IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2026

DOI: 10.17148/IARJSET.2026.13360

AI BASED TEXT TO TEXT MACHINE TRANSLATION FROM NEPALISE AND SINHALESE TO ENGLISH

Yogeshkumar.V[1], Dr. R. Praba[2]

Student, Department of Information Technology, Dr. N.G.P. Arts and Science College, Coimbatore, Tamil Nadu, India[1]

Associate Professor, Department of Information Technology, Dr. N.G.P. Arts and Science College, Coimbatore, Tamil Nadu, India[2]

Abstract: The rapid growth of digital communication has increased the demand for efficient and accurate language translation systems. However, most existing translation tools rely heavily on internet connectivity, limiting accessibility in low-network regions, and raising significant data privacy concerns. Furthermore, traditional rule-based translation systems often produce contextually incorrect and slow translations, particularly for less-resourced languages such as Nepali and Sinhala. To address these limitations, this paper presents an AI-Based Text-to-Text Machine Translation System that translates Nepali and Sinhala text into English in an offline environment. The proposed system leverages deep learning techniques using PyTorch and HuggingFace Transformers to implement a pre-trained neural machine translation model capable of generating accurate and context-aware translations. The system is developed using Python and Django, ensuring a secure, user-friendly interface while maintaining offline functionality. The architecture consists of multiple modules including input handling, text pre-processing, neural translation engine, post-processing, and result display. By eliminating internet dependency and ensuring local processing, the system enhances privacy, accessibility, and reliability. Experimental evaluation demonstrates that the proposed solution provides efficient, grammatically coherent, and contextually meaningful English translations while maintaining complete offline usability.

Keywords: Machine Translation, Low-Resource Languages, Nepali, Sinhalese, Transformer, Transfer Learning, Sub word Tokenization

I. INTRODUCTION

Language plays a crucial role in global communication, education, business, and digital interaction. With the increasing globalization of information, the need for accurate and efficient machine translation systems has grown significantly. Machine Translation (MT) enables automatic conversion of text from one language to another, reducing communication barriers and facilitating cross-lingual understanding. In recent years, Artificial Intelligence (AI) and deep learning techniques have greatly improved translation quality compared to traditional rule-based and statistical approaches.

Most widely used translation systems, such as online neural machine translation platforms, rely heavily on internet connectivity. While these systems provide fast and accurate translations, they present several limitations. Continuous internet dependency restricts access in rural or low-network regions, especially in developing countries. Additionally, online translation services raise serious concerns regarding data privacy and security, as user input text is processed and stored on external servers. Sensitive information such as personal documents, academic records, or confidential communication may be exposed to potential risks.

1.1 LITERATURE REVIEW

Machine Translation (MT) has evolved significantly over the past decades, transitioning from rule-based systems to advanced neural network-based models. Early translation systems were primarily **Rule-Based Machine Translation (RBMT)** systems, which relied on predefined linguistic rules and bilingual dictionaries. Although these systems provided structured translations, they lacked contextual understanding and often produced grammatically incorrect outputs, particularly for complex sentence structures.

Subsequently, **Statistical Machine Translation (SMT)** emerged as an improvement over rule-based approaches. SMT systems utilized large parallel corpora and probabilistic models to generate translations based on statistical patterns. While SMT improved translation fluency and scalability, it still struggled with long-range dependencies and contextual coherence, especially for morphologically rich and low-resource languages such as Nepali and Sinhala.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2026

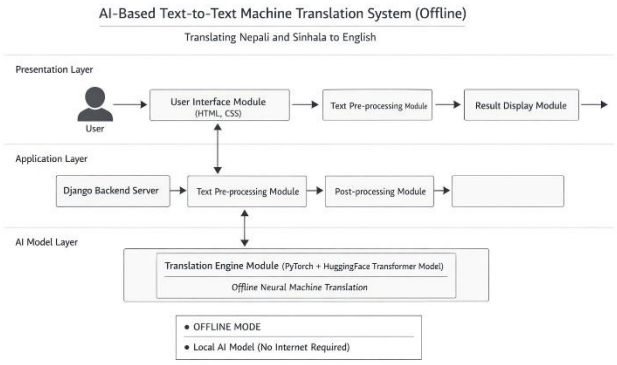
DOI: 10.17148/IARJSET.2026.13360

The introduction of **Neural Machine Translation (NMT)** marked a major breakthrough in the field of Natural Language Processing (NLP). NMT systems employ deep learning architectures, particularly encoder–decoder models with attention mechanisms, to capture semantic meaning and contextual relationships within sentences. The development of the Transformer architecture further enhanced translation quality by enabling parallel processing and better handling of long-distance dependencies. Transformer-based models have significantly outperformed traditional RBMT and SMT systems in terms of fluency, accuracy, and contextual relevance.

II. SYSTEM ARCHITECTURE

The proposed AI-Based Text-to-Text Machine Translation System follows a modular and layered architecture designed to ensure efficient offline translation, secure processing, and user-friendly interaction. The system integrates web technologies with deep learning frameworks to provide accurate translation from Nepali and Sinhala to English without requiring internet connectivity.

The overall architecture consists of five major components: User Interface Module, Text Pre-processing Module, Translation Engine Module, Post-processing Module, and Result Display Module. These components interact sequentially to perform end-to-end translation.



System Flow Diagram

2.2 Modules of the Proposed System

The proposed AI-Based Text-to-Text Machine Translation System is designed using a modular architecture to ensure maintainability, scalability, and efficient processing. Each module performs a specific function in the translation pipeline, enabling smooth interaction between user input and the AI-based translation engine. The major modules of the system are described below.

1. User Interface Module

The User Interface (UI) Module serves as the interaction layer between the user and the system. It is developed using HTML and CSS to provide a simple, responsive, and user-friendly environment.

This module allows users to:

A. Enter text in Nepali or Sinhala
B. Select the source language
C. Submit the translation request

The interface ensures accessibility and ease of use, even for non-technical users. It forwards the user input to the backend server for further processing.

2. Text Pre-processing Module

The Text Pre-processing Module prepares the input text for neural translation. Since AI models require structured and tokenized input, this module performs several preprocessing operations, including:

D. Removal of unnecessary symbols or special characters
E. Text normalization
F. Tokenization

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2026

DOI: 10.17148/IARJSET.2026.13360

G. Conversion of text into numerical format compatible with the transformer model
These steps ensure that the input text is properly formatted and suitable for accurate model inference.

3. Translation Engine Module
The Translation Engine Module is the core component of the system. It implements a pre-trained Neural Machine Translation (NMT) model using PyTorch and HuggingFace Transformers.
This module performs the following functions:
H. Encodes the source language text
I. Captures contextual and semantic relationships
J. Decodes the processed data into English text
The model is stored locally, enabling offline translation without internet dependency. By leveraging transformer architecture, the system ensures improved contextual accuracy and grammatical coherence.

4. Post-processing Module
After translation, the output generated by the neural model may contain unwanted tokens or formatting inconsistencies. The Post-processing Module refines the output by:
K. Removing special tokens
L. Correcting spacing and punctuation
M. Formatting the text into readable English
This module enhances the clarity, structure, and presentation quality of the translated content.

5. Result Display Module
The Result Display Module presents the final translated English text to the user. It ensures:
N. Clear formatting of output
O. Quick response time
P. Proper alignment and readability
The translated text is displayed immediately after processing, providing a smooth and efficient user experience.

Summary of Modules (Tabular Format)

Module Name	Description
User Interface Module	Accepts user input and language selection
Text Pre-processing Module	Cleans, tokenizes, and prepares input text
Translation Engine Module	Performs AI-based neural translation offline
Post-processing Module	Refines and formats translated output

III. METHODOLOGY

The proposed AI-Based Text-to-Text Machine Translation System follows a structured methodology to ensure accurate, context-aware, and privacy-preserving translation from Nepali and Sinhala to English. The methodology consists of sequential stages including input acquisition, text preprocessing, neural model inference, post-processing, and result generation. Each stage plays a crucial role in maintaining translation quality and offline functionality.

Input Acquisition
The translation process begins when the user enters text in Nepali or Sinhala through the user interface. The user selects the source language and submits the input for translation. The Django backend receives the request and forwards the input text to the processing pipeline. Proper validation is performed to ensure that the input field is not empty and contains valid textual data.

Text Pre-processing
Before translation, the input text undergoes preprocessing to convert it into a structured format suitable for the neural model. This stage includes:

- Removal of unnecessary symbols and unwanted characters
- Text normalization to maintain uniform structure
- Tokenization of input text
- Conversion of tokens into numerical representations using the tokenizer associated with the transformer model

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2026

DOI: 10.17148/IARJSET.2026.13360

Tokenization ensures that the input sentence is broken down into meaningful subword units that the model can process efficiently. This step enhances translation accuracy by maintaining contextual relationships within the sentence.

Neural Machine Translation

The core translation process is performed using a pre-trained Neural Machine Translation (NMT) model based on the Transformer architecture. The methodology follows an encoder–decoder framework:

Encoding Phase:

The encoder processes the tokenized source language text and generates contextual embeddings representing semantic meaning.

Attention Mechanism:

The attention layer captures long-range dependencies and contextual relationships within the sentence, improving translation quality.

Decoding Phase:

The decoder generates the corresponding English translation token by token based on encoded representations.

The model is implemented using PyTorch and HuggingFace Transformers and stored locally within the system. This ensures that translation is performed entirely offline, eliminating the need for internet connectivity and protecting user data privacy.

Post-processing

After the neural model generates the translated output, the system performs post-processing to improve readability and presentation. This includes:

- Removal of special tokens generated during decoding
- Correction of spacing and punctuation
- Formatting the output into grammatically structured English sentences

This stage ensures that the final translation is coherent, fluent, and user-friendly.

Output Generation

The processed English translation is sent back to the user interface through the Django backend and displayed instantly. The system ensures minimal latency and smooth response time. Since all computations are executed locally, the methodology guarantees secure and uninterrupted operation.

Key Features of the Methodology

- End-to-end offline translation
- Transformer-based context-aware translation
- Secure local processing
- Modular and scalable workflow
- Reduced dependency on external servers

The structured methodology ensures that the system delivers accurate, efficient, and privacy-preserving translation for Nepali and Sinhala languages, making it suitable for deployment in low-connectivity environments.

IV. DATAFLOW DESCRIPTION

The proposed AI-Based Text-to-Text Machine Translation System follows a structured data flow mechanism to ensure accurate translation, secure processing, and efficient offline operation. The data flow is organized into sequential stages, beginning from user input and ending with translated output display. Each stage ensures smooth transformation of raw text into contextually accurate English translation.

Step 1: User Input and Language Selection

The data flow begins when the user enters Nepali or Sinhala text into the input field through the user interface. The user selects the source language and submits the request. The frontend interface forwards the input data to the Django backend server for further processing. Input validation is performed to ensure that the text field is not empty and contains valid characters.

Step 2: Backend Processing and Request Handling

Once the input is received, the Django backend processes the request and initiates the translation workflow. The backend acts as the central controller, directing the input text to the appropriate processing modules. Since the system operates offline, no external API calls are made, ensuring data privacy and secure local execution.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2026

DOI: 10.17148/IARJSET.2026.13360

Step 3: Text Pre-processing

The input text is passed to the Text Pre-processing Module. In this stage:

- Unwanted symbols and extra spaces are removed.
- Text normalization is performed.
- Tokenization converts the text into smaller subword units.
- Tokens are transformed into numerical vectors compatible with the transformer model.

This step prepares the text in a structured format suitable for neural model inference.

Step 4: Neural Translation Processing

The processed tokens are forwarded to the Translation Engine Module. The transformer-based neural machine translation model performs:

- Encoding of source language text
- Contextual analysis using attention mechanisms
- Decoding into English text

The model generates translated output tokens sequentially. Since the model is stored locally, all computations occur within the system environment without internet dependency.

Step 5: Post-processing of Output

The generated output tokens are sent to the Post-processing Module. This stage:

- Removes special tokens (such as start/end markers)
- Corrects punctuation and spacing
- Converts token sequences into readable English sentences

This ensures clarity and grammatical correctness in the final output.

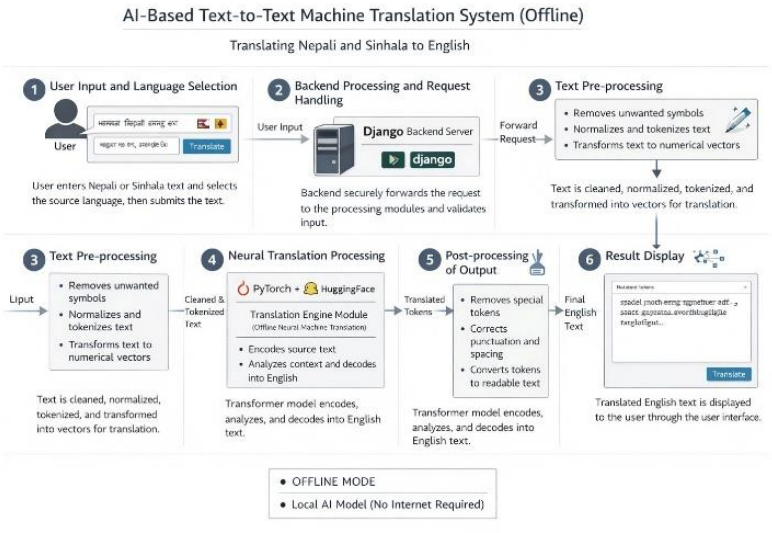
Step 6: Result Display

The finalized English translation is returned to the Django backend and displayed through the Result Display Module on the user interface. The user can view the translated text instantly. The entire data flow is completed in real time, providing a smooth and efficient user experience.

Key Characteristics of Data Flow

- Fully offline processing
- Secure local execution
- Sequential modular workflow
- Minimal latency
- No third-party data sharing

The structured data flow ensures that the system maintains high accuracy, privacy protection, and efficient performance while translating Nepali and Sinhala text into English.



Data Flow diagram

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2026

DOI: 10.17148/IARJSET.2026.13360

V. IMPLEMENTATION

The AI-Based Text-to-Text Machine Translation System was implemented as a fully offline web-based application integrating deep learning frameworks with a robust backend architecture. The implementation focuses on accuracy, privacy, modularity, and efficient performance while translating Nepali and Sinhala text into English.

5.1 Development Environment

The system was developed using the following technologies:

- **Programming Language:** Python 3.x
- **Web Framework:** Django
- **Deep Learning Framework:** PyTorch
- **NLP Library:** HuggingFace Transformers
- **Frontend Technologies:** HTML and CSS
- **Operating System:** Windows
- **Hardware Requirement:** Minimum 8 GB RAM, Intel i5 / Ryzen 5 or above

The selected technology stack ensures compatibility, scalability, and efficient model execution in an offline environment.

5.2 Backend Implementation

The backend of the system was developed using the Django framework. Django handles:

- HTTP request processing
- Routing and URL management
- Input validation
- Integration with the neural translation model
- Output rendering

When a user submits text, Django captures the request and forwards the input to the preprocessing pipeline. The backend manages the interaction between the frontend interface and the AI model, ensuring smooth data exchange and secure local processing.

5.3 Model Integration

The core translation functionality is powered by a pre-trained Neural Machine Translation (NMT) model based on Transformer architecture. The model was loaded using HuggingFace Transformers and executed through PyTorch. The implementation includes:

- Loading the tokenizer associated with the pre-trained model
- Converting input text into tokenized format
- Generating translation using model inference
- Decoding output tokens into English text

The model is stored locally within the system directory. During execution, inference is performed entirely offline, ensuring data confidentiality and eliminating internet dependency.

5.4 Pre-processing and Post-processing Implementation

The pre-processing component was implemented using Python-based text handling utilities. It performs:

- Text cleaning
- Tokenization
- Encoding into model-compatible tensors

After translation, post-processing functions remove special tokens and ensure proper formatting of the generated text. This step improves readability and grammatical structure.

5.5 Frontend Implementation

The frontend was designed using HTML and CSS to provide a clean and responsive user interface. The interface includes:

- Text input area
- Language selection dropdown
- Translate button
- Output display section

The design ensures simplicity and ease of navigation, making the system accessible to users with minimal technical knowledge.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2026

DOI: 10.17148/IARJSET.2026.13360

VI. CONCLUSION

The AI-Based Text-to-Text Machine Translation System successfully demonstrates the implementation of an offline neural machine translation framework for converting Nepali and Sinhala text into English. The project addresses key limitations of existing online translation tools, particularly internet dependency, data privacy concerns, and limited support for low-resource languages.By leveraging transformer-based Neural Machine Translation models implemented using PyTorch and HuggingFace Transformers, the system provides context-aware, grammatically coherent, and semantically meaningful translations. The integration of Django as the backend framework ensures smooth request handling, modular workflow management, and efficient interaction between the user interface and the AI model. The modular architecture enhances maintainability, scalability, and system performance.

REFERENCES

[1]. HuggingFace, "Transformers Documentation," Available: https://huggingface.co/docs/transformers
[2]. Helsinki-NLP, "Marian Neural Machine Translation Models," Available: https://huggingface.co/Helsinki-NLP
[3]. OPUS, "OPUS Parallel Corpus – Open Parallel Corpus for NLP," Available: https://opus.nlpl.eu
[4]. PyTorch, "PyTorch Official Documentation," Available: https://pytorch.org/docs
[5]. Django Software Foundation, "Django Official Documentation," Available: https://docs.djangoproject.com
[6]. A. Vaswani et al., "Attention Is All You Need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 6000–6010.
[7]. P. Koehn, "Neural Machine Translation," *Cambridge University Press*, 2020.
[8]. Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *arXiv preprint arXiv:1609.08144*, 2016.
[9]. J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional

