and A. A. A. Islam, "Towards achieving a delicate blending between rule-based translator and neural machine translator," *Neural Comput. Appl.*, vol. 33, pp. 12141-12167, Sep. 2021, https://doi.org/10.1007/s00521-021-05895-x.