From stracth translation model: Telugu to English

Team No: 6

Lead: Rushika Sritha

Members:

1. Narendra
2. Abhi Ram
3. Meghana

Made with GAMMA

Abstract: Bridging Languages with AI

Project Goal

Develop a Neural Machine Translation (NMT) system from scratch for Telugu to English using the Transformer architecture.

The Challenge

Telugu, a low-resource Dravidian language with over 80 million speakers, presents unique MT challenges: limited parallel corpora, complex morphology, and syntactic differences from English.

Our Approach

Implement a sequence-to-sequence Transformer model in Python, trained on available Telugu-English parallel datasets (AI4Bharat, Samanantar).

Evaluation & Impact

Performance evaluated using BLEU scores. Aims to improve accessibility for Telugu speakers, outperforming basic rule-based systems.

Made with GAMMA

Introduction: The Need for Advanced MT

Machine Translation (MT) is crucial for bridging communication gaps in multilingual societies like India. Telugu, a classical Indian language, is morphologically rich with agglutinative features, free word order, and pro-drop tendencies.

Translating Telugu to English is challenging due to these structural differences and limited high-quality parallel data. Traditional methods struggle with complexity and context.

Our project builds a Transformer-based NMT model for Telugu-English, focusing on low-resource scenarios, aiming for practical applications in education and content localisation.

Encoder-Decoder Architecture

Parallel Corpora Training

Achieve Translation Quality

Real-World Application

Made with GAMMA

Limitations of Existing Systems

While commercial tools and older approaches exist, they fall short for low-resource languages like Telugu.

Commercial Tools

Google Translate, Microsoft Translator, QuillBot often use pre-trained NMT models. They perform well on common phrases but struggle with idiomatic expressions, cultural nuances, and domain-specific text due to limited Telugu training data.

Rule-Based & Statistical

Early approaches relied on hand-crafted rules or statistical probabilities. They achieved moderate accuracy for simple sentences but lacked fluency, struggled with complex structures, and failed to handle long-range dependencies effectively.

Key Limitations

Poor handling of Telugu's agglutination, sandhi (word joining), and cultural specifics. Low accuracy on rare words and metaphors. Dependency on large pre-trained models rather than custom implementations for specific low-resource challenges.

Made with GAMMA

Our Proposed System: A Custom Transformer

We implement a from-scratch Transformer-based NMT model, tailored to address Telugu's unique linguistic characteristics.

1

Architecture

Encoder-Decoder structure with multi-head self-attention, feed-forward networks, and positional encodings.

2

Subword Tokenization

Utilising techniques like SentencePiece to effectively manage vocabulary sparsity inherent in low-resource languages.

3

Training Data

Leveraging parallel corpora from IITB Corpus, Samanantar, or AI4Bharat datasets for robust training.

4

Low-Resource Techniques

Employing back-translation, transfer learning from related languages (e.g., Tamil), or data augmentation to overcome data scarcity.

5

Decoding

Implementing beam search decoding for generating high-quality and contextually relevant translations.

Made with GAMMA

System Analysis: Software & Hardware

Software Requirements

- OS: Windows/Linux/macOS
- Programming: Python 3.8+
- Libraries: PyTorch/TensorFlow, NumPy, Pandas, SentencePiece, SacreBLEU
- Datasets: Open-source parallel corpora (OPUS, AI4Bharat)
- Development Tools: Jupyter Notebook, VS Code

Hardware Requirements

- CPU: Intel i5 or equivalent (minimum)
- RAM: 16 GB (recommended for efficient training)
- GPU: NVIDIA with CUDA support (e.g., GTX 1060 or better) for accelerated training (highly recommended)
- Storage: 50 GB+ for datasets and model checkpoints

This setup ensures technical viability and economic affordability for an academic research environment.

Made with GAMMA

Transformer System Architecture

Encoder Attention

Multi-head self-attention layers

Decoder Attention

Masked self-attention and encoder-decoder attention

Input Embedding

Telugu tokens + positional encodings

Feed-Forward

Position-wise feed-forward networks

Output Layer

Linear + softmax produces English tokens

The core of our NMT model is the Transformer architecture, designed for efficient sequence-to-sequence translation.

1

Input Embedding + Positional Encoding

Converts tokens into vectors, incorporating vital positional information.

2

Encoder

A stack of N layers (e.g., 6), each with multi-head self-attention and a feed-forward network to process the source sequence.

3

Decoder

Another stack of N layers, featuring masked self-attention, encoder-decoder attention, and a feed-forward network to generate the target sequence.

4

Output Layer

A linear layer followed by softmax to compute token probabilities for the English output.

Made with GAMMA

Project Modules

Data Preprocessing

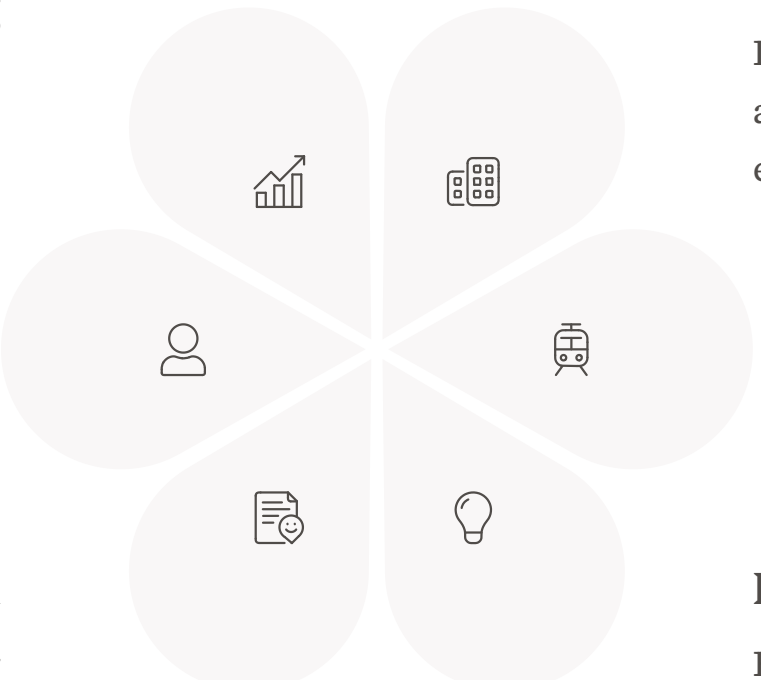
Cleaning, tokenization, subword segmentation, and vocabulary building.

User Interface

A simple web/app interface (e.g., Streamlit/Gradio) for demonstration.

Evaluation

Computation of BLEU, PER, and other metrics on the test set.



Model Building

Implementation of Transformer layers: attention mechanisms, embeddings, and encoder/decoder stacks.

Training

Loss calculation (cross-entropy), optimizer (Adam), and training loop with validation.

Inference

Beam search decoding for generating translations from new input.

Made with GAMMA

Input & Output Screens

Input Screen

A user-friendly interface for inputting Telugu text or uploading files.

- Text box for Telugu sentences.
- Option to upload Telugu text files.
- "Translate" button to initiate the process.

Example Input: "నాకు జ్వరంగా ఉంది"

"అతనికి తలనొప్పి ఉంది"

```
telugu_text = "నాకు జ్వరంగా ఉంది"
english_output = translate_telugu_to_english(telugu_text)

print("TELUGU :", telugu_text)
print("ENGLISH:", english_output)
```

```
TELUGU : నాకు జ్వరంగా ఉంది
ENGLISH: I have a fever.
```

Output Screen

Displays the translated English text alongside the original Telugu for comparison.

- Translated English output.
- Original Telugu input for context.
- Additional screens for training dashboards, loss/BLEU graphs, and error examples.

Example Output: " I have a fever."

"He has a headache."

```
telugu_text = "అతనికి తలనొప్పి ఉంది"
english_output = translate_telugu_to_english(telugu_text)

print("TELUGU :", telugu_text)
print("ENGLISH:", english_output)
```

```
TELUGU : అతనికి తలనొప్పి ఉంది
ENGLISH: He has a headache.
```

A processing screen with a progress bar will be displayed during inference.

Made with GAMMA

Conclusion & Future Scope

Conclusion

Our project successfully implements a Transformer-based NMT model from scratch for Telugu-to-English translation. It demonstrates the power of attention mechanisms in handling low-resource languages, achieving satisfactory results on benchmark datasets despite data scarcity. This work contributes to preserving Telugu cultural content and improving digital accessibility, outperforming basic existing systems.

Future Scope

- Fine-tune with larger/multilingual datasets (e.g., NLLB-200 integration).
- Extend to multilingual (add Tamil/Hindi) or bidirectional translation.
- Handle domain-specific content (e.g., poetry, news) via specialised corpora.
- Incorporate speech-to-text for voice translation applications.
- Deploy as a mobile/web app with real-time translation capabilities.
- Improve low-resource handling using advanced techniques like zero-shot learning or synthetic data generation.

Technologies for future deployment: Frontend with React Vite and Tailwind CSS, Backend with Flask, and the facebook/nllb-200-distilled-600M model for enhanced capabilities.

Made with GAMMA