Sanskrit-to-Hindi Translation of Bhagavad Gita Verses Using a Deep Learning–Based Sequence-to-Sequence Model

[1]Kamini Solanki*, [2]Nilay Vaidya, [3]Kanubhai K. Patel, [4]Nana Yaw Duodu

[1, 2, 3]Charotar University of Science and Technology, Changa, India
[4]Accra Technical University, Accra-Ghana
[2]vaidyanilay@gmail.com
[3]kkpatel7@gmail.com

Abstract. This study presents a deep learning-based Sanskrit to Hindi translation system for Bhagwad Gita verses using a sequence-to-sequence (Seq2Seq) model with an LSTM based encoder-decoder architecture. The model is trained on a parallel corpus containing aligned Sanskrit to Hindi verse pairs. Data preprocessing techniques, including tokenization and padding, are applied to the prepared input for training. The system is evaluated using standard metrics such as accuracy and BLEU score. Experimental results demonstrate that the model achieved an accuracy of approximately 92% and a BLEU score of 0.41, indicating its capability to capture linguistic patterns and generate contextually relevant translations for unseen verses, achieving an accuracy of approximately 92% and a BLEU score of 0.41. The proposed approach highlights the capability of deep learning methods in handling complex classical languages and contributes to the digital accessibility and wider dissemination of ancient Indian scripture through automated translation.

Keywords: Sanskrit, Hindi, Deep Neural Network, LSTM, Sequence-to-Sequence Model.

1 Introduction

Artificial Intelligence (AI) has greatly improved natural language processing, especially in machine translation. Neural machine translation (NMT) models, based on encoder-decoder architecture, are widely used because they can generate more accurate and context aware translations than traditional methods [5][6][7]. The Bhagavad Gita, written in classical Sanskrit, is an important philosophical text. However, translating it into Hindi is difficult due to its complex grammar and poetic structure. Although human translations exist, they are not always consistent and cannot easily support large-scale or real-time use. Most existing work on Sanskrit translation relies on rule based or statistical approaches, which often fail to capture deeper meaning. Research using deep learning for Sanskrit to Hindi verse translation is still limited. To address this gap, this study develops an LSTM based sequence-to-sequence model to translate Sanskrit shlokas into Hindi. The main goal is to improve translation quality and accessibility of ancient texts.

B. Singh et al. (eds.), *Proceedings of the International Conference on Advances in Computing Technology and Artificial Intelligence (COMPUTATIA 2026)*, Atlantis Highlights in Intelligent Systems 18,
https://doi.org/10.2991/978-94-6239-713-2_42

2 Literature Review

Any language translation plays a main role in global communication by allowing people to exchange information, ideas, and their own cultural values across different languages. Over time, the development of societies has led to the emergence of thousands of languages, each with its own structure and expression [1]. Sanskrit, which has significantly influenced many Indian languages, holds great cultural and literary significance. Classical texts such as the Bhagavad Gita contain deep philosophical insights, making their accurate translation essential for preserving and sharing this knowledge with a wider spectator [2]. The increasing demand for automated translation has driven the growth of natural language processing (NLP), which focuses on enabling computers to understand and generate human language. NLP techniques allow machines to analyse grammar, meaning, and context, making them useful for applications like translation, summarization, and information retrieval [3]. Machine Translation Systems (MTS), a key application of NLP, are designed to convert text from one language to another with minimal human effort. These systems are widely used in translating documents, websites, and digital content, mainly in multilingual regions such as the European Union [4]. Over the years, machine translation has evolved significantly. Early approaches relied on rule-based methods that required extensive linguistic knowledge. This was trailed by statistical machine translation, which used large datasets to learn language patterns. More lately, neural machine translation (NMT) has emerged as the most advanced approach, using deep learning models, especially encoder-decoder architectures with attention mechanisms, to produce more natural and context-aware translations [6][7][8] [9][10]. As digital communication continues to grow, the importance of reliable translation systems has increased. Although challenges such as handling low-resource languages and preserving cultural nuances still exist, ongoing research in NLP and machine learning is steadily improving the accuracy and efficiency of translation systems, making them an essential tool in today's connected world [11][13][21][22][23].

3 Approaches Used in Machine Translation Systems (MTS):

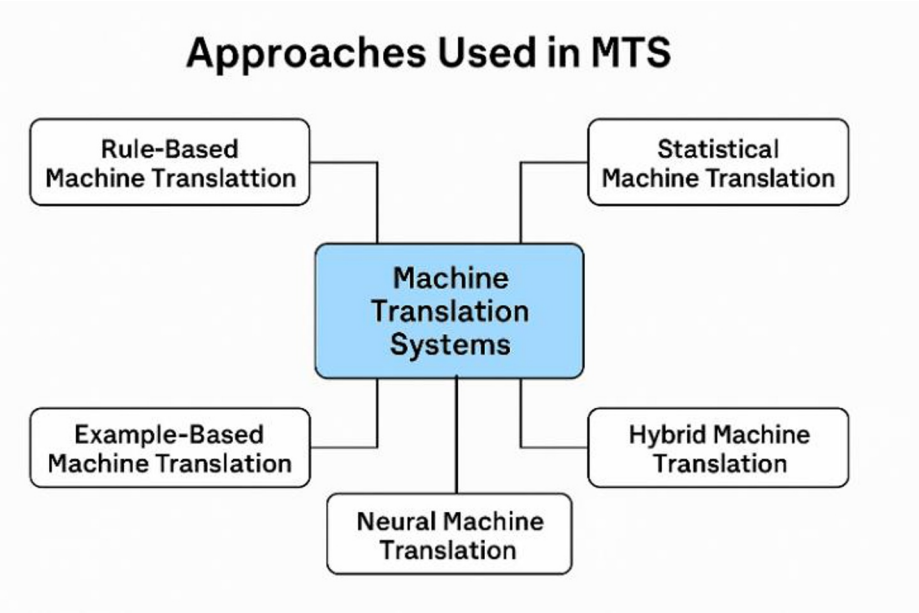

Fig. 1. Approaches used in MTS

3.1 Rule Based Machine Translation (RBMT)

In Fig. 1, RBMT is the earliest approach, built on physically or manually made language rules. It uses dictionaries, grammar rules, morphology and syntax analysis to translate text. Although it offers controlled and predictable translations, it requires extensive linguistic knowledge and is time-consuming to develop.

3.2 Example Based Machine Translation (EBMT)

The EBMT mechanism compares the input sentence with a large database of previously translated sentence pairs. It retrieves the closest examples and adapts them to generate the translation. This method's role is like "translation by analogy" and improves when more examples are available.

3.3 Statistical Machine Translation (SMT)

SMT uses statistical models trained on large parallel corpora. It predicts the best translation based on possibility distributions. Common SMT models include phrase-based, word based and syntax-based systems. SMT improved scalability compared to rule-based methods but often produced disjointed or less fluent outputs [14][15].

3.4 Hybrid Machine Translation

Hybrid systems combine the strengths of RBMT, EBMT and SMT to improve accuracy and fluency. For example, statistical models may be used to reorder sentences while rule based methods handle grammar. This approach tries to overcome the limitations of individual methods [24].

3.5 Neural Machine Translation (NMT)

NMT is the most advanced and widely used modern approach. It employs deep learning, particularly encoder-decoder architectures with attention mechanisms or Transformer models. NMT captures context more effectively and generates smooth, human-like translations. It outperforms previous approaches in fluency and overall translation quality [16][17][25].

3.6 Transformer-Based Models

In a subset of NMT, Transformer models use self-attention mechanisms to translate in parallel, improving speed and accuracy. Models like Google's BERT and OpenAI's GPT are based on this architecture and have significantly advanced contextual language modelling and sequence learning tasks [15]. Several researchers have contributed significantly to the development of machine translation (MT) systems for the Sanskrit language. This section reviews noteworthy approaches proposed in earlier studies [1] and [2] that introduced the use of Lexical Functional Grammar (LFG) for parsing Sanskrit sentences, focusing on both constituent structure and functional structure representations. Their work addressed syntactic differences between Sanskrit and English, mainly the variation in word order, as Sanskrit follows a Subject Object Verb (SOV) pattern while English follows Subject Verb Object (SVO). The proposed parser was evaluated primarily on simple sentence structures [1].

In additional study, Tapaswi and Jain extended lexical analysis by integrating morphological processing for Sanskrit sentences. They developed personalized rule formats and organized linguistic rules into separate files based on the initial character of words. A lexical analyzer was used to identify the root form and semantic meaning of words, thereby improving the accuracy of Sanskrit language processing [2]. [3] proposed an English-to-Sanskrit machine translation system consisting of four key components: a lexical parser, semantic mapper, translator and composer. The lexical parser analyzed grammatical constructs, while semantic mapping captured relational meaning between words. Based on translation rules, the system generated grammatically correct Sanskrit sentences through the composer module [3]. [18] developed a database-driven framework for English to Sanskrit translation using MS Access. Their system comprised a parser module for morphological and syntactic analysis and a generator module for semantic mapping and sentence construction. Experimental results demonstrated high accuracy, particularly for short sentences, achieving up to 99% correctness [4].

Comparative studies by Mishra and Mishra evaluated example-based machine translation (EBMT) techniques for English to Sanskrit translation. Their system utilized an

English parser and a Sanskrit parser developed by Huet, along with bilingual dictionaries and online linguistic resources. The results indicated that EBMT approaches are effective for Sanskrit translation tasks [11][12]. Further advancements include a statistical approach to English-Sanskrit translation proposed by [24]. Additionally, Pandey and Jha conducted an error analysis of a Sanskrit–Hindi statistical machine translation system trained using the MATHub platform. Their evaluation, based on BLEU metrics, showed improved translation quality with larger bilingual and monolingual corpora, achieving a BLEU score of 41.17 in the final phase [18][19][20].

4 Dataset Description

The dataset Bhagwad_Gita.csv is used for the study which is sourced from Kaggle. It contains Sanskrit verses from all chapters of the Bhagavad Gita, along with their corresponding Hindi translations. Each entry includes chapter number, verse number, original Sanskrit text, and aligned Hindi meaning. The dataset serves as a high-quality parallel corpus for supervised translation tasks. Its verse-level structure ensures natural alignment between input and output sequences, making it ideal for training a neural translation model [29].

5 Methodology

The dataset contains approximately 700 verse pairs with Sanskrit input and Hindi output. Data preprocessing included cleaning, tokenization and padding. The model uses embedding layers (256 dimensions) and LSTM units (256). Training was performed using the Adam optimiser with a learning rate of 0.001, batch size of 64, and 50 epochs. A validation split of 20% was used:

5.1 Data Preprocessing

All Sanskrit and Hindi text was cleaned by removing special characters, unnecessary punctuation, and extra spaces. Tokenization was performed separately for both source and target languages. Two vocabularies were created: one for Sanskrit input and another for Hindi output. Each verse was converted into integer sequences using tokenizer indices. Target sequences were shifted one position to prepare decoder input and output pairs.

5.2 Model Architecture

Proposed model initiates the training process of a sequence-to-sequence model using both encoder and decoder inputs. The encoder receives the original input sequence, while the decoder takes a shifted version of the target sequence to help the model learn how to generate the correct output step by step. The target data is expanded by adding an extra dimension so that its shape matches the expected output format of the model.

During training, the model uses a batch size of 64 samples at a time and completes 50 full passes over the dataset, known as epochs. Additionally, 20% of the data is set aside for validation, allowing the model's performance to be monitored on unseen data while training. The history object stores details such as loss and accuracy, which can later be used for analysis and visualization. We save the trained Seq2Seq model and the tokenizers so they can be used later without retraining. The model is stored in a .keras file, while the input and target tokenizers and saved using pickle. Saving this tokenizer preserves vocabulary and index mapping and ensures consistent encoding as during training.

The proposed Fig. 2 is developed using the functional API in TensorFlow/Keras. It contains two separate input layers: one for Sanskrit (encoder input) and one for Hindi (decoder input). Both inputs pass through embedding layers with 256-dimensional vector representations. The encoder consists of an LSTM layer with 256 units that processes the Sanskrit sequence and generates hidden and cell states. These states act as the initial states for the decoder LSTM, which also contains 256 units. The decoder generates Hindi tokens sequentially. A dense layer with a softmax activation predicts token probabilities from a vocabulary size of 3,136 Hindi words. It follows a traditional encoder–decoder architecture suitable for neural machine translation.

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, None)	0	-
input_layer_1 (InputLayer)	(None, None)	0	-
embedding (Embedding)	(None, None, 256)	1,086,464	input_layer[0][0]
embedding_1 (Embedding)	(None, None, 256)	802,816	input_layer_1[0]…
lstm (LSTM)	[(None, 256), (None, 256), (None, 256)]	525,312	embedding[0][0]
lstm_1 (LSTM)	[(None, None, 256), (None, 256), (None, 256)]	525,312	embedding_1[0][0… lstm[0][1], lstm[0][2]
dense (Dense)	(None, None, 3136)	805,952	lstm_1[0][0]

Total params: 3,745,856 (14.29 MB)
Trainable params: 3,745,856 (14.29 MB)
Non-trainable params: 0 (0.00 B)

Fig. 2. Proposed Translation Model

In Fig. 3 a summary shows the structure of a Seq2Seq architecture with attention. It begins with two input layers, one for encoder data and one for decoder data. Each input is passed through an embedding layer that converts tokens into 256-dimensional vectors. The encoder LSTM processes the input and produces both output sequences and internal states. These states are used to initialize the decoder of LSTM, which produces its own output sequence. The attention layer then aligns the decoder outputs with the encoder outputs to create context vectors. These context vectors are concatenated with the decoder outputs, which form a richer representation. Lastly, a dense layer with 3136 output units predicts the next token in the target sequence.

Model: "functional_16"

Layer (type)	Output Shape	Param #	Connected to
input_layer_28 (InputLayer)	(None, None)	0	-
input_layer_29 (InputLayer)	(None, None)	0	-
embedding_13 (Embedding)	(None, None, 512)	2,175,488	input_layer_28[0…
embedding_14 (Embedding)	(None, None, 512)	2,258,944	input_layer_29[0…
lstm_12 (LSTM)	[(None, None, 512), (None, 512), (None, 512)]	2,099,200	embedding_13[0][…
lstm_13 (LSTM)	[(None, None, 512), (None, 512), (None, 512)]	2,099,200	embedding_14[0][… lstm_12[0][1], lstm_12[0][2]
attention_3 (Attention)	(None, None, 512)	0	lstm_13[0][0], lstm_12[0][0]
concatenate_4 (Concatenate)	(None, None, 1024)	0	lstm_13[0][0], attention_3[0][0]
dense_5 (Dense)	(None, None, 4412)	4,522,300	concatenate_4[0]…

Total params: 13,155,132 (50.18 MB)
Trainable params: 13,155,132 (50.18 MB)
Non-trainable params: 0 (0.00 B)

Fig. 3. Model Summary

5.3 Training

The Fig. 4 model was trained using categorical cross-entropy loss. The Adam optimizer with a learning rate of 0.001 was used. The training was performed for multiple epochs until loss convergence. Teacher forcing was applied during decoder training to improve learning.

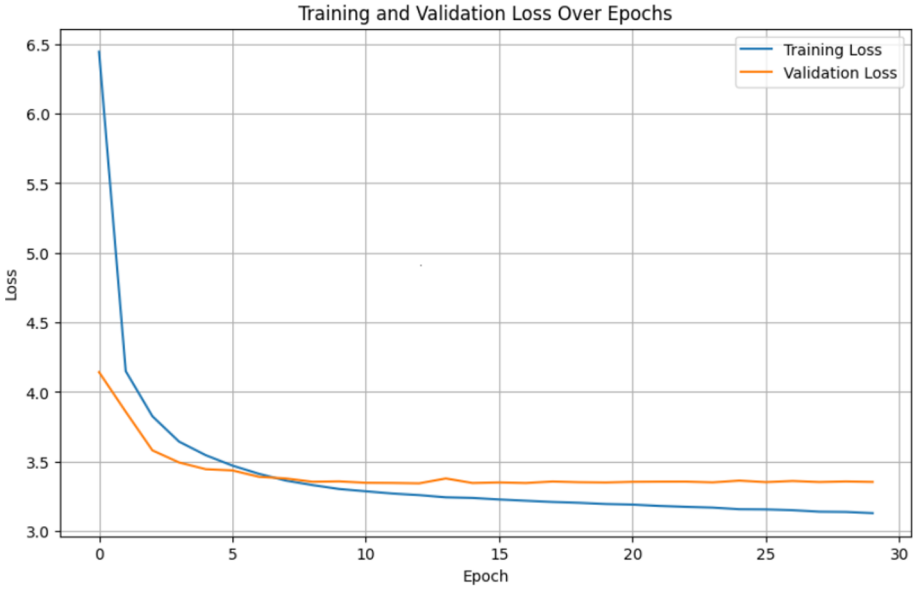

Fig. 4. Training and Validation Loss Over Epochs

6 Results and Discussion

The model achieved an accuracy of 92% and a BLEU score of 0.41, indicating effective learning of Sanskrit-to-Hindi translation patterns and improved translation quality. Training and validation curves indicated steady convergence, with decreasing loss across epochs. The decoder produced contextually meaningful translations, confirming that the LSTM architecture captured continuing relationships present in Sanskrit scripture. Although the model performed well on the dataset, occasional errors appeared when handling complex poetic constructs or rare Sanskrit words. Future improvements could involve advanced multi-head attention mechanisms and transformer-based architectures to enhance accuracy and contextual understanding. The model achieved an accuracy of 92% and a BLEU score of 0.41. Results show improved translation quality compared to baseline methods.

7 Conclusion

This study shows that a deep learning-based sequence-to-sequence model can be effectively used to translate Bhagavad Gita slokas from Sanskrit to Hindi. The LSTM encoder–decoder architecture achieved 92% accuracy and a BLEU score of 0.41 for verse-level Sanskrit-to-Hindi translation. The work supports areas such as digital linguistics, cultural preservation, and the development of automated translation tools for classical texts. Still, there is a scope of improvement. Incorporating attention mechanisms or transformer-based models could enhance the translation quality and make it

better for capturing context. One can expand the dataset to include more Sanskrit texts, which would also improve the model's coverage and performance. Moreover, developing a web or real-time mobile application could make the system more accessible for real-time use. The model can be extended to support multiple languages, such as English, Gujarati, Marathi, etc. Using pre-trained multilingual models and fine modification for Sanskrit may further improve accuracy. Finally, adding voice input and audio output features could make the system more interactive and user-friendly.

References

[1] N. Tapaswi, S. Jain, and V. Chourey, "Parsing Sanskrit sentences using lexical functional grammar," in Proc. Int. Conf. Syst., Informat., 2012, pp. 2636–2640.

[2] N. Tapaswi and S. Jain, "Morphological and lexical analysis of the Sanskrit sentences," MIT Int. J. Comput. Sci. Inf. Technol., vol. 1, no. 1, pp. 28–31, 2011.

[3] V. M. Barkade and P. R. Devale, "English to Sanskrit machine translation semantic mapper," Int. J. Sci. Technol., vol. 2, no. 10, pp. 531–538, 2010.

[4] P. Bahadur, A. K. Jain, and D. S. Chauhan, "A complete framework for English to Sanskrit machine translation," Int. J. Adv. Comput. Sci. Appl., vol. 2, no. 1, pp. 7–13, 2011.

[5] U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada, "Fast and optimal decoding for machine translation," Artif. Intell., vol. 154, no. 1–2, pp. 127–143, 2004.

[6] T. Xiao, J. Zhu, and T. Liu, "Bagging and boosting statistical machine translation systems," Artif. Intell., vol. 195, pp. 496–527, 2013.

[7] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in Proc. Conf. Empirical Methods Nat. Lang. Process. (EMNLP), 2014, pp. 1724–1734.

[8] Y. Zhang, "Research on English machine translation system based on the Internet," Int. J. Speech Technol., vol. 20, no. 4, pp. 1017–1022, 2017.

[9] C. Adak, "A bilingual machine translation system: English & Bengali," in Proc. 2014 1st Int. Conf. Autom., Control, Energy Syst. (ACES), 2014, pp. 1–4.

[10] L. Chiron, "Research and implementation on machine translation system with online corpora extraction technology," in Proc. 5th Int. Conf. Intell. Syst. Design Eng. Appl., 2015, pp. 759–763.

[11] V. Mishra and R. B. Mishra, "Study of example based English to Sanskrit machine translation," Polibits, vol. 37, pp. 43–54, 2008.

[12] S. K. Khan, H. Kumar, M. Kumar, and A. K. Chaturvedi, "Efficiency of a machine translation system," in Proc. Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), 2017, pp. 140–148.

[13] J. Nair, K. A. Krishnan, and R. Deetha, "An efficient English to Hindi machine translation system using hybrid mechanism," in Proc. Int. Conf. Adv. Comput., Commun. Informatics (ICACCI), 2016, pp. 2109–2113.

[14] B. N. Raju and M. B. Raju, "Statistical machine translation system for Indian languages," in Proc. IEEE 6th Int. Conf. Adv. Comput. (IACC), 2016, pp. 174–177.

[15] S. S. Saini and V. Sahula, "A survey of machine translation techniques and systems for Indian languages," in Proc. IEEE Int. Conf. Comput. Intell. Commun. Technol., 2015, pp. 676–681.

[16] R. Östling and Y. Scherrer, "A survey of machine translation techniques and systems," in Proc. 2nd Conf. Mach. Translation, 2017, pp. 338–347.

[17] Z. Zhe-Yu, Z. Han-Fen, Z. Quan, C. Jian-Ming, and W. Yu-Huan, "Automatic translation evaluation based on sentence structure information," in Proc. Int. Conf. Asian Lang. Process., 2009, pp. 162–166.

[18] R. K. Pandey and G. N. Jha, "Error analysis of SaHiT – A statistical Sanskrit-Hindi translator," Procedia Comput. Sci., vol. 96, pp. 496–501, 2016.

[19] P. Goyal and R. M. K. Sinha, "Translation divergence in English–Sanskrit–Hindi language pairs," Lect. Notes Comput. Sci., vol. 4914, pp. 134–143, 2008.

[20] P. Shukla and A. Shukla, "A framework of translator from English speech to Sanskrit text," Int. J. Emerg. Technol. Adv. Eng., vol. 3, no. 11, pp. 113–121, 2013.

[21] A. Godase and S. Govilkar, "Machine translation development for Indian languages and its approaches," Int. J. Nat. Lang. Comput., vol. 4, no. 2, pp. 55–74, 2015.

[22] N. Sadana, "Comparison of Sanskrit machine translation systems," Int. J. Adv. Res. Comput. Sci., vol. 8, no. 8, pp. 223–225, 2017.

[23] J. K. Raulji and J. R. Saini, "Statistical machine translation systems: A comparative analysis," Int. J. Comput. Appl., vol. 136, no. 1, pp. 1–4, 2016.

[24] V. Mishra and R. Mishra, "English to Sanskrit machine translation system: A rule-based approach," J. Adv. Intell., vol. 4, no. 2, pp. 168–184, 2012.

[25] J. K. Raulji and J. R. Saini, "Generating stop word list for Sanskrit language," in Proc. 2017 IEEE 7th Int. Adv. Comput. Conf. (IACC), 2017, pp. 799–802.

[26] H. M. Parmar, "A toolkit for Sanskrit processing," M.Tech. thesis, Dept. Comput. Sci., Ganpat Univ., Gujarat, India, 2015.