ENGLISH TO TELUGU TEXT TRANSLATION SYSTEM

Report submitted to
National Institute of Technology Manipur
for the award of the degree

of

Bachelor of Technology
In Computer Science & Engineering

By

Komal Reddy (20103005)

Jayanth Reddy (20103010)

Jnana Yasaswini (20103023)

Banothu Akhila (20103052)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR

DECLARATION

We hereby declare that the project work entitled "Design and Development of English to Telugu Translation System based on Transformer with Deep Learning" submitted to the NIT MANIPUR, is a record of an original work done by us under the guidance of **Dr. Khelchandra thongam** , Associate professor, Computer Science and Engineering NIT MANIPUR, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Komal Reddy K Jayanth Reddy Jnana Yasaswini Akhila Banothu

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organisations.We would like to extend our sincere thanks to all of them. We are highly indebted to **Dr. Khelchandra thongam** for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project as far as we did. We would like to express our gratitude towards our parents and member of NIT MANIPUR for their kind cooperation and encouragement which help us in completing the project as far as we did. We would like to express our special gratitude and thanks to industry persons for giving us such attention and time. Our thanks and appreciation also go to our colleagues in developing the project and people who have willingly helped us out with their abilities

राष्ट्रीय प्रोद्योगिकी संस्थान, मणिपूर
NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR
Imphal, Manipur, Ph.(0385) 2058566 / 2445812
Email: admin@nitmanipur.ac.in , Website: www.nitmanipur.ac.in
An Autonomous Institute under MHRD, Govt. Of India

Certificate

This is to certify that Dissertation Report entitled, "English to Telugu Text Translation System" Submitted by "*Komal Reddy, Jayanth Reddy, Jnana Yasaswini, Banothu Akhila*" to National Institute of Technology Manipur, India ,is a record of bonafide Project work carried out by them under the supervision of **Dr. Khelchandra Thongam**, **Associate professor of NIT Manipur** under the department of **Computer Science & Engineering**, NIT Manipur and is worthy of consideration for the award of the degree of Bachelor of Technology in **Computer Science & Engineering Department** of the Institute.

Dr. Khundrakpam Johnson Singh
Head of Department
CSE

Dr. Khelchandra thongam
Project Supervisor
CSE

Date: 6th December 2023

ABSTRACT

In our rapidly globalizing world, the ability to seamlessly communicate across linguistic boundaries is more crucial than ever. As a response to the growing demand for efficient language translation systems, our project focuses on the "Design and Development of an English to Telugu Translation System based on Transformer with Deep Learning."

This project harnesses the power of Transformers, combined with deep learning techniques, to create a robust and accurate translation system. The choice of English to Telugu translation serves as a testament to our commitment to addressing linguistic diversity and enabling information exchange for the Telugu-speaking community.

Our system aims not only to bridge the language gap but also to enhance translation quality, ensuring that nuances and cultural context are preserved in the process. The deep learning algorithms employed in this project undergo extensive training on large parallel corpora of English and Telugu text, enabling the model to grasp the intricacies of both languages and produce contextually accurate translations.

CONTENTS

INTRODUCTION

1.1 Text to Text Translation using Transformer

In the ever-evolving landscape of natural language processing (NLP), the advent of transformer-based models has revolutionized the field, offering unprecedented capabilities in tasks like text-to-text translation. Transformers, with their attention mechanisms and parallel processing, have proven to be a game-changer by capturing contextual nuances and dependencies within language. This innovation has not only overcome traditional limitations in translation but has also paved the way for more accurate, context-aware, and fluent translations across a multitude of languages. This article explores the transformative impact of employing transformers in text-to-text translation, delving into the underlying mechanisms that make these models highly effective and showcasing their potential to bridge linguistic gaps in our interconnected globalized world.

1.2 Objectives

1. Implement a transformer-based deep learning model for text translation

2. Fine-tune the model to handle language-specific nuances and improve performance

3.implement appropriate evaluation metrics (e.g., BLEU score, accuracy) to quantitatively assess the translation quality.

4.Create an intuitive and user-friendly interface for users to input English text and receive the corresponding translated Telugu text.

1.3 Motivation

Our project, aiming to create an English to Telugu transformer-based translation system, is fueled by the commitment to empower the Telugu-speaking community. By breaking language barriers, this initiative provides seamless access to global information, education, and diverse perspectives, allowing the Telugu community to express themselves on a broader stage.

This transformative tool not only translates words but preserves the cultural nuances of Telugu, fostering cross-cultural communication and understanding. Beyond convenience, it

becomes a vital resource for students, professionals, and enthusiasts, contributing to the preservation and revitalization of the Telugu language in the digital age.

In essence, our motivation is to build a bridge between languages, enhancing accessibility and fostering a more interconnected world where the richness of Telugu culture can thrive alongside global conversations.

1.4 Scope

Language-specific Challenges:

Address Telugu's linguistic nuances to ensure contextually accurate translations.

User Interface Design:

Develop an intuitive interface for seamless English to Telugu text translation.

Real-time Translation:

Strive for real-time translation capabilities to enhance user experience.

Cross-Platform Compatibility:

Ensure the system is compatible across various platforms and devices.

Scalability:

Design the system to be scalable for potential expansion to additional languages or model enhancements.

1.5 Literature Review

Survey - 1

Rule-Based Machine Translation (RBMT):

1. RBMT relies on explicit linguistic rules and grammatical structures to translate text from one language to another. These rules are typically created by linguists and translation experts and may involve syntax, semantics, and morphology.
2. RBMT systems heavily depend on linguistic knowledge and require extensive manual rule creation. Linguists need to encode language-specific rules and translation patterns, making the process labor-intensive.
3. RBMT systems may struggle with adapting to new language pairs or handling informal language. They are less flexible and may not capture the dynamic nature of languages in various contexts.

4. Handling polysemy and ambiguity can be challenging for RBMT systems. The rigid rule structures may not effectively navigate through multiple possible translations or diverse interpretations.
5. Updating or modifying an RBMT system requires manual intervention by linguistic experts. This makes these systems less adaptable to evolving languages and dynamic linguistic environments.

Survey - 2

Statistical Machine Translation (SMT):

1. SMT relies on a combination of rule-based and statistical methods. It involves the creation of linguistic rules and the estimation of statistical models based on bilingual corpora.
2. SMT often operates on a phrase-based translation model, where translation units are phrases or subphrases rather than individual words. These units are aligned across parallel corpora to learn translation probabilities.
3. SMT systems require extensive feature engineering, where linguistic experts manually design features and weights to capture translation patterns, word alignments, and other linguistic phenomena.
4. SMT may struggle to capture long-range dependencies and contextual nuances in language, as it typically relies on local context and may not consider the entire sentence during translation.

Survey - 3

LSTM (Long Short-Term Memory):

1. LSTMs are based on recurrent neural networks (RNNs) and process input sequences sequentially. They maintain hidden states that capture information from previous time steps, allowing them to model temporal dependencies.
2. LSTMs have hidden states and memory cells that enable them to capture and store information over long sequences. The memory cells help in mitigating the vanishing gradient problem associated with standard RNNs.

3. Computation in LSTMs is performed step by step, where the model updates its hidden states and memory cells at each time step based on the current input and the information stored in the previous hidden states and memory cells.
4. Due to their sequential nature, LSTMs face challenges in parallelizing computations, limiting their ability to efficiently utilize parallel processing capabilities of modern hardware.

Why Transformers?

1. Transformers leverage self-attention mechanisms, allowing the model to focus on different parts of the input sequence when generating each part of the output sequence. This attention mechanism captures dependencies effectively and enables the model to handle long-range dependencies.
2. Transformer-based machine translation follows an end-to-end learning approach. It learns representations directly from parallel corpora without explicit rule-based feature engineering, making the training process more data-driven.
3. The attention mechanisms in Transformers allow for efficient parallelization of training and inference, making them computationally efficient. This scalability is particularly advantageous for handling large datasets and accelerating translation processes.
4. Transformers excel at capturing contextual understanding, as they consider the entire context of a sentence during both training and inference. This enables them to generate translations that are more contextually relevant and coherent.
5. Transformer-based models can be fine-tuned with additional data, facilitating continuous learning and adaptation to evolving language patterns. This adaptability contributes to the system's ability to improve over time.
6. Unlike SMT, Transformers do not require explicit word alignment as they learn the alignment implicitly through the attention mechanism. This simplifies the training process and eliminates the need for separate alignment steps.

1.6 Problem Statement

Develop an efficient and accurate English to Telugu text translation system using deep learning techniques, specifically leveraging transformer architectures and to bridge language barriers, enhance communication, and accessibility by providing an advanced tool for seamless translation between English and Telugu.

PREPROCESSING

2.1 Data Preprocessing

2.1.1 Data Cleaning

decontractions(phrase):

This function takes a text phrase and expands English contractions into their full forms.

For instance, it replaces "won't" with "will not" and "can't" with "can not."

It covers various contraction patterns using regular expressions.

preprocess_english(text):

Converts all text to lowercase to ensure consistency.

Utilizes the decontractions function to expand contractions in the text.

Adds spaces around punctuation marks like question marks, periods, exclamation marks, and commas for better tokenization.

Removes extra spaces to clean up the text.

Eliminates special characters, keeping only letters and essential punctuation.

Appends '<start>' at the beginning and '<end>' at the end of the text to indicate the start and end of a sequence.

Strips any leading or trailing spaces to finalize the preprocessing.

In essence, these operations collectively prepare English text data for subsequent natural language processing tasks by addressing common language nuances and ensuring a standardized, cleaned-up format for analysis or model training.

2.2 Creating a vocab file from our corpus

Reading Data:

Reading text data from a specified file (path) and stores it in a list called data.

Creating TensorFlow Dataset:

Converting the list of text data (data) into a format suitable for TensorFlow processing.

Setting BERT Tokenizer Parameters:

It decides how the BERT tokenizer should handle the text, such as converting it to lowercase.

Reserved tokens like "[PAD]", "[UNK]", "[START]", and "[END]" are specified.

Defining BERT Vocabulary Parameters:

It sets up parameters for creating the vocabulary using BERT methods.

This includes specifying the target vocabulary size (150,000) and using the previously defined tokenizer parameters.

Creating Vocabulary:

Generating the vocabulary from the dataset using BERT vocabulary functions.

Writing Vocabulary to File:

Finally, saving the generated vocabulary to a file at the specified location (saving_path).

2.3 Custom Tokenizer

BERT tokenizer considers the bidirectional context of words, meaning it takes into account the surrounding words on both sides when encoding a word. This helps capture richer contextual information compared to traditional tokenization methods.

BERT tokenizer utilizes WordPiece tokenization, breaking down words into smaller subwords or pieces. This is particularly useful for handling rare words, out-of-vocabulary terms, and morphologically rich languages.

Typically, BERT tokenizers convert text to lowercase. This ensures consistency in representation, preventing the model from treating differently cased words as distinct entities.

BERT introduces special tokens like "[CLS]" (classification token), "[SEP]" (separator token), "[PAD]" (padding token), "[MASK]" (mask token), "[START]", and "[END]". These tokens serve specific purposes during model training and inference.

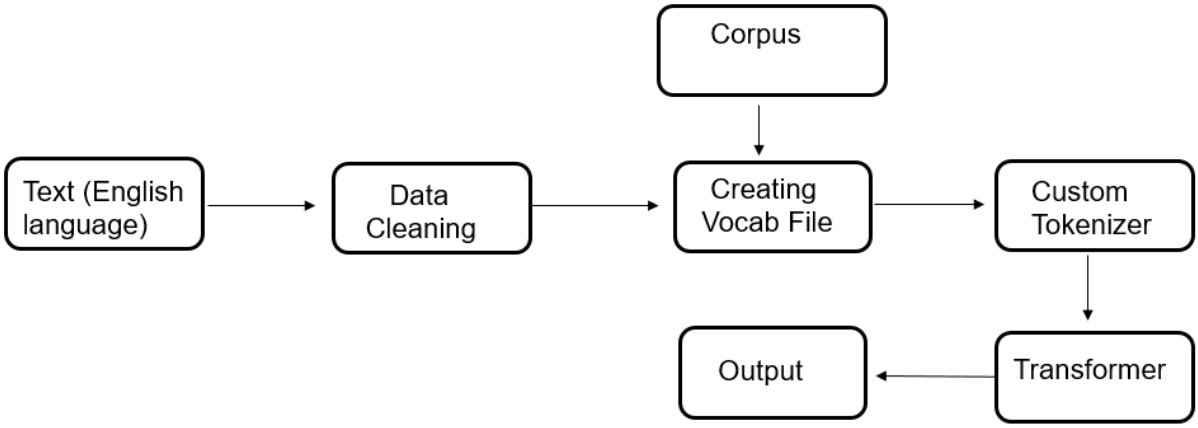
BERT tokenization is often applied in conjunction with pre-trained BERT models. The tokenizer is used to process input text, and the pre-trained model produces embeddings (contextualized representations) for each token.

During training, BERT utilizes a masked language model (MLM) objective. Some tokens in the input sequence are randomly replaced with the "[MASK]" token, and the model is trained to predict the original words based on the context.

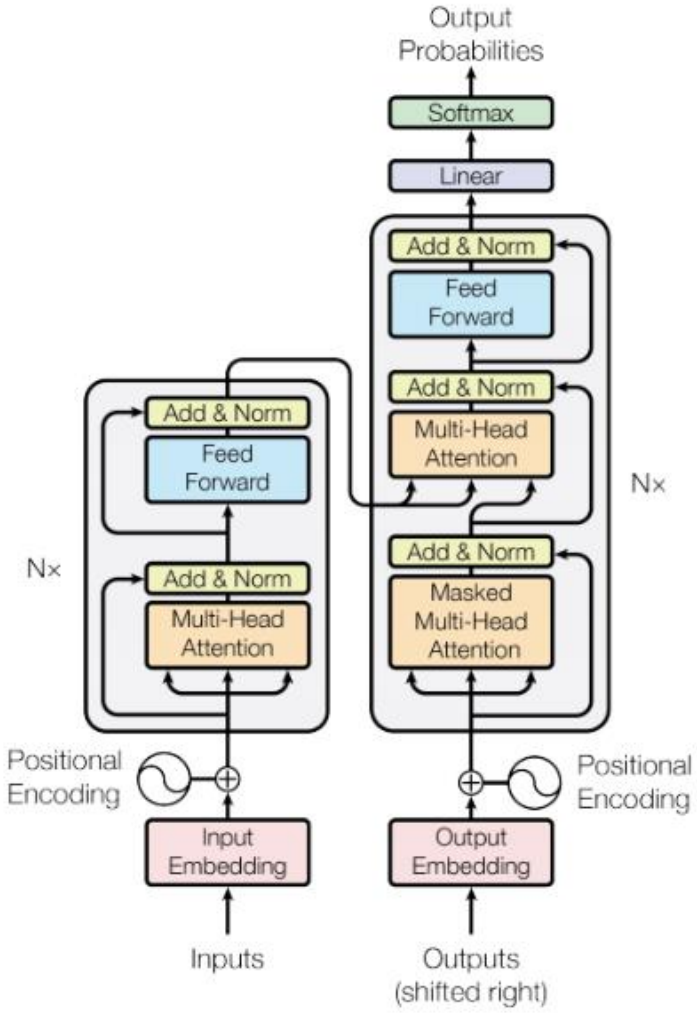
BERT tokenizer is widely used in various natural language processing (NLP) tasks, such as text classification, named entity recognition, sentiment analysis, and machine translation. It has become a foundational tool in the NLP landscape due to its ability to capture intricate language patterns.

OVERVIEW OF PROJECT DESIGN

3.1 Project Flow



3.2 Architecture of model



3.3 Software Requirements

1. Python
2. Pandas and Numpy
3. Tensorflow

3.4 Hardware Requirements

1. GPU - Training deep learning models, especially Transformer architectures, can be computationally intensive. Having a GPU accelerates the training process
2. RAM - 16GB or higher
3. CPU - A multi-core CPU, such as an Intel Core i7 or higher, is beneficial for parallel processing during training.

EXPERIMENTAL STUDY OF PROJECT

4.1 Algorithm for text to text translation using transformers

1. Regex To Convert Contractions Into Natural Form ,For example, won't to Will not
2. Custom Tokenizer Using Bert
3. Positional Encoding to give the model some information about the relative position of the words in the sentence
4. The Attention Function used by transformer takes three inputs q(query) k(key) v(value). used multi head attention consisting of four parts linear layer and split into heads scales dot product attention concatenation of heads final linear layer
5. Implementation Of Evaluation Metric BLEU

4.2 History of transformers

- **Introduction of Transformers (2017):**
  Vaswani et al. introduced transformers, revolutionizing sequence transduction tasks with self-attention mechanisms.
- **BERT Model (2018):**
  Devlin et al. presented BERT, pre-training transformers on vast unlabeled text data, achieving state-of-the-art NLP performance.
- **GPT-2 and GPT-3 Models (2019-2020):**
  OpenAI's GPT series, especially GPT-3, set new benchmarks in language understanding and generation.
- **Transformer-Based Machine Translation Models (2018 Onward):**
  Transformers became dominant in machine translation, with models like Transformer, BERT-based translation models, and M4 showcasing impressive results.
- **T5 Model (2019):**
  Google's T5 proposed a unified framework for NLP tasks, treating them as text-to-text, highlighting the versatility of transformers.
- **Ongoing Developments (2021 Onward):**
  Research continues in areas like multilingual models, domain adaptation, and training strategies to enhance transformer-based translation systems. In summary,

transformers have reshaped text-to-text translation, from their foundational introduction to the latest state-of-the-art models, driving advancements in natural language processing.

PARTIAL RESULTS & DISCUSSIONS

5.1 Data Pre-processing

5.1.1 Decontractions

```
def decontractions(phrase):
    """decontracted takes text and convert contractions into natural form.
     ref: https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python/47091490#47091490"""
    # specific
    phrase = re.sub(r"won\'t", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"won\’t", "will not", phrase)
    phrase = re.sub(r"can\’t", "can not", phrase)
```

This function takes a text phrase as input a nd expands English contractions into their full forms. It uses regular expressions to replace contraction patterns like "won't" with "will not" and "can't" with "can not," among others.

5.1.2 Text Preprocessing

```
def preprocess(text):
    # convert all the text into lower letters
    # use this function to remove the contractions: https://gist.github.com/anandborad/d410a49a493b56dace4f814ab5325bbd
    # remove all the spacial characters: except space ' '
    text = text.lower()
    text = decontractions(text)
    text = re.sub('[$-)\"’°;\'€%:(/]', '', text)
    # text = re.sub('[^A-Za-z0-9 ]+', '', text)
    text = text.strip()
    return text
```

Converts all text to lowercase.

Utilizes the decontractions function to expand contractions.

Adds spaces around punctuation marks (question marks, periods, exclamation marks, and commas).

Removes extra spaces.

Eliminates special characters except for letters and essential punctuation.

Strips any leading or trailing spaces.

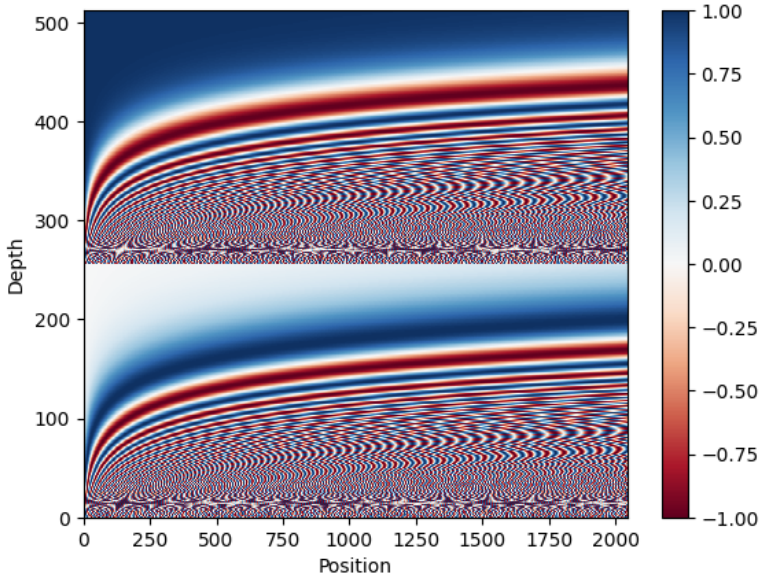
5.2 Positional Encoding

```
def get_angles(pos, i, d_model):
  angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model))
  return pos * angle_rates
```

```
def positional_encoding(position, d_model):
  angle_rads = get_angles(np.arange(position)[:, np.newaxis],
                          np.arange(d_model)[np.newaxis, :],
                          d_model)

  # apply sin to even indices in the array; 2i
  angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])

  # apply cos to odd indices in the array; 2i+1
  angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])

  pos_encoding = angle_rads[np.newaxis, ...]

  return tf.cast(pos_encoding, dtype=tf.float32)
```

```
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]

# Juggle the dimensions for the plot
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))

plt.pcolormesh(pos_encoding, cmap='RdBu')
plt.ylabel('Depth')
plt.xlabel('Position')
plt.colorbar()
plt.show()
```

```
(1, 2048, 512)
```



Since this model doesn't contain any recurrence or convolution, positional encoding is added to give the model some information about the relative position of the words in the sentence. The positional encoding vector is added to the embedding vector. Embedding represent a token in a d-dimensional space where tokens with similar meaning will be closer to each other. But the embedding do not encode the relative position of words in a sentence. So, after adding the positional encoding, words will be closer to each other based on the similarity of their meaning and their position in the sentence, in the d-dimensional space. The formula for calculating the positional encoding is as follows:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In conclusion, our progress in building a Transformer model for language translation is halfway with the successful implementation of positional encoding. The next crucial step our team works on involves incorporating scaled dot-product attention, a pivotal mechanism for the model's efficiency in capturing dependencies.

6.2 Future Work

Our team will be concentrating on enhancing the transformer model through the integration of multi-head attention. This step is crucial for improving the model's overall performance. We aim to finalize the transformer model by the end of December, ensuring it meets the desired standards of accuracy and efficiency.

January will be dedicated to the comprehensive evaluation of our transformer model. We will use various metrics(like BLEU) to assess its performance across different benchmarks.

Our final phase in February involves deploying the refined model into practical environments, ensuring seamless integration and real-world applicability.