Neural Computing and Applications (2021) 33:1103–1122
https://doi.org/10.1007/s00521-020-04990-9

ORIGINAL ARTICLE

Improving neural machine translation for low-resource Indian languages using rule-based feature extraction

Muskaan Singh[1] · Ravinder Kumar[2] · Inderveer Chana[2]

Received: 29 July 2019 / Accepted: 2 May 2020 / Published online: 2 June 2020

Abstract
Languages help to unite the world socially, culturally and technologically. Different natives communicate in different languages; there is a tremendous requirement for inter-language information translation process to transfer and share information and ideas. Though Sanskrit is an ancient Indo-European language, a significant amount of work for processing the information is required to explore the full potential of this language to open vistas in computational linguistics and computer science domain. In this paper, we have proposed and presented the machine translation system for translating Sanskrit to the Hindi language. The developed technique uses linguistic features from rule-based feed to train neural machine translation system. The work is novel and applicable to any low-resource language with rich morphology. It is a generic system covering various domains with minimal human intervention. The performance analysis of work is performed on automatic and linguistic measures. The results show that proposed and developed approach outperforms earlier work for this language pair.

Keywords Recurrent neural network · Linguistic feature extraction · Deep learning · Rule-based system · Sanskrit–Hindi translation

1 Introduction

Machine translation (MT) is a sub-field of computational linguistics and Natural Language Processing (NLP) which enables an automatic translation of sentences or documents from one language to another. It aims at reducing the language barriers of human communication belonging to different linguistic backgrounds. Language perplexity has a tremendous impact on several aspects of human subsistence, and the same can be mitigated with effective use of MT. It endeavours at minimising the involvement of human-being. Although machine-generated output differs from human translation but is understandable. It manifests its effectiveness by producing grammatically and semantically fluent output. The classical translation approaches delve the deep insight of linguistic knowledge of source language and cognitively translate to target language word by word. The approaches for MT are categorized predominantly into rule-based, i.e. based on linguistic handcrafted rules [1] involving transfer-based mechanism [2] and interlingua mechanism [3], corpus-based approach that is entirely based on corpus, i.e. statistical phrase-based [4] and neural-based [5]. Even some approaches are less used nowadays, i.e. example-based approach [6], knowledge-based [7].

✉ Muskaan Singh
muskaan_singh@thapar.edu

Ravinder Kumar
ravinder@thapar.edu

Inderveer Chana
inderveer@thapar.edu

1 Language Engineering and Machine Learning Research Labs, CSED, Thapar Institute of Engineering and Technology (TIET), Patiala, India

2 CSED, Thapar Institute of Engineering and Technology (TIET), Patiala, India

1.1 Need for machine translation systems

The tremendous increase in industrial growth over the past decades has a significant impact on the global MT market which enables content to be available in all regional languages across the globe. Computational activities have become mainstream nowadays. As the internet opens up the wider multilingual and global community, research and development in MT continue to grow at a rapid rate. MT system is required in small, medium and large organisations. Some deal with only specific domain, but some provide services in multiple domains. Various fields are requiring domain-specific services such as government, software and technology, healthcare, legal, military and defence, e-commerce, finance and many more. The other online MTS, i.e. commercial systems deal with instant translation where source language is converted into target language for text, image and audio data. These MT systems are generic, light-weight, cloud-based and have high accuracy. It offers translations like Text-to-text, Speech to Speech, Text to Speech, Speech to Text and Image to Text. These commercial systems are not economical; therefore, it necessitates a need for economic MT system providing translation services.

1.2 Objective of the research work

India is a country of enormous language diversity, with more than 1.3 billion people with 29 official languages and more than 12 scripts; hence, there is a need to provide a translation of the content from one language to another language. One of the main requirements in Sanskrit computational linguistics is to translate the life-transforming stories (epics), Vedas and so forth to make them available in other languages, for the public at large. As per the literature survey, the machine translation system (MTS) using Sanskrit as a source or target language is in developing stages as very few systems have been developed using rule-based and statistical system [8, 9]. This approach is not adequate for extending the system to generic and huge domains as requires linguistic insights. Therefore, there is a need to develop a machine-engineered generic MTS for translating Sanskrit–Hindi. We have merged the rule-based linguistic approach with a recurrent neural network. Although neural is a new approach to MT, it has produced promising results in the translation domain. It also has some pitfalls which we would overcome by merging along with rules. Thus, a hybrid system combining the best of neural-based and rule-based is developed and presented in this paper.

1.3 Problem domain

According to the census of India in 2018, Sanskrit is the mother tongue of 24,821 people and Hindi of 52,83,47,193 people, i.e. 43% of total languages in India [10]. Reasons for choosing Sanskrit for translation purpose is the richness of its scientific literature with extensiveness and comprehensive analysis, structured approach and traditional grammar [11]. There are numerous characteristics of this language, some of them have been listed.

- It is also considered as the ’donor’ of almost all Indian languages. Most of the languages have been derived from Sanskrit either partially or fully.
- The vast reserves in this language could be converted into other languages [12]. As this language comprises of rich literature, Ayurveda, Vedas which can be produced in other languages to improve its accessibility.
- It holds a rich grammar confined by Panini near 2500 years ago formulating 3949 rules, which extended later on [13].
- It has the strongest and non-ambiguous grammar [14]. Many people have attempted to write a grammar for Sanskrit language using the Paninian framework and used it to develop translation system [15].
- Panini focused on decoding the information contained in the language string of particularly given input language by Karaka (syntax-semantics) relations and not thematic roles [16]. It also highlights the importance of case markers, postpositions and word-order. The central element of a semantic model in Paninian framework is that every verbal root (dhaatu) denotes an action consisting of an activity (or vyaapara), and a result (or phala). The result is a state reached after completion of an action. An activity consists of steps that are performed by different participants or Karakas involved in the action.
- The concept of Karaka relation is a central theme to Paninian grammar. These Karaka relations refer to syntactic-semantic relations, and on a surface level, it highlights syntactic information and also captures semantic information at a deeper level.
- The Sanskrit grammar is termed as ‘Father of Informatics’ as it builds a relationship between speech and utterance of speaker and meaning derived by the listener [17].
- The primary objective of Sanskrit Paninian grammar is to form a theory of human natural language communication.
- Sanskrit and Hindi belong to the same Indo-Aryan family [18]. They both have structural and lexical similarity as Hindi inherits from Sanskrit.
- Sanskrit has the rich and structured grammar in the form of Panini Astadhayayi whereas in Hindi such parallel grammar does not exist. Therefore, it becomes difficult to map the divergence between these two languages.
- The non-existence of parallel grammar leads to exceptional cases which uncover linguistic generalisations such as Vibhakti in Hindi. The cases where Vibhakti in Sanskrit and Hindi diverges [18] are optional,

exceptional, differential, alternative, non-Karaka, verb and complex-predicate divergence.

Despite these features, choosing Sanskrit as a source language is difficult on using both rule-based as well as neural-based approach. In the rule-based approach, parsing fails due to its synthetic nature in which single word can run up to 32 pages. Whereas in case of training the translation system using NMT approach lead to the high occurrence of Out-of-vocabulary words. These words are morphologically rich words, carrying multiple meanings according to context. The work presented in this paper clearly outlines these challenges and overcomes them by providing a sound translation.

1.4 Challenges in neural machine translation

There are many challenges for neural machine translation (NMT) as compared to other approaches. This new trend in MT has emerged neural network in yielding quite effective translation systems. Besides, the promising result provided by NMT, there are few challenges listed.

- NMT systems undertake fluency rather than adequacy leading to out of domain low-quality translation.
- NMT systems are directly proportional to the training data. Therefore, rich-resource language pair exhibits better performance.
- NMT systems face problem in translating low-frequency words with an inflected category such as verbs. It has encountered that NMT systems applied with Byte-Pair Encoding perform better than SMT systems on low-frequency words.
- NMT systems result for higher translation quality for smaller sentences rather than longer sentences.
- The word alignment performed by the attention model in NMT does not produce desirable results.
- In decoding phase, translation quality is improved by beam search only for narrow beams and decreases for larger beams.
- NMT system requires better analytics for its computation as the working is not interpreted.

These challenges are overcome in our proposed approach by merging NMT with linguistic features extracted by applying the rule-based approach.

1.5 Contributions

Neural machine translation (NMT) has been outperforming traditional statistical phrase-based and RBMT. Motivating from its performance, we implemented a neural model along with linguistic rich features from rule-based System for Indian language pair, i.e. Sanskrit–Hindi.

- Initially, parallel corpus was gathered which was available from different sources and rest of the data was manually created for building a neural model using encoder–decoder architecture with attention mechanism [5].
- We trained and tested the NMT system with different models, activation functions, training data, epochs, sentence lengths to yield better accuracy of the proposed system.
- We then merged linguistic tools output from the classical rule-based approach as features embedding matrices in NMT to test whether it guides for the disambiguation translation of words as the same word may have a different meaning in a different context. We also tested whether it reduces the data sparseness and performs meaningful tokenization.
- Performance testing was performed to compare rule-based, neural and hybrid systems on four measures, i.e. BLEU, *F*-measure, METEOR, Word-Error-Rate.
- Various Sanskrit to Hindi MT sentence were evaluated corresponding to reference translation based on BLEU score, Precision, Recall, F-Measure, WER, F-mean, Penalty and Meteor.
- Human-based evaluation was also performed based on the grammatical category. We have formed 15 cases of grammatical category and tested the system translation corresponding to them. The existing system for Sanskrit–Hindi translation was also compared.

1.6 Paper organization

The remainder of the paper is organised as follows: Sect. 2 presents the background of both NMT and MTS for processing the Sanskrit language. An elaborate system description of our proposed system described in Sect. 3 followed by the experimental design of the system in Sects. 4 and 5 discusses detailed result analysis, and finally, Sect. 6 concludes the paper along with the future scope.

2 Background

Neural network (NN) has been in research since 1980's-1990's [19] till then these methods for machine translation were unexplored. Later in the year neural network (NN) and finite state machine (FSM) were trained for English–Spanish, and experimental results show NN performing better than FSM on the same size of training data [20]. Another model "Recurrent hetro-associative memories" capable of learning simple translation task proposed by building a state-space representation of the input string to obtain the output string [21]. These approaches turned out to be similar to current NN approaches, but it lacked the computational complexity to process a large amount of data and decrease the learning time or size of NN. The computational complexity involved deserted these ideas for decades. Meanwhile, data-intensive approaches, i.e. statistical machine translation (SMT) came into dominance and turned out to be useful for applications of information extraction to professional translators. In SMT, the analysis is performed on data and calculated by the maximum probability (Pr) of target (t) sentence for the given source sentence (s) in Eqs. (1–2)

$$Pr(s|t) = \frac{Pr(s)Pr(t|s)}{P(t)} \tag{1}$$

$$e = argmax_e Pr(s)Pr(t|s) \tag{2}$$

It builds a language model from monolingual text and translation model from the parallel corpus. The decoding algorithm uses both these translation model and language model for predicting the target language sentence. It follows two objectives, i.e. adequacy and fluency. Adequacy refers to target translation that must carry the same meaning as the source. It requires a translation model that assigns a score to each sentence. Fluency refers to the fluent translation which requires language model that assigns a score to each sentence. These statistical models do not consider context, as a single word may have a different meaning in a different context. The statistical, probabilistic language model improved with context, perplexity and feature vectors by proposing a neural probabilistic language model [22]. Later, an explorer in this field using neural language models in SMT showed large improvements [23], but the computational drawback was still there for many research groups as it required GPU for training the data. There were few more experiments carried out by integrating neural components into SMT such as statistical language model based on neural network [24], neural network joint model, combining into the decoder. The later model showed empirically good results [25]. Other integrations of language model include continuous space language model on a GPU for SMT [26], factored recurrent neural network language model in TED lecture transcription [27]. Also used in source side pre-ordering for faster and better SMT [28]. Neural models used for improvised reordering of SMT systems [29, 30].

2.1 Neural machine translation

Neural machine translation (NMT) involves the incorporation of context and better calculation of probability for similar words. It is a machine learning technique which takes inputs as a source sentence and predicts output as a target sentence. It has been outperforming the traditional methods for MT such as rule-based and statistical-based. Early efforts for pure NMT include convolution neural network which uses a continuous-space representation of words, phrase and sentence [31]. Deep long short-term memory (LSTM) was used for encoding and decoding of input sequence into a fixed dimension vector and vice-versa. These models performed quite well on short sentences but were not able to produce a translation for long sentences. Later, neural models added with attention method was proposed by selectively choosing parts of source sentence during translation. [32]. These models produced quite promising results but embedding them into fixed vectors restricted the user. Therefore, [5] proposed embedding of sentences into variable-length vectors. This work yields promising results and is being widely used to train NMT systems.

2.2 Machine translation systems for processing Sanskrit language

In this paper, Sanskrit–Hindi hybrid machine translation (SHH-MTS) is proposed, where hybrid mechanism (a combination of linguistic features extracted from the rule-based and recurrent neural network) is applied for translating Sanskrit–Hindi language. Sanskrit is one of the oldest Indo-European languages [33]. In Uttarakhand, Sanskrit is the official language [34], and it is also known as the donor of the other [35]. Hinduism, Buddhism and Jainism used this language in their holy books, Vedas and other philosophical texts [36]. As currently, the Sanskrit-based systems are using rule-based approaches which are not easily extendable and are time-consuming. The challenging task in the translation field for Sanskrit is its morphologically rich features that consist of various modules, resources and tools. Thus, it is a complex system. Sanskrit has a richness of scientific literature with in-depth analysis for cognitive knowledge description. Sanskrit has extendability to other languages. The phonetic basis of the Sanskrit language is useful for speech analysis as well [37]. Research in the field of Sanskrit MT is in the early stages. Some of the developed systems to provide computation and translation of Sanskrit language include Sanskrit Translator (2009) which [38] uses an example-based technique to translate from English to Sanskrit using different modules. Merging rule-based approach with artificial neural network (2010) by [39]for translating English to the Sanskrit language was developed. English to Sanskrit MT and Synthesizer System (2010) by [40] using a dictionary-based approach which includes speech synthesiser in translation process developed. English to Sanskrit Machine Translator system was formed (2010) by [41] using rule-based approach and modules such as the lexical parser, semantic Mapper, translator and composer. [42] formed a complete

English to Sanskrit speech translation system. It was later enhanced in (2012) by [43] merging rule-based approach. E-Trans system (2012) by [44] suggests using a rule-based approach from English to Sanskrit using a synchronous context-free grammar. The results are quite promising for small and large sentences. A system for translating Sanskrit to English, i.e. TranSish (2014) by [45] uses a rule-based approach but only for present tense. This system extended for other tense sentences. These research works were useful for sharing extensive knowledge of Sanskrit with traditional translation approaches. Recent approaches to Sanskrit language translation were performed using a statistical-based approach such as English to Sanskrit Machine Translator with Ubiquitous Application (2012) by [46]. It uses features like phrase translation probability, inverse phrase translation probability, lexical weighting probability, phrase penalty, language model probability, distance-based distortion model, word penalty to train the system. All of this research work based on processing the Sanskrit language to English or vice-versa. There was an ample amount of work for Sanskrit to Hindi translation including a rule-based system and a statistical-based system. The work for Sanskrit to Hindi MT system include [47] in 2015 for translating children stories (e-learning content, multimedia for Kids) using a rule-based approach. Extension to yoga and Ayurveda domain. Though this is not yet available as it is under progress. Another system of translation and computational tools of Sanskrit, i.e. Samsaadhani (2009) by [48–50] for translating Sanskrit to Hindi also uses rule-based linguistic approach. Recent attempts using a statistical approach for translating Sanskrit to Hindi have proposed. The system trained on MT-Hub platform [51] and Moses [8] resulting in significant improvement in Sanskrit language computations.

In this paper, we have proposed Sanskrit–Hindi hybrid machine translation system (SHH-MTS) handling these limitations of the existing MTS developed for the Sanskrit Language.

3 System description

The section consists of data pre-processing, the addition of linguistic features, encoder for embedding source input sentence into vectors, a decoder for converting the trained vectors into target sentence and a web-interface to make the translation available as a service for users.

3.1 Data pre-processing

The corpus data are formed suitable for applying the NMT model. It is a twofold process, i.e. clean text and split text. The cleaning of the text involves dividing the document into sentences. Then, removing all non-printable characters, punctuation markers, normalise Unicode characters to ASCII value, changing uppercase letters to lowercase and removing any remaining tokens that are not alphabetic or numeric. We performed these operations on each phrase for each pair of dataset loaded. Secondly, splitting operation is performed on cleaned data. The dataset contains different length sentence pairs; therefore, different computation graphs were drawn. We then sort the sentences in a batch based on sentence pair by length and break similar-length sentences into mini-batches. Therefore, we recurrently, shuffle the training corpus and break the corpus into maxi-batches and again perform splitting in mini-batches. It is processed further by applying gradient for parameter update.

3.2 Rule-based machine translation system; extraction of linguistic features

The proposed work has a pipeline architecture, which takes input from its previous phase and performs computations and passes the output to the next step. Different tools are divided into different modules or phases developed under Sanskrit Consortium Project funded by MIT using Anusaaraka [52]. There are 10 modules in the rule-based pipeline architecture of Sanskrit to Hindi translation [49]. All of the modules provide an individual output as linguistic features to Neural based encoder–decoder to train the system more efficiently. We used Anusaaraka engine for translation [52].

- Pre-processing of user input: It takes input from the user, cleanses, normalise the text, converts the input notations into WX notation, call and invokes MT system which performs computation and shows the output result.
- Tokenizer: Tokenizer receives a flow of character, and that character breaks into individual words called as tokens (words, punctuation, markers). It removes the formatting information and adds a sentence tag. Here, the term morphology is used for linguistics. It refers to a study of words, their internal process and their word meaning. The model has a stream of words; those words tokenized first, and then morphology gives meaning to those words [53].
- Sandhi-Splitter: It is invoked when the input text contains Sanskrit sandhi words. It splits these words as well as compound words [50, 54, 55].
- Morphological Analyzer: It splits words into their roots and grammatical suffixes. There are different units, and each unit provides meaning as well as grammatical function. It also provides inflectional analysis, prunes the answer, uses local morph analysis to handle

unrecognised words and produce a derivational analysis of the derived roots. [56–58].

- Parsing: Parser is used as a compiler or interpreter that breaks data into smaller units for easy translation of one language to another. Parsers take input from the sequence of words or tokens. These inputs translated in the form of a parse tree. It converts the source language into the target language in the form of a tree with labels of noun, verbs and their associated attributes. Morph analysis according to context along with karaka analysis is performed. According to computational Paninian grammar, it identifies and names the relation between the verb and its participants. [59–63].
- Shallow Parsing: If the parser fails on any input, it does minimum parsing of the sentence and produces pruned morph analysis to next layer. [50, 59, 59, 64].
- Word Sense Disambiguation (WSD): The modules perform word sense disambiguation of input sentence words roots, vibhakti and lakara. It identifies a correct sense of a Sanskrit word [16].
- Parts of Speech Tag (POS): It adds parts of speech tags to each word such as adjective, verb or noun. tags [65, 66].
- Chunker: This phase performs a minimum grouping of words in a sentence such as a noun phrase, verb phrase, adjective phrase. The rule base allocates an appropriate chunk tag to it. [67].
- Hindi Lexical Transfer: The Sanskrit Lexicon is transferred to Hindi identifying root words using the dictionary. The output formatted according to the Hindi Generator, which generates the output in Hindi Language corresponding to the Sanskrit language. This module also performs transliteration in case of translation fails [68].
- Hindi Generator: This phase involves sentence level generator which performs agreement between a noun, adjective and verb in the target language. Addition of vibhakti markers ’ne’ and dropping ’ko’ at required positions. Final generation involves root words and their associated grammatical features, corresponding suffixes and concatenates them by generating words into a sentence [16].

Hence, a translation of each Sanskrit word to its corresponding Hindi word is performed using linguistic rules and tools. Further, these data are passed to consequent phase. The data passed to next phase is converted into Comma-separated values (CSV) format suitable for training, model development and fitting the values for neural-based encoder–decoder architecture for predicting translation of Sanskrit word to Hindi word. These linguistic tools output is embedded as features for input encoding of source sentence.

3.3 RNN encoder–decoder with attention mechanism embedding extracted features

The proposed work shown in Fig. 1 is novel and can be applied to any low-resource language having less amount of parallel data. We have used improved NMT with attention mechanism proposed by [32] along with GRU cells for computation [5]. The implementation consists of stacked Bi-directional RNN layers for both encoder and decoder with attention mechanism. The NMT proposed first encodes the source sentence $W_s = w_{s1}, \ldots, w_{sn}$ into variable sequence of context vectors $S = h_1, h_2, h_3 \ldots h_n$. The decoder decodes the context vector S_i and generates target sentence one word at a time $W_h = w_{h1}, w_{h2}, w_{h3} \ldots w_n$ by maximizing the target word probability given previous generated word h_{i-1}, hidden state decoder d_{s_i} and context vector s_i $P(h_i|d_{s_i}, h_{i-1}, s_i)$. The detailed explanation of the encoder, attention mechanism and decoder have been mentioned in further sub-sections. Entire implementation has also been depicted in Algorithm 1 in the form of pseudocode.

3.3.1 Encoder

Given a source sentence in Sanskrit Language $W_s = w_{s_1}, w_{s_2}, w_{s_3} .. w_{s_z}, s_i \in \mathbb{R}^{Ks}$ and target sentence in Hindi Language $W_h = w_{h_1}, w_{h_2}, w_{h_3}, \ldots w_{h_l}, h_x \in \mathbb{R}^{kh}$ from the parallel corpus. Here, K_s and k_h are vocabulary sizes and z and x are length of input sentence and output sentence. The model first tokenizes W_s to form input representation where probability of a sequence of $T(w_{s_1}, w_{s_2}, \ldots w_{s_n})$ words is denoted as $P_1(w_{s_1} \ldots\ldots w_{s_t})$. It is usually conditioned on a window words rather than all previous words. Since the number of words coming before a previous word w1 varies depending on locations with input document in Eq. (3).

$$\begin{aligned} P_1(w_{s_1}, w_{s_2} \ldots, w_{s_{t_z}}) &= \prod_{i=1}^{t} P(w_{s_1} \ldots w_{s_{i-1}}) \\ &\approx \prod_{i=1}^{t} P(w_{s_i}|w_{s_1} \ldots w_{s_{z-1}}) \ldots w_{s_{z-1}} \end{aligned} \tag{3}$$

As we cannot directly apply neural network to text data. Text is converted into numbers or integer-tokens which is further converted into vectors by embedding layers. By setting the maximum number of words in the vocabulary, we use the tokenizer for source and target language. The data-set once converted into sequence of integers-tokens are then padded and truncated and saved as numpy arrays. The encoder uses this output of tokenizer as arrays and computes embedded vectors $(\mathbf{w_{s_1}}, \mathbf{w_{s_2}}, \mathbf{w_{s_3}} \ldots \mathbf{w_{s_z}})$ for hidden layers computation. These vectors have value between

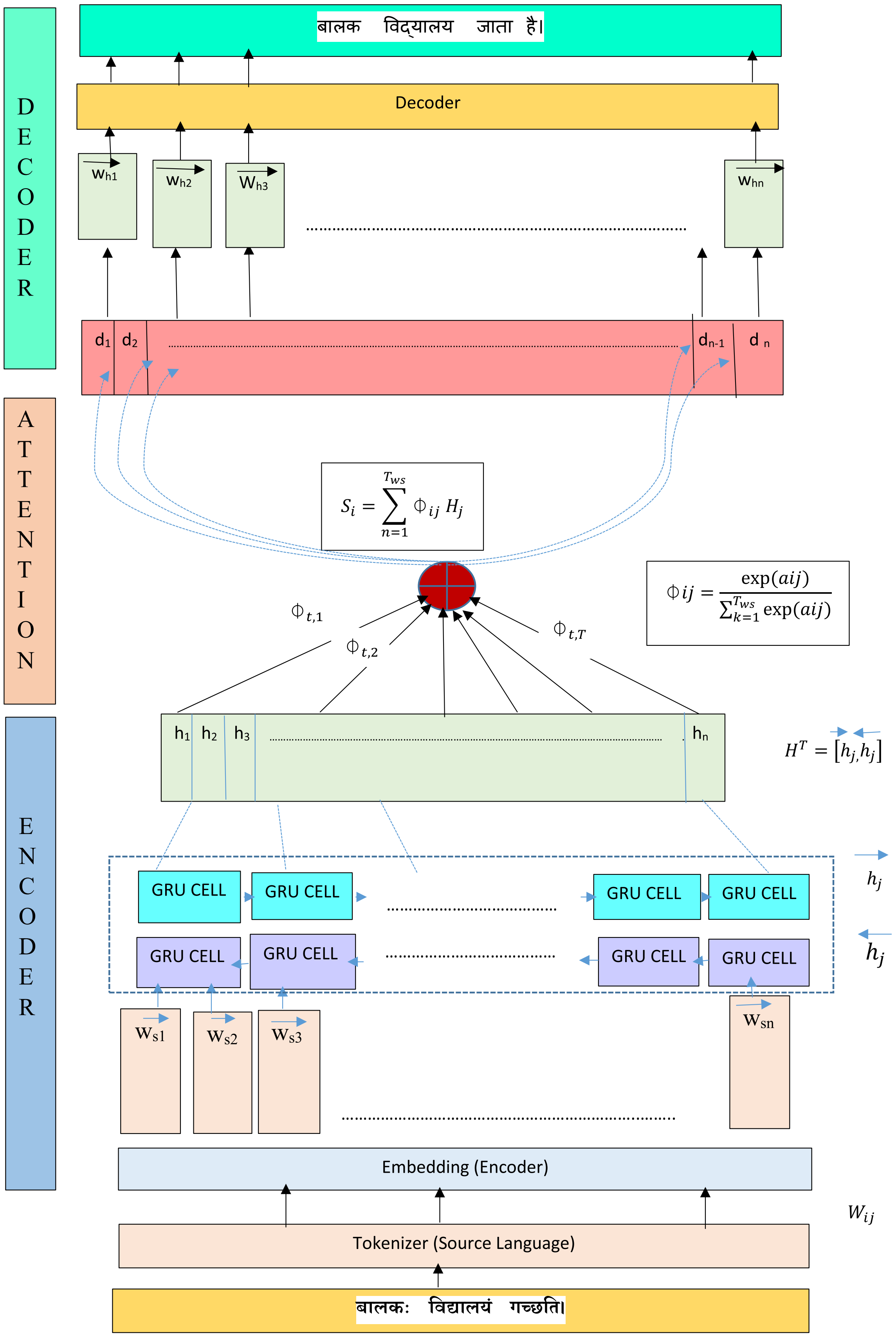


Fig. 1 Deep neural network architecture

]

1 to − 1 having similar semantic meaning words mapped to similar vectors.

Forward RNN reads input sentence from starting to end $\mathbf{f}$ and compute the hidden states $(\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3} \ldots \mathbf{h}_{\tau_j})$. The backward RNN computes the hidden states $(\boldsymbol{h_1}, \boldsymbol{h_2} \ldots, \boldsymbol{h}_{\tau_j})$ by reading the sentence in reverse order. These hidden states, i.e. forward and backward are combined to form an annotation vector $H_i = [\mathbf{h_j}^T; \boldsymbol{h_j}^T]$. The traditional encoder consisting of an embedding lookup of each input word s_z and mapping steps through hidden states $\mathbf{h}_\tau$and $\boldsymbol{h}_\tau$in Eq. (4).

$$\mathbf{H_j} = f(\mathbf{h_i} - \mathbf{1}, \bar{E}W_{s_n}) \tag{4}$$

The encoder computations are deeply stacked in following manner as in Eqs. (5–6). For first layer,

$$h_{t,1} = f_1(h_{t-1}, 1, w_s t) \tag{5}$$

For $i > 1$

$$h_{t,i} = f_{h_{t-1,i}, h_{t,i-1}} \tag{6}$$

where, $h_{t-1,i}$:previous time stamp value and $h_{t,i-1}$:previous layer in sequence value. The context vector s_i contains the summary of the input sentence computed by processing backwards and forward RNN's. We have used gated recurrent units for the function of encoder and decoder [69] as well. We have designed the gated recurrent units (GRU) in a manner to have more persistent memory thereby making it easier for RNN to capture long-term dependency. Mathematically, GRU has previous state h_{t-1} and input w_{s_t} to generate the next hidden state h_t

For update gate in Eq. (7), reset in Eq. (8), new memory in Eq. (9) and hidden State, for all I words of a sentence in Eq. (10).

$$\mathbf{up_i} = \sigma(\mathbf{W_{up}}\bar{E}_{s_i} + \mathbf{O_{up}}\mathbf{h_{i-1}}) \tag{7}$$

$$\mathbf{res_i} = \sigma(\mathbf{W_{res}}\bar{E}s_i + \mathbf{O_{res}}\mathbf{h_{i-1}}) \tag{8}$$

$$\mathbf{h}_i = tanh(\mathbf{W}\bar{E}s_i + \mathbf{O}[\mathbf{res_i} \odot \mathbf{h_{i-1}}] \tag{9}$$

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

Here, d is the dimensionality of word embedding and u is number of hidden units $\bar{E} \in \mathbb{R}^{d \times ks}$.

$$\mathbf{W}, \mathbf{W_{up}}, \mathbf{W_{res}} \in \mathbb{R}^{u \times d}$$
$$\mathbf{O}, \mathbf{O_{up}}, \mathbf{O_{res}} \in \mathbb{R}^{u \times u}$$

σ is logistic sigmoid function. The backward states of bidirectional recurrent neural network are computed similarly for update gate in Eq. (11), reset gate in Eq. (12), new memory in Eq. (13) and hidden State, for all i words of a sentence in Eq. (14).

$$\boldsymbol{up_i} = \sigma(\boldsymbol{W_{up}}\bar{E}_{s_i} + \boldsymbol{O_{up}}\boldsymbol{h_{i-1}}) \tag{11}$$

$$\boldsymbol{res_i} = \sigma(\boldsymbol{W_{res}}\bar{E}s_i + \boldsymbol{O_{res}}\boldsymbol{h_{i-1}}) \tag{12}$$

$$\boldsymbol{h}_i = tanh(\boldsymbol{W}\bar{E}s_i + \boldsymbol{O}[\boldsymbol{res_i} \odot \boldsymbol{h} - \boldsymbol{i} - \boldsymbol{1}] \tag{13}$$

$$h_i = (1 - \boldsymbol{up_i}) \odot \boldsymbol{h_{i-1}} + up_i \odot \boldsymbol{h_i}) \tag{14}$$

The forward and backward states are combined as $h_i = [\mathbf{h_i} + \boldsymbol{h_i}]$.

3.3.2 Addition of linguistic features to encoder

Our framework integrates linguistic features [70] extracted from the pipeline architecture of rule-based to train recurrent neural network. Each feature has a distinct vector word embedding s_{zy}. Combining all these word vectors form a feature embedding matrix $E \in \mathbb{R}^{dy \times ky}$ with d_k as a summation of the dimension of all feature embedding and ky as vocabulary size of K^{th} feature. These embeddings are later concatenated with total embedding size as the length matches. The input embedded sentence vectors are multiplied with these extracted linguistic features. All other functionality and parameters of the model remain the same, only this change in the encoder is performed as in Eq. (15) which result in exceptional improvement in the fluency of output.

$$h_l = tanh\left(\mathbf{W}\prod_{y}^{F}\bar{E}_y s_{zy} + \mathbf{O}\mathbf{h}_{l-1}\right) \tag{15}$$

3.3.3 Attention mechanism

The attention layer bridges the gap of encoder that produces a sequence of word representation $h_j = (\mathbf{h_i}, \boldsymbol{h_i})$ and decoder expecting S_i context vector at each time step t_i. It calculates association between input word W_s to produce the next output wordw_h by calculating the impact of word representation $(\mathbf{h_i}, \boldsymbol{h_i})$. The context vector mathematically calculated as a weighted sum of annotations h_i. For this, we first need to calculate alignment model a_{ij} as in Eqs. (16–18), the score of output position around i to input position around j. It takes hidden state d_{i-1} and h_j as jth annotation of input Sanskrit sentence.

$$a_{ij} = J_a^\tau tanh(W_a d_{i-1} + O_a h_j) \tag{16}$$

$$\alpha_{ij} = \frac{exp(a_{ij})}{\sum_{y=1}^{Ts} exp(a_{iy})} \tag{17}$$

$$S_i = \sum_{j=1}^{ts} \alpha_{ij} h_j \tag{18}$$

Here, S is feed-forward neural network.

$W_a \in \mathbb{R}^{n'_1}, O_a \in \mathbb{R}^{n' \times n}, J_a \in \mathbb{R}_{n' \times 2n}$ are weight matrices. The computed scalar attention value is normalized using softmax activation function, so all input words s adds up to 1.

3.3.4 Decoder

The decoder at each time step t takes sequence of previous hidden state d_{i-1} , some representation of input context s_i and embedding of previous word output $E_{h_{i-1}}$ to output a new word prediction w_{h_i} and new output decoder hidden state. The initial hidden state is computed in Eq. (19).

$$d_0 = f(W_d \mathbf{h_1}) \tag{19}$$

where $W_d \in \mathbb{R}^{d\times d}$. The hidden state d_i is computed given annotation from encoder in Eq. (20) and for update in Eq. (21),Reset in Eq. (22).

$$d_i = tan\bar{h}(WEh_{y_{i-1}}) + O[res_i + d_{i-1}] + Ss_i) \tag{20}$$

$$up_i = \sigma(W_{up}E_{h_i-1} + O_{up}d_{i-1}S_{up}s_i) \tag{21}$$

$$res_i = \sigma(W_{res}Eh_{i-1} + O_{res}h_{i-1} + S_{res}S_i) \tag{22}$$

Where, E is embedded matrix of word for target language with u as number of hidden units and d is word embedding dimension. $W, W_{up}, W_{res} \in \mathbb{R}^{u\times d}$, $0, O_{up}, O_{res} \in \mathbb{R}^{d\times 2d}$ are weight matrices. The vector for prediction p_i for a output word is based on decoder hidden state d_{i-1}, input context s_i and embedding of previous output word h_{i-1} as in Eq. (23).

$$p_i = softmax(O_{ot_{d_i-1}} + V_{ot}E_{h_i-1} + S0s_i) \tag{23}$$

Where, $V_{ot} \in \mathbb{R}^{2l\times d}, O_{ot}\mathbb{R}^{2l\times u}, C_o\mathbb{R} \in 2l \times 2u$ are output word embedding matrices. On, $E_{W_{h_i-1}}$ condition is repeated as we use d_{i-1} rather than d_i as it fragments the encoder state progress from d_{i-1} to d_i for prediction of output word p_i in Eq. (24). Here, token for output word w_{h_i} is the highest value in the vector.

$$p_i = [\max p_i, 2\bar{j} - 1, i, \bar{2}j]^{\tau}_{j=1...,l}. \tag{24}$$

Even training is performed accordingly as the network being aware of correct output w_{h_i} assigned larger probability value as in Eq. (25).

$$prob(h_i|d_{i-1}, s_i) \propto (h^{\tau}W_o pi) \tag{25}$$

Activation function softmax is used to convert the raw vector into a probability distribution having a sum of values as 1. We have also used here Relu [71] that combines input to yield the next hidden state. To predict the target variable more efficiently activation function is passed to the model. It also works as a rectifier. The model follows a deep output as suggested by [72]and has been visualized in the table: 1. A web interface as shown in Fig. 2 is designed for SHH-MTS proposed for delivering it as a service.

Algorithm 1 Recurrent Neural Network Implementation for SHH-MTS

```
 1: Get the encoded data (data).          /* obtain data in matrix form and then convert using countvectorize in vectors. */
 2: X = data[:,x:y] and Y = data[:,z]          /* Split the data into training, development and testing form,where x,y,z are number of
    rows and columns we used to train the model */
 3: Create the sequential model using Keras.
 4: Add n number of hidden layers with different activations i.e.
 5: for i in range(n) do
 6:     model← add(Dense(i,input_dim=723,init=uniform,activation=relu))
 7:     model←add(Dense(iy,init=uniform,activation=relu))
 8:     model←add(Dense(ix,init=uniform,activation=sigmoid))
 9: end for
10: Train and test the model
11: Add the n number of Bidirectional layers using relu activation function
12: for i in range(n) do
13:     x=Bidirectional(SimpleRNN(32*2**i,return_sequences=True))(x)
14:     x= Activation(relu)(x)
15:     Add the n number of Bidirectional layers and using gru activation function
16: end for
17: for i in range(n) do
18:     gru_1=Bidirectional(SimpleRNN(i,return_sequences=True, init='HE_normal', name='gru1'))(x)
19:     gru_1b=Bidirectional(SimpleRNN(i, return_sequences=True, go_backwards=True, init='HE_normal', name=gru1_b))(x)
20:     gru1_merged=merge([gru_1, gru_1b], mode=sum)
21:     Add the n number of Dense layers and using Prelu/Linear activation function.
22: end for
23: for i in range(n) do
24:     x=Dense(i,activation=linear,kernel_initializer=normal)(x)
25:     x=PReLU(alpha_initializer=zeros
26:     alpha_regularizer=None
27:     alpha_constraint=None
28:     shared_axes=None)(x)
29:     Divide and merge the dataset.
30:     Normalize the batch.
31: end for
32: Pass the neurons to dense step.
33: Achieved the model with 99% accuracy with huge speed.
```

Table 1 Visualizing model structure on keras

Layers	Input	Output
Input Layer:1	(None,723,1)	(None,723,1)
Bidirectional (Simple RNN):1	(None,723,1)	(None,723,64)
Activation:1	(None,723,64)	(None,64)
Bidirectional (SimpleRNN):2	(None,723,64)	(None,723,128)
Activation:2	(None,723,64)	(None,723,128)
Bidirectional (SimpleRNN):3	(None,723,128)	(None,723,256)
Activation:3	(None,723,256)	(None,723,256)
Bidirectional (SimpleRNN):4	(None,723,256)	(None,723,512)
Activation:4	(None,723,512)	(None,723,512)
Bidirectional (SimpleRNN):5	(None,723,512)	(None,723,1024)
Activation:5	(None,723,1024)	(None,723,1024)
Bidirectional (SimpleRNN):6	(None,723,1024)	(None,723,2048)
Activation:6	(None,723,2048)	(None,723,2048)
Bidirectional_gru1_a (SimpleRNN):7	(None,723,2048)	(None,723,256)
Bidirectional_gru1_b (SimpleRNN):8	(None,723,2048)	(None,723,256)
Merge:1	[(None,723,256), (None,723,256)]	(None,723,256)
Bidirectional_gru2_a (SimpleRNN):9	(None,723,256)	(None723,256)
Bidirectional_gru2_b (SimpleRNN):10	(None,723,256)	(None,723,256)
Merge:2	[(None,723,256), (None,723,256)]	(None,723,256)
Bidirectional (SimpleRNN):11	(None,723,256)	(None,1024)
Activation:7	(None,1024)	(None,1024)
Dense:1	(None,1024)	(None,723)
PReLU:1	(None,723)	(None,723)

Fig. 2 Web interface

4 Experimental design

The section contains details of corpora, model Size, parameter initialisation and training performed. The experiment performed on different epochs, beams sizes and different sentence length which lead to change in the update, BLEU score, training probability and development probability shown in Figs. 4 and 5.

4.1 Corpora

The system architecture designed required parallel as well as monolingual corpora. Firstly, the existing parallel corpora available for Sanskrit–Hindi was gathered as depicted in Table 2, [73]. These data were gathered from different domains such as news, healthcare, tourism, literature, Wikipedia, judicial and general domain. Even a parallel corpus of Bhagwad-Geeta was manually created containing 700 slokas along with Hindi conversion. Further, 50,000

Table 2 Parallel and monolingual dataset from different domains

Domain	Parallel corpus	Monolingual corpus
News	25,000	202,269
Health care	NA	5,000
Tourism	NA	15,395
Literature	28,760	50,000
Wikipedia	NA	259,305
Judicial domain	NA	152,776
General domain	49,000	36,000

Table 3 Additional monolingual dataset

# Lines	221528
#Words	2849514
#Characters	38413350
#Total	2.8 Million

parallel Sanskrit–Hindi corpus was made available on request from the project of Indian Languages Corpora Initiative (ILCI) [51]. The entire parallel corpora 162,760 parallel sentences, which were used for training the system. It trained the system with less accuracy, and the output was not understandable. So, to overcome this problem we applied synthetic technique as suggested by [74]. This technique was applied on 2.3 million monolingual sentences and Sanskrit prose sentences [75] as shown in Tables 2 and 3. Later on, various others corpus of Sanskrit books were available from "Development of Sanskrit computational tools and Sanskrit–Hindi machine translation System (2008-2012)", funded by DeiTy, Government of India, under the TDIL program, manually developed [76]. To enhance the system, the data were incorporated and entire data were pre-processed and divided into training, development and test set as given in Table: 4.

4.2 Model size

The neural model along with the extracted linguistic features trained on TensorFlow platform consists of various parameters with their respective dimensionality in Table 5. The model structure visualization is depicted in Table: 1 using Keras.

Table 5 Model size

Parameter	Dimensionality
Encoder (forward and backward unit each)	1000
Decoder	1000
x (Word Embedding)	1000
n (GRU hidden state)	1000
d (dimension of word embedding)	620
v (Output maxout hidden layer)	500
n‘ (alignment model hidden units)	1000
W_s	32×400
W_h	32×1
W_{up}	64×400
W_{res}	65×2

4.3 Parameter initialization

The parameters used to train the model are mentioned in Table 5. We have used weight matrices recurrently, as random orthogonal matrices. The bias component has been omitted for forming simpler equations. All the alignment element (V_a) and bias component were initialized as 0. The alignment matrices are initialized, with a $variance = 0.001$ and $mean = 0$ from Gaussian distribution. All other matrices were initialized with the same mean with a variation in $variance = 0.01$.

4.4 Training

In the proposed work, Keras sequential model is used to process the data. The proposed model is processed through a highly configured core GPU with 32 GB of RAM to achieve a high throughput speed approximately 2500 words per second. This speed is not possible for normal systems because in this one epoch will take approximately two hours to run. So, we use a highly configured GPU along with NVIDIA Geforce GTX 1050 and Quadro K6000. Each epoch is a pass over the training set and test set as shown in Fig. 3 and update are performed for each mini-batch parameters. On increasing the number of epochs over the training set, accuracy also increases, whereas in case of test set the accuracy fluctuates. As the trained neural network fits the training data, accuracy is increasing. The training accuracy is effected by setting the learning rate,

Table 4 Dataset division into training, development and testing

Dataset	Sentences	Words		Vocabulary	
		Source	Target	Source	Target
Training	145,34,215	131,575,835	123,425,654	355.465	124.278
Development	192679	172,799	122,645	NA	NA
Test	12698	77,322	26,273	NA	NA

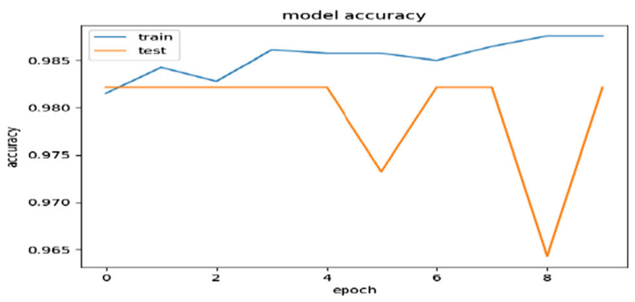


Fig. 3 Epochs for training and test set

regularization, etc., for the model. As the network fits only the training data so it modifies the weights according to data. Whereas in the case of test data, which is never used for weight update in gradient descent algorithm. Despite these test data are not exactly the training data. Though test accuracy should globally improve over the iterations if the network is learning, but it is not bound to improve at each iteration exactly. The graph was a generated output of the developed model on Keras platform for modelling overfitting of training data on model. The training and development probability is the average conditional log-probability of the sentence to be in either of the sets.

Vanilla stochastic gradient descent (SGD) algorithm has been used with automatically updating learning rate using Adadelta [77] (parameters$\rho = 0.95$ and $\epsilon = 10^{-6}$). Adam optimizer is employed for stochastic optimization [78]. Normalization is performed [79] for each of the mini-batch (distributed data set). As the distribution of input layer changes due to the change in the parameter of the previous layer during the training, it makes training difficult to perform. Normalization conducted to reduce the internal covariate shift, and it increases the learning rate by reducing the initialization process. It even reduces the need for dropout, by acting as a regulariser. We have taken a minibatch of 64 sentences which was normalized when exceeded the threshold value of 1. Each update took time equivalent to its longest sentence. To minimize the time, we manually sorted and shuffled sentences by retrieving 1500 pair sentence after every 20th update.

5 Result analysis and discussion

The performance of this proposed SHH-MTS evaluated using automated metrics and human evaluators. The sentence length of the corpora affecting the update of the model. Figure 4a depicts, i.e. on increase in the sentence length in the training corpus the number of updates of weights in the model training drastically increases over a point (till 20 words length) and then decreases after the sentence length exceeds more than 20 words. Figure 4b denotes the effect of sentence length on number of iterations performed on training set, i.e. epochs. It can be clearly deduced from the graph that the number of epochs is decreased after the point (i.e. 20 sentence length). Figure 4c denotes the effect of sentence length on time for building the model. As depicted in the figure, the time fluctuates drastically. For sentence length 10–20, the model training time remains same, whereas for 20–30 sentence length it increases rapidly. In conclusion, the sentence length upto 20 words limits the update, epochs and time. If the sentence length in corpus exceeds this limit, there is a drop seen as the graphs plotted. Figure 5a depicts the BLEU score varies with the beam size. A beam search was used during the inference to find the most likely sequence of words for each translation. The beam problem in neural machine translation exists for relatively small beam sizes – especially when compared to traditional beam sizes in statistical machine translation systems. We can see from the figure beam size (1–4) have a constant change in the BLEU score, whereas from 5-10 the BLEU score was enhanced. In case if there is a large increase in beam size, it drops the BLEU score. So, here in our experiment training, we have limited the beam size by normalizing the length of

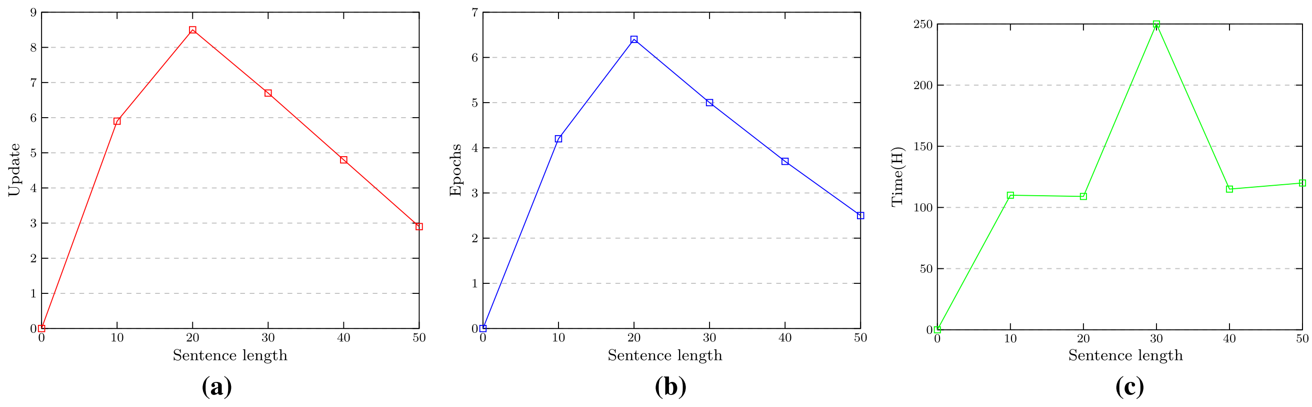


Fig. 4 Sentence length affecting. **a** Updates. **b** Epochs and **c** Time

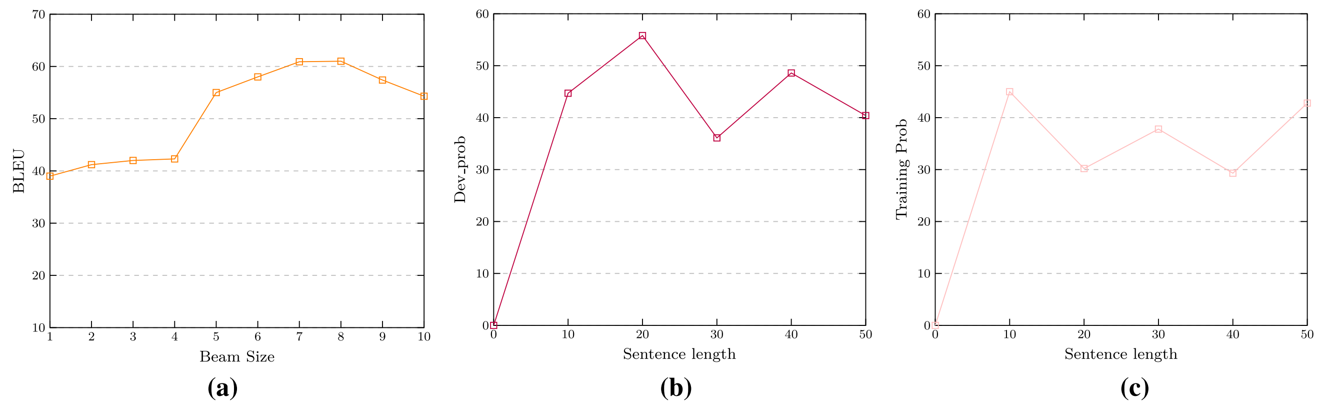


Fig. 5 **a** BLEU varies with Beam sizes. **b** Development probability varies depending on Sentence length **c** Training probability varies depending on Sentence length

sentences. Figure 5b depicts the sentence length effect on the development probability. As shown in the plot, the development probability increases only till 20 sentence length and decreases thereafter. Therefore, either splitting the longer sentences or normalizing the sentences of length greater than 20 would be an ideal strategy. Figure 5c models the sentence length effect on the training probability. As depicted, the training probability decreases with the time. So in order to increase the training probability, shorter sentences would be better modelled . The different automated metrics used such as BLEU, Word Error Rate (WER), F-measure and Meteor have been covered in detail in further sections.

5.1 Automatic error analysis

- BiLingual Evaluation Understudy (BLEU) is an important metric used for calculating the accuracy of translated sentences as compared to the human-generated reference translations as in Eq. (26). It provides accurate results for longer sentences but fails for shorter sentences [80]. The BLEU score is evaluated at each iteration of performance enhancement as depicted in Table 6. Firstly, a simple sequential model is built which gives an accuracy of 10.23%. To improve the accuracy of the model, we applied Keras model with the bidirectional layer that significantly improved the accuracy to 29.12%. The result produced a readable translation but required significant improvement. It was enhanced with gated recurrent unit (GRU) cells which perform better computation. To attain the output activation function is applied in the neural model, but with the implementation of different activation function better level of accuracy is achieved, i.e. 56.78% for our proposed system. The accuracy attained at a stagnant level with all the significant experimentation but auto-tuning our model gave a bit of enhancement to the proposed model and gave an accuracy of 61.02% as a final accuracy.

$$BLEU = min\left(1, \frac{output\_length}{Reference\_length}\right)\left(\prod_{i=1}^{4} precision_i\right) \tag{26}$$

It will compute precision w.r.t human-generated translation without taking into account any grammatical corrections/errors. As we experimented with different models to enhance the performance of MTS. The BLEU score computed for all the three models, i.e. RBMT, neural and hybrid depicted in Fig. 6a. As we build our model in three phases. Firstly, we build the rule-based model as explained in Sect. 3.2, secondly neural model

Table 6 BLEU score of different experiment performed

Models	BLEU (%)
Simple Sequential Model	10.23
Keras Model with bi-directional layers	29.12
Keras Model + bi-directional layer + gru	40.34
Keras Model + bi-directional layer + gru+ Relu and sigmoid activation function	56.78
Keras Model + bi-directional layer + gru+ Relu and sigmoid activation function + auto-tuning	61.02

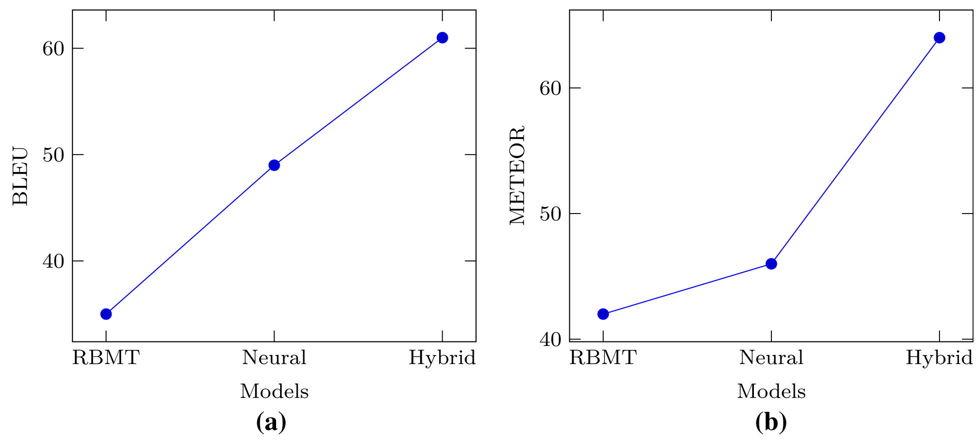


Fig. 6 Different models, i.e. RBMT, neural and hybrid across their **a** BLEU, **b** METEOR

as in Sect. 3.3 and finally hybrid model combining other models which perform better than the rest of the models. The result demonstrates the efficiency of our novel technique applied in our proposed work. The BLEU score obtained for our proposed system varies with the beam size as depicted in Fig. 5a. The development probability varies with the sentence length as shown in Fig. 5b and training probability depends on sentence length.

- Word Error Rate (WER) It is a metric used to calculate the error rate by comparing MT output with the human translated output as in Eq. (27). The lower the WER, the better the model.

$$WER = \frac{substitutions + insertions + deletions}{reference\_length} \tag{27}$$

Here, substitution means replacement of one word with another in a particular sentence. Insertion means the addition of words, and deletion means dropping of words. The WER score computed for all the three models, i.e. RBMT, neural and hybrid depicted in Fig. 7a. As we build our model in three phases, firstly, we build the rule-based model as explained in Sect. 3.2, secondly neural model as in Sect. 3.3 and finally hybrid model combining other models which perform better than the rest of the models. The result demonstrates the efficiency of our novel technique applied in our proposed work. As hypothesised from figure, hybrid model has a minimum WER score. BLEU and WER are inversely proportional to each other.

- *F*-measure It is a metric for calculation of accuracy and precision of model as in Eq. (28–30). It calculates the quality or exactness of an output. Mathematically, the calculation of F-measure requires precision and recall values also. Therefore,

$$Precision = \frac{Correct}{Output\_Length} \tag{28}$$

$$Recall = \frac{Correct}{reference\_length} \tag{29}$$

F-measure

$$F\text{-}measure = \frac{(Precision\ddot{O}recall)(Precision + Recall)}{2} \tag{30}$$

The F-Measure computed for all the three models, i.e. RBMT, neural and hybrid depicted in Fig. 7a. As we

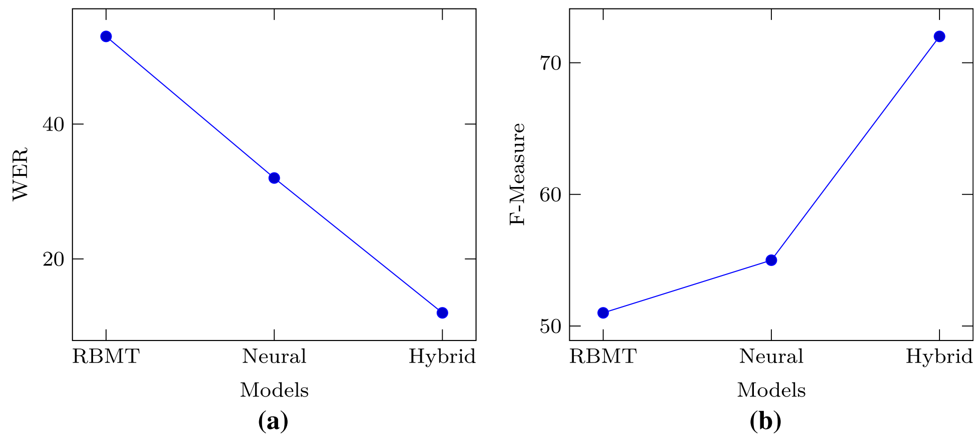
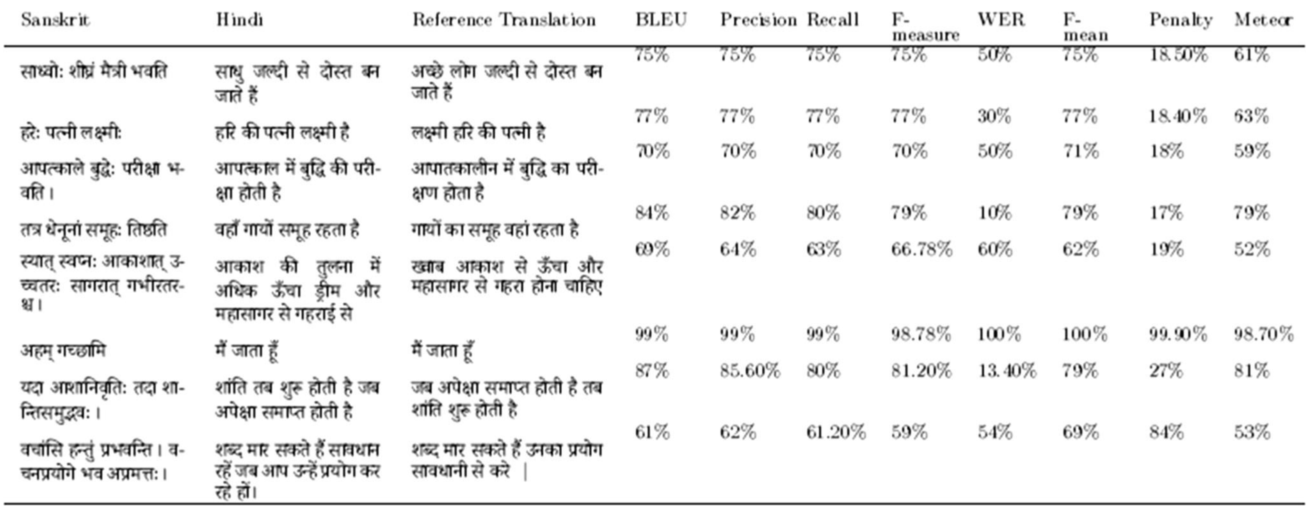


Fig. 7 Different models, i.e. RBMT, neural and hybrid across their **a** Word Error Rate (WER), **b** F_measure

build our model in three phases, firstly we build the rule-based model as explained in Sect. 3.2, secondly neural model as in Sect. 3.3 and finally hybrid model combining other models which perform better than the rest of the models. The result demonstrates the efficiency of our novel technique applied in our proposed work. It is observed that the hybrid model has more F-measure as compared to other existing models.

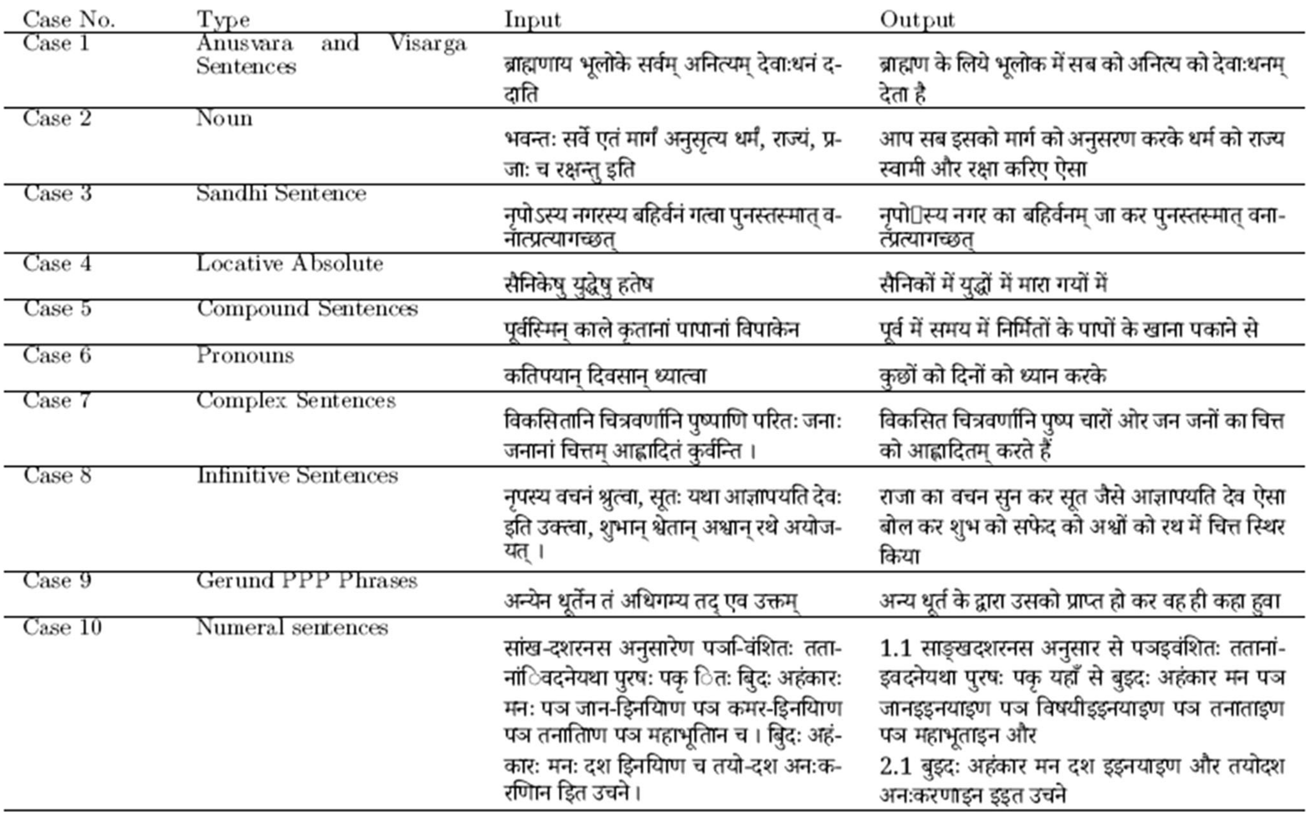
- METEOR It is used to find the correlation between the machines translated output and the human-generated

Table 7 Metric analysis of Sanskrit to Hindi translation



Sanskrit	Hindi	Reference Translation	BLEU	Precision	Recall	F-measure	WER	F-mean	Penalty	Meteor
साध्वो: शीघ्रं मैत्री भवति	साधु जल्दी से दोस्त बन जाते हैं	अच्छे लोग जल्दी से दोस्त बन जाते हैं	75%	75%	75%	75%	50%	75%	18.50%	61%
हरे: पत्नी लक्ष्मी:	हरि की पत्नी लक्ष्मी है	लक्ष्मी हरि की पत्नी है	77%	77%	77%	77%	30%	77%	18.40%	63%
आपत्काले बुद्धे: परीक्षा भवति।	आपत्काल में बुद्धि की परीक्षा होती है	आपातकालीन में बुद्धि का परीक्षण होता है	70%	70%	70%	70%	50%	71%	18%	59%
तत्र धेनूनां समूह: तिष्ठति	वहाँ गायों समूह रहता है	गायों का समूह वहां रहता है	84%	82%	80%	79%	10%	79%	17%	79%
स्यात् स्वप्न: आकाशात् उच्चतर: सागरात् गभीरतरश्च।	आकाश की तुलना में अधिक ऊँचा ड्रीम और महासागर से गहराई से	ख्वाब आकाश से ऊँचा और महासागर से गहरा होना चाहिए	69%	64%	63%	66.78%	60%	62%	19%	52%
अहम् गच्छामि	मैं जाता हूँ	मैं जाता हूँ	99%	99%	99%	98.78%	100%	100%	99.90%	98.70%
यदा आशानिवृति: तदा शान्तिसमुद्भव:।	शांति तब शुरू होती है जब अपेक्षा समाप्त होती है	जब अपेक्षा समाप्त होती है तब शांति शुरू होती है	87%	85.60%	80%	81.20%	13.40%	79%	27%	81%
वचांसि हन्तुं प्रभवन्ति। वचनप्रयोगे भव अप्रमत्त:।	शब्द मार सकते हैं सावधान रहें जब आप उन्हें प्रयोग कर रहे हों।	शब्द मार सकते हैं उनका प्रयोग सावधानी से करे ।	61%	62%	61.20%	59%	54%	69%	84%	53%

Table 8 A case study of linguistic analysis of Sanskrit to Hindi translation (cont.)



Case No.	Type	Input	Output
Case 1	Anusvara and Visarga Sentences	ब्राह्मणाय भूलोके सर्वम् अनित्यम् देवा:थनं ददाति	ब्राह्मण के लिये भूलोक में सब को अनित्य को देवा:थनम् देता है
Case 2	Noun	भवन्त: सर्वे एतं मार्गं अनुसृत्य धर्मं, राज्यं, प्रजा: च रक्षन्तु इति	आप सब इसको मार्ग को अनुसरण करके धर्म को राज्य स्वामी और रक्षा करिए ऐसा
Case 3	Sandhi Sentence	नृपोऽस्य नगरस्य बहिर्वनं गत्वा पुनस्तस्मात् वनात्प्रत्यागच्छत्	नृपो□स्य नगर का बहिर्वनम् जा कर पुनस्तस्मात् वनात्प्रत्यागच्छत्
Case 4	Locative Absolute	सैनिकेषु युद्धेषु हतेष	सैनिकों में युद्धों में मारा गयों में
Case 5	Compound Sentences	पूर्वस्मिन् काले कृतानां पापानां विपाकेन	पूर्व में समय में निर्मितों के पापों के खाना पकाने से
Case 6	Pronouns	कतिपयान् दिवसान् ध्यात्वा	कुछों को दिनों को ध्यान करके
Case 7	Complex Sentences	विकसितानि चित्रवर्णानि पुष्पाणि परित: जना: जनानां चित्तम् आह्लादितं कुर्वन्ति।	विकसित चित्रवर्णानि पुष्प चारों ओर जन जनों का चित्त को आह्लादितम् करते हैं
Case 8	Infinitive Sentences	नृपस्य वचनं श्रुत्वा, सूत: यथा आज्ञापयति देव: इति उक्त्वा, शुभान् श्वेतान् अश्वान् रथे अयोजयत्।	राजा का वचन सुन कर सूत जैसे आज्ञापयति देव ऐसा बोल कर शुभ को सफेद को अश्वों को रथ में चित्त स्थिर किया
Case 9	Gerund PPP Phrases	अन्येन धूर्तेन तं अधिगम्य तद् एव उक्तम्	अन्य धूर्त के द्वारा उसको प्राप्त हो कर वह ही कहा हुवा
Case 10	Numeral sentences	सांख-दशरनस अनुसारेण पञ्-विंशित: ततानांिवदनेयथा पुरष: पकृ ि त: बिुद: अहंकार: मन: पञ जान-इिनयािण पञ कमर-इिनयािण पञ तनाितािण पञ महाभूतािन च। बिुद: अहंकार: मन: दश इिनयािण च तयो-दश अन:करणािन इित उचने।	1.1 साङ्खदशरनस अनुसार से पञइवंशित: ततानांइवदनेयथा पुरष: पकृ यहाँ से बुइद: अहंकार मन पञ जानइइनयाइण पञ विषयीइइनयाइण पञ तनाताइण पञ महाभूताइन और 2.1 बुइद: अहंकार मन दश इइनयाइण और तयोदश अन:करणाइन इइत उचने

Table 9 A case study of linguistic analysis of Sanskrit to Hindi translation

Case No.	Type	Input	Output
Case 11	Anvaya sentences	शीभगवान्उवाच । पाथर, मेनानिावथिान िद-विान नाना-वणर-आकृ तिीन च रपािण शतशः अथ सहसशः पश ।	1.1 शीभगवानुवाच 2.1 पाथर मेनानाइवथाइन इदवाइन नानावणरआकृ तीइन और रपाइण सौ तरह से और सहसशः पश
Case 12	Comparative and Superlative Adjectives	नलंतका एव तेहंसाः वनात्समुतत िवदभर-नगरीम्आगम दमयनाः समीपि नपेतुः ।यदा सखी-गण-वृता दमयनी तान्सवेषांपिकणांशेषान्कनक-अलंकृ तान्हंसान्अपशत्तदा संतुष- मानसा एकं हंसंगहीतुंशीघम्उपचकमे । अननरंतेसवेसवरत वनिवससृपुः एकै कशः तुताः कनाः तान्समुपादवन्। ततः यंहंसंसा दमयनी उपाथावत्सः मानुषी वाचंकृ ता ताम्अबवीत् ।	1.1 नलन्तका ही तेहंसाः वनात्समुतत इवदभरनगरीमागम दमयनाः समीपेइनपेतुः 2.1 जब पसन्द की दमयनी तान्सवेषाम्पिकणांशेषान्कनकअलङ्कृ तान्हंसानपशत्तदा मानसा एक को हंसङ्गहीतुंशीघमुपचकम 3.1 अननरन्तेसवेसवरत वनेइवससृपुः 4.1 एकै कोड़ा तुताः कनाः तान्समुपादवन् 5.1 वहाँ से यंहंसंसा दमयनी उपाथावत्सः मानवी वाचङ्कृ ता तामबवीत्
Case 13	Causative Verbs and Denominative Verbs	पुरष-ऋषभ, एतिह यंसम-सुख-दःखं ु, थीरं, पुरषंन वथयिन, सः अमृतताय कलते ।	पुरषऋषभ एतेइह यंसमसुखदःखम् ऊ ऊ थीर को पुरषंन वथयिन वह अमृतताय कलतायें
Case 14	Meter Passive Stems and Gerundives Passive Impersonal	आचायारः िपतरः पुताः तथा एव च िपतामहाः मातुलाः शशुराः पौताः सालाः तथा संबिननः ।धर्मक्षेत्रे कुरुक्षेत्रे समवेता युयुत्सवः ।मामकाः पाण्डवाश्चैव किमकुर्वत सञ्जय ॥तदा इदंवाकंहषीके शंआह महीपते । “अचुत सेनयोः उभयोः मेरथंसापय ।	1.1 अआचायारः इपतरः पुत वैसे ही और इपतामहाः मामा शशुराः पौताः साल वैसे सम्बिननः 2.1 धर्मक्षेत्र कुरुक्षेत्र एकत्रित लड़ने की इच्छा वाले 3.1 मेरे पाण्डवाश्चैव किमकुर्वत संजय तब इदंवाकंहषीके शंअआह महीपतायें 4.1 अचुत सेनाओं का उभका मेरथंसापय
Case 15	Conjugation sentences	तासांनदीनांसपमीगंगासयंससननम्आिसतंभगीरथम्अनगचत् ।	1.1 तासांनदीनांसपमीगङ्गासयंससननमआइसतम्भगीरथमनगचत्

sample output as in Eq. (31-33). This score is also directly proportional to accuracy. Reducing the effect of F-mean is helpful [81]

$$F\ mean = \frac{10PR}{9 + RP} \tag{31}$$

Here, P is Precision and R is Recall. F-mean, Precision and Recall are based upon the unigrams matches. For longer values penalty required computation. Mathematically,

$$Penalty = 0.5\frac{chunks}{unigrams\_matched}^{3} \tag{32}$$

$$Score = F\ mean \cdot (1 - penalty) \tag{33}$$

The METEOR score computed for all the three models, i.e. RBMT, neural and hybrid depicted in Fig. 6b. As we build our model in three phases, firstly we build the rule-based model as explained in Sect. 3.2, secondly neural model as in Sect. 3.3 and finally hybrid model combining other models which perform better than the rest of the models. The result demonstrates the efficiency of our novel technique applied in our proposed work. In the proposed hybrid model, meteor value is high as compared to other models due to the high correlation between the words of the output sentences. In conclusion, the hybrid model has higher BLEU, F-score, Meteor but low WER as compared to other models.

The performance analysis was also conducted on different category Sanskrit sentences on different automated metrics BLEU, Precision, Recall, F-measure, WER, F-mean, Penalty and Meteor as shown in Table 7. These sentences run on the developed system provides with Hindi output sentences used for performance evaluation along with their reference translation (Table 8).

5.2 Human error analysis

In this work, linguistically errors are identified by performing a case study. It includes 15 different grammatical cases corresponding to which Sanskrit sentences are tested. The output generated by passing different kinds of sentences is displayed in Tables 9 and 10. These results show that the highest error rate in proposed hybrid MTS is

Table 10 Comparison of proposed system with the existing work [8] based on adequacy and fluency

Measure	[8]	Proposed system
Adequacy	91%	96%
Fluency	66.72%	71.26%

encountered in the sentences of category verb 4%, whereas other categories have less than 3% of error rate.

5.3 Comparison of proposed system with existing work

The comparison of the proposed work is performed with the existing work [49, 51] and [8]. The comparison of proposed work with the existing work [49] on automated metrics is depicted in Fig. 8.

The comparative analysis is exhibited in fig. 10b display that the error rate of the Sanskrit–Hindi statistical machine translation (SaHit) [51] is higher compared to the proposed system in Fig. 10a. It was also compared with a recent research work by [8] in Fig. 10c.

The proposed hybrid MTS has 61% accuracy, whereas the accuracy of the existed MTS proposed by [51] is only 27% and by [8] is 57%. The comparison of result was also performed by human evaluators based on grammatical categories, i.e. Sandhi, Compound, Verb, Over Generation, Less Generation, Visarga/Anusarva, Karaka and others. Comparison results show that proposed system provides with a more readable and grammatically correct output than existing work for this domain. The proposed work has been also compared based on adequacy and fluency of the output generated by MTS with existing work as depicted in Table 7. The proposed work for Sanskrit language processing in novel in technique and accuracy achieved in terms of both automated as well as human analysis.

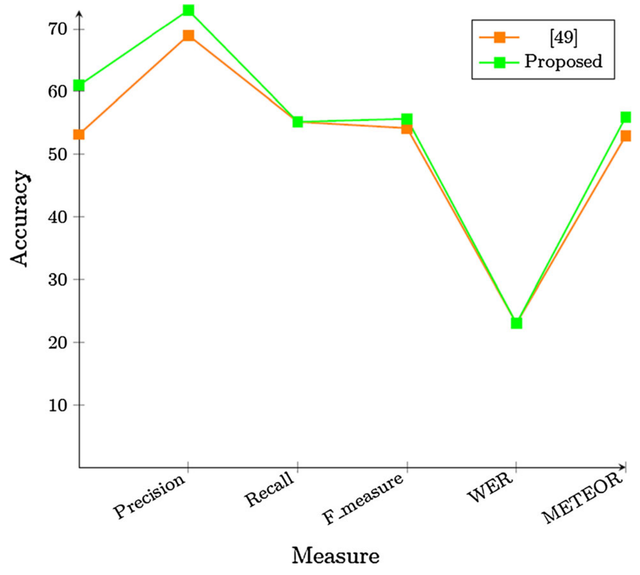


Fig. 8 Comparison of baseline system, i.e. RBMT for Sanskrit–Hindi [49] corresponding to proposed system on various evaluation measures

6 Conclusion and future scope

In our work, three different translation models (rule-based, neural-based and hybrid) have been produced and analysed for Sanskrit to Hindi. MT is a challenging job; hence, the model becomes complex and time-consuming. Earlier work [8, 9, 51] lacks extensibility, generalizability and adaptability which have been overcome by the proposed system. The work developed and presented here is novel and can apply to any low-resource language pair with minimum linguistic knowledge. The work extracts features from the linguistic rule and further passes these features to train the recurrent neural network. Performance evaluation performed on automatic and human measures results in high performance of the hybrid system. It even performs quite well in accuracy, speed and response time. The proposed hybrid model is fast and more efficient than the existing rule-based systems. In non-rule match cases, the rule-based model does not return any output; on the other hand, our proposed model always returns the best solution. The complexity of existing model becomes very high for long sentences, and these are practically infeasible sometimes, but our proposed model is efficient for such cases also. In future, multiple linguistic languages can be taken to convert into the single target language and multi-lingual platform for the same purpose (Figs. 9, 10).

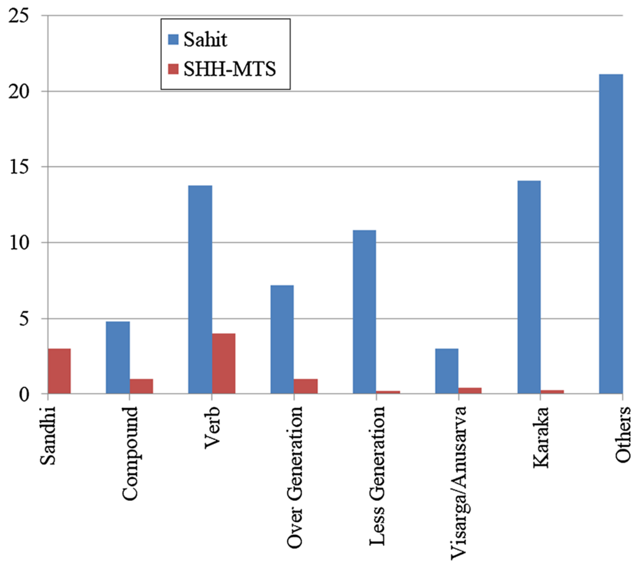


Fig. 9 Human evaluation of proposed system with Sahit

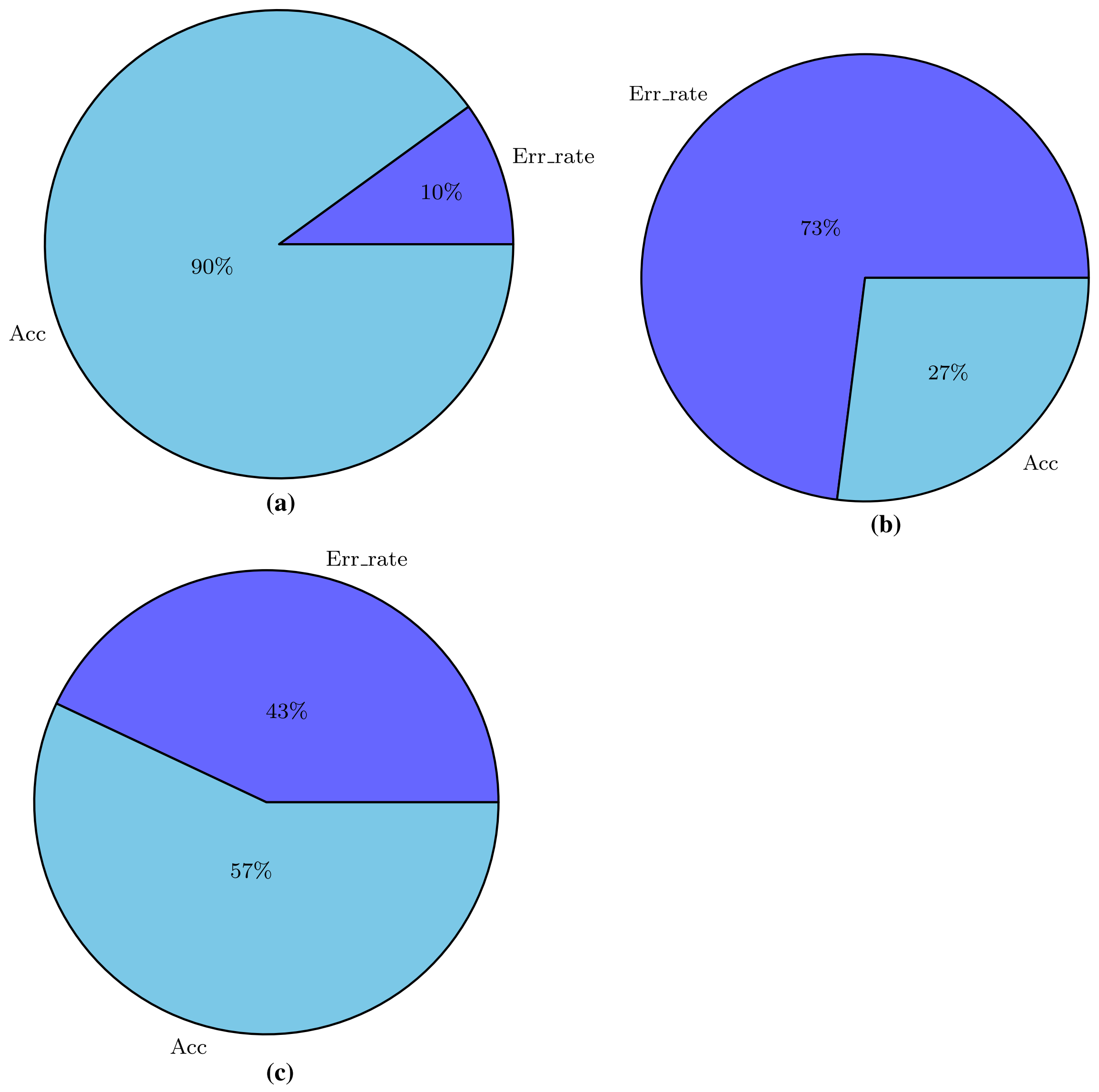


Fig. 10 Comparison of overall error rate and accuracy **a** SHH-MTS, **b** [51], **c** [8]

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Forcada ML, Ginestí-Rosell M, Nordfalk J, O'Regan J, Ortiz-Rojas S, Pérez-Ortiz JA, Sánchez-Martínez F, Ramírez-Sánchez G, Tyers FM (2011) Apertium: a free/open-source platform for rule-based machine translation. Mach Transl 25 (2):127–144
2. Noone G (2003) Machine translation a transfer approach. Computer Science, Linguistics and a Language (CSLL) Department, University of Dublin, Trinity College, Final Rep
3. Dave S, Parikh J, Bhattacharyya P (2001) Interlingua-based english-hindi machine translation and language divergence. Mach Transl 16 (4):251–304
4. Brown PF, Pietra VJD, Pietra SAD, Mercer RL (1993) The mathematics of statistical machine translation: parameter estimation. Comput Linguist 19 (2):263–311
5. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:14090473
6. Somers H (1999) Example-based machine translation. Mach Transl 14 (2):113–157
7. Nirenburg S et al (1989) kbmt-89 project report. Center for Machine Translation, Carnegie Mellon University, Pittsburg p 286
8. Pandey R, Ojha AK, Jha GN (2018) Demo of Sanskrit–hindi smt system. arXiv preprint arXiv:180406716
9. Huet G, Kulkarni A, Scharf P (2009) Sanskrit computational linguistics. Lecture Notes in Computer Science 5402
10. of India G (2018) Census-2018, language-census of india, states and union territories. www.censusindia.gov.in/2011Census/C-16_25062018_NEW.pdf

11. Baindur M (2015) Nature in Indian philosophy and cultural traditions. Springer, Berlin
12. Jha GN, Mishra SK, Chandrashekar R (2005) Developing a sanskrit analysis system for machine translation. In: Proc. National Seminar on Translation Today: state and issues, Dept. of Linguistics, University of Kerala, Trivandrum, pp 23–25
13. Bharati A, Chaitanya V, Sangal R, Ramakrishnamacharyulu K (1995) Natural language processing: a Paninian perspective. Prentice-Hall of India, New Delhi
14. Huet G (2009) Formal structure of Sanskrit text: Requirements analysis for a mechanical Sanskrit processor. In: Sanskrit Computational Linguistics, Springer, Berlin, pp 162–199
15. Bharati A, Chaitanya V, Sangal R (1994) Paninian framework and its application toanusaraka. Sadhana 19 (1):113–127
16. Bharati A, Kulkarni A (2007) Sanskrit and computational linguistics. In: 1st International Sanskrit Computational Symposium. Department of Sanskrit Studies, University of Hyderabad
17. Nair RR, Devi LS (2011) Sanskrit Informatics: Informatics for Sanskrit studies and research. Centre for Informatics Research and Development
18. Shukla P, Shukl D, Kulkarni A (2010) Vibhakti divergence between Sanskrit and Hindi. In: International Sanskrit Computational Linguistics Symposium, Springer, Berlin, pp 198–208
19. Waibel A, Jain AN, McNair AE, Saito H, Hauptmann AG, Tebelskis J (1991) Janus: a speech-to-speech translation system using connectionist and symbolic processing strategies. In: International conference on acoustics, speech, and signal processing, 1991. ICASSP-91, IEEE, pp 793–796
20. Castano MA, Casacuberta F, Vidal E (1997) Machine translation using neural networks and finite-state models. Theoretical and Methodological Issues in Machine Translation (TMI), pp 160–167
21. Forcada ML, Ñeco RP (1997) Recursive hetero-associative memories for translation. In: International work-conference on artificial neural networks. Springer, Berlin, pp 453–462
22. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. J Mach Learn Res 3 (Feb):1137–1155
23. Schwenk H (2007) Continuous space language models. Comput Speech Lang 21 (3):492–518
24. Mikolov T (2012) Statistical language models based on neural networks. Presentation at Google, Mountain View, 2nd April
25. Devlin J, Zbib R, Huang Z, Lamar T, Schwartz R, Makhoul J (2014) Fast and robust neural network joint models for statistical machine translation. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (vol 1: Long Papers), vol 1, pp 1370–1380
26. Schwenk H, Rousseau A, Attik M (2012) Large, pruned or continuous space language models on a GPU for statistical machine translation. In: Proceedings of the NAACL-HLT 2012 workshop: will we ever really replace the N-gram model? On the future of language modeling for HLT, Association for Computational Linguistics, pp 11–19
27. Wu Y, Yamamoto H, Lu X, Matsuda S, Hori C, Kashioka H (2012) Factored recurrent neural network language model in ted lecture transcription. In: International workshop on spoken language translation (IWSLT) 2012
28. De Gispert A, Iglesias G, Byrne B (2015) Fast and accurate preordering for SMT using neural networks. In: Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 1012–1017
29. Kanouchi S, Sudoh K, Komachi M (2016) Neural reordering model considering phrase translation and word alignment for phrase-based translation. In: Proceedings of the 3rd workshop on Asian translation (WAT2016), pp 94–103
30. Li J, Marton Y, Resnik P, Daumé III H (2014) A unified model for soft linguistic reordering constraints in statistical machine translation. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (vol 1: Long Papers), vol 1, pp 1123–1133
31. Kalchbrenner N, Blunsom P (2013) Recurrent continuous translation models. In: Proceedings of the 2013 conference on empirical methods in natural language processing, pp 1700–1709
32. Luong MT, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. arXiv preprint arXiv: 150804025
33. Thieme P (1958) The indo-european language. Sci Am 199 (4):63–78
34. Indhuja K, Indu M, Sreejith C, Sreekrishnapuram P, Raj PR (2014) Text based language identification system for indian languages following devanagiri script. Int J Eng 3 (4):1
35. Chandra S (2005) Developing a sanskrit analysis system for machine translation. National seminar on translation today state and issues
36. Staal JF (1963) Sanskrit and sanskritization. J Asian Stud 22 (3):261–275
37. Ramanujan P (1992) Computer processing of sanskrit. Computer Processing of Asian Languages CALP-2, IIT Kanpur
38. Mishra V, Mishra R (2009a) Divergence patterns between english and sanskrit machine translation. INFOCOMP 8 (3):62–71
39. Mishra V, Mishra R (2009b) Ann and rule based model for english to sanskrit machine translation. INFOCOMP 9 (1):80–89
40. Mane D, Devale P, Suryawans S (2010) A design towards english to sanskrit machine translation and sy thesizer syste 1 si grule base approach. Int J Multidisp Res Adv Eng 1:1
41. Barkade V, Devale PR (2010) English to sanskrit machine translation semantic mapper. Int J Eng Sci Technol 2 (10):5313–5318
42. Shukla P, Shukla A (2014) English speech to sanskrit speech (ESSS) using rule based translation. Int J Comput Appl 92 (10):1
43. Rathod SG, Sondur S (2012) English to sanskrit translator and synthesizer (ETSTS). Int J Emerg Technol Adv Eng 2 (12):1
44. Bahadur P, Jain A, Chauhan D (2012) Etrans-a complete framework for english to sanskrit machine translation. In: International Journal of Advanced Computer Science and Applications (IJACSA) from International Conference and workshop on Emerging Trends in Technology, Citeseer
45. Upadhyay P, Jaiswal UC, Ashish K (2014) Transish: translator from sanskrit to english-a rule based machine translation. Int J Curr Eng Technol E-ISSN 1:2277–4106
46. Warhade SR, Patil SH, Devale PR (2012) English-to-sanskrit statistical machine translation with ubiquitous application. Int J Comput Appl 51 (1):1
47. Jha GN (2010) The tdil program and the indian langauge corpora intitiative (ilci). In: LREC
48. Nair SS, Kulkarni A (2010) The knowledge structure in amarakośa. In: Sanskrit Computational Linguistics, Springer, Berlin, pp 173–189
49. Kulkarni A (2002) Samsaadhani, a sanskrit computational toolkit. http://scl.samsaadhanii.in/scl/
50. Kumar A, Mittal V, Kulkarni A (2010) Sanskrit compound processor. In: Sanskrit Computational Linguistics, Springer, Berlin, pp 57–69
51. Pandey RK, Jha GN (2016) Error analysis of sahit-a statistical sanskrit-hindi translator. Procedia Comput Sci 96:495–501
52. Chaudhury S, Rao A, Sharma DM (2010) Anusaaraka: An expert system based machine translation system. In: International conference on natural language processing and knowledge engineering (NLP-KE), 2010, IEEE, pp 1–6

53. Pappu A, Sanyal R (2008) Vaakkriti: Sanskrit tokenizer. In: Proceedings of the 3rd international joint conference on natural language processing: volume-II
54. Sachin K (2007) Sandhi splitter and analyzer for sanskrit (with reference to ac sandhi). Mphil degree at SCSS, JNU (submitted, 2007)
55. Kumar S (2007) Sandhi splitter and analyzer for sanskrit. with special reference to aC sandhi
56. Bharati A, Kulkarni AP, Sheeba V (2006) Building a wide coverage sanskrit morphological analyser: A practical approach. In: The 1st national symposium on modelling and shallow parsing of Indian languages, IIT-Bombay
57. Mittal V (2010) Automatic sanskrit segmentizer using finite state transducers. In: Proceedings of the ACL 2010 student research workshop, Association for Computational Linguistics, pp 85–90
58. Jha GN, Agrawal M, Mishra SK, Mani D, Mishra D, Bhadra M, Singh SK, et al (2009) Inflectional morphology analyzer for sanskrit. In: Sanskrit computational linguistics, Springer, pp 219–238
59. Kulkarni A, Pokar S, Shukl D (2010) Designing a constraint based parser for sanskrit. In: Sanskrit Computational Linguistics, Springer, pp 70–90
60. Goyal P, Arora V, Behera L (2009) Analysis of sanskrit text: Parsing and semantic relations. In: Sanskrit Computational Linguistics, Springer, pp 200–218
61. Kulkarni A, Kumar A (2011) Statistical constituency parser for sanskrit compounds. In: Proceedings of ICON
62. Kulkarni A, Ramakrishnamacharyulu K (2013) Parsing sanskrit texts: Some relation specific issues. In: Proceedings of the 5th international sanskrit computational linguistics symposium. DK Printworld (P) Ltd
63. Kulkarni A (2013) A deterministic dependency parser with dynamic programming for sanskrit. In: Proceedings of the 2nd international conference on dependency linguistics (DepLing 2013), pp 157–166
64. Huet G (2006) Shallow syntax analysis in sanskrit guided by semantic nets constraints. In: Proceedings of the 2006 international workshop on Research issues in digital libraries, ACM, p 6
65. Hellwig O (2009) Sanskrittagger: A stochastic lexical and pos tagger for sanskrit. In: Sanskrit computational linguistics, Springer, pp 266–277
66. Hellwig O (2010) Performance of a lexical and pos tagger for sanskrit. In: Sanskrit computational linguistics, Springer, pp 162–172
67. Nandi M, Ramasree R (2013) Rule-based extraction of multi-word expressions for elementary sanskrit texts. Int J 3 (11):1
68. Kumar A, Sheebasudheer V, Kulkarni A (2009) Sanskrit compound paraphrase generator. In: Proceedings of ICON
69. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555
70. Sennrich R, Haddow B (2016) Linguistic input features improve neural machine translation. arXiv preprint arXiv:160602892
71. Li Y, Yuan Y (2017) Convergence analysis of two-layer neural networks with relu activation. In: Advances in neural information processing systems, pp 597–607
72. Montufar GF, Pascanu R, Cho K, Bengio Y (2014) On the number of linear regions of deep neural networks. In: Advances in neural information processing systems, pp 2924–2932
73. News S (2018) Sanskrit News, "department of public health relations". http://mpinfo.org/News/SanskritNews.aspx
74. Belinkov Y, Bisk Y (2017) Synthetic and natural noise both break neural machine translation. arXiv preprint arXiv:171102173
75. Computational Linguistics J (2019) Sanskrit corpora. http://sanskrit.jnu.ac.in/corpora/tagset.jsp
76. Department of Sanskrit Studies UoH (2019) Digital sanskrit corpora. http://sanskrit.uohyd.ac.in/Corpus/
77. Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv preprint arXiv:12125701
78. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980
79. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167
80. Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics, pp 311–318
81. Mikolov T, Karafiát M, Burget L, Černockỳ J, Khudanpur S (2010) Recurrent neural network based language model. In: 11th Annual conference of the international speech communication association