Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 4 (2023)

Shattering Language Barriers: Singlish to English Translation with Transformer Neural Network

G. K. Nalinka, G. H. M. Iroshan, R. M. S. N. Rathnayake, G. M. N. Monali, D. I. De Silva, E. Weerasinghe

Dept. of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka.

Abstract:-This paper presents an innovative sequence-to-sequence machine translation system that leverages the state-of-the-art Transformer neural network architecture to translate sentences from Singlish to English. Notably, this marks the first Singlish-to-English machine translation system developed utilizing deep neural networks. The user-input sentence undergoes a systematic transformation, encompassing vectorization, positional embedding, and translation through the self-attention mechanisms, an innovation introduced by Google in 2017. Unlike dominant sequence transduction models reliant on intricate traditional recurrent or convolutional neural networks featuring encoders and decoders, the proposed model adopts the Transformer architecture, which relies exclusively on attention mechanisms. This innovative approach eschews the need for traditional recurrent and convolutional layers, offering enhanced translation quality, improved parallelization, and significantly reduced training time. The primary objective of this translator is to facilitate seamless translation, bridging the linguistic gap for both local and international users, thus dismantling language barriers. Impressively, the system demonstrates the capability to translate sentences containing over 20 tokens in less than one seconds. This achievement was made possible through the use of a minimal set of language rules and vocabulary for both the source and target languages.

Keywords*: Sinhala to English translator, English, Singlish, Sinhala, transformer neural network, self-attention mechanism.*

1. Introduction

Sinhala is a language with a rich history and cultural value that is mostly spoken in Sri Lanka. It is an essential part of the nation's social, political, and economic fabric and has over sixteen million native speakers. However, English, an Indo-European language with its roots in England, has become well-known as the world's lingua franca and is recognized as an official language in many nations. Sinhala and English are both Indo-European languages, despite their linguistic distinctions. The growing globalization and Sri Lanka's incorporation into the international community have created a need for accurate and contextually aware translation between Sinhala and English. English proficiency has become essential in several fields, including trade, diplomacy, academia, and technology [1].

Existing translation methods often fail to capture the subtleties of Sinhala expression, resulting in translations that may lack grammatical correctness and cultural relevance. The central problem this study addresses is the development of a translation system that can effectively convert Singlish sentences into grammatically sound English sentences while preserving the original meaning and cultural context. When translating from Singlish to English, translators' efficiency and accuracy are not satisfied. There are some limitations based on the inputs and translator; the sentence count is restricted.

The research aims to bridge the communication gap faced by many Sri Lankans who, despite proficiency in Sinhala, encounter challenges when expressing their ideas and perspectives in English. It responds to a pressing need for accurate and culturally sensitive translation tools in Sri Lanka, where proficiency in English is increasingly crucial for participation in the global economy and academia. It contributes to the broader field of natural language processing (NLP) and machine translation by exploring the complexities of translating between languages with distinct linguistic roots and cultural contexts. To facilitate effective communication and intercultural understanding, a system capable of translating Singlish into grammatically sound English is developed. Moreover, this research showcases the adaptability and versatility of sequence-to-sequence transformers in addressing the unique challenges posed by non-standard languages and dialects.

Developing a sequence-to-sequence machine translation model using Keras [2] utilizing a transformer neural network that has been specially designed to handle the complexities of Singlish-to-English translation is what this study's specific goals are. Deep neural networks called transformers replace CNNs and RNNs with self-attention [3]. Accordingly, neural networks for machine translation typically include an encoder that reads the input sentence and creates a representation of it.

An encoder reads the input language in neural networks for machine translation and creates a representation of it. The output sentence is then created word by word by a decoder while reviewing the encoder's representation. For each word, the Transformer first creates initial representations or embeddings. Then, employing self-attention, it compiles data from every other word, creating a new representation for each word that is influenced by the complete context, represented by the filled balls. Then, for each word, this phrase is repeated numerous times in parallel to produce new representations one after the other.

The decoder then uses the encoder's representation to create the output sentence word by word. For each word, the Transformer first creates initial representations or embeddings. Transformers may thus easily transport data across input sequences while collecting data from all the other words, creating a new representation for each word that is informed by the overall context, which is represented by the filled balls. step is then iterated over and again in parallel for each word to produce successively different representations [4]. Training and assessment of the model's ability to convert Sinhala sentences into grammatically sound English sentences. Examine the translations' cultural adequacy and contextual accuracy. Examine the system's practical applications in a variety of fields, including commerce, academia, and diplomacy, in the Sri Lankan setting.

Comparing the created sequence-to-sequence transformer model to current approaches will greatly increase the accuracy and cultural relevance of Sinhala-to-English translations. The technique will improve Sinhala speakers' ability to speak English fluently, especially in business and international settings. The system's implementation will benefit collaboration between English- and Sinhala-speaking individuals and cross-cultural understanding. This study will give a summary of the related work in the areas of machine translation and linguistic analysis that is pertinent to Singlish and English. The process that was utilized to create and train the Singlish-to-English translation model. Finally, a summary of contributions will be provided, the limitations of the study will be acknowledged, and avenues for future research in the domain of machine translation for culturally rich and diverse languages like Sinhala will be proposed.

2. Literature Review

Language translation has garnered considerable attention in an increasingly interconnected world, holding the potential to bridge linguistic gaps and facilitate cross-cultural interactions. Machine translation has emerged as a pivotal field of innovation within this landscape.

While translation tools exist for language families like Indo-European, Indo-Aryan, and Sino-Tibetan, the unique structural differences between Sinhala and English present challenges. As a result, creating effective translation systems for the Sinhala-English language pair has faced limitations.

Recent research has introduced a range of translation systems aimed at addressing these challenges. Particularly noteworthy is the Example-Based Machine Translation System, tailored for governmental use in Sri Lanka [5].

This system achieved remarkable accuracy in English-to-Sinhala translation, boasting BLEU scores [6] ranging from 0.17 to 0.26. These scores were determined through a 3-gram analysis with a single reference translation. The system's approach revolves around a bilingual corpus of English-Sinhala sentences, serving as its knowledge base. When presented with a source phrase, it retrieves English sentences along with their corresponding Sinhala sentences (Intra-Language Matching). Subsequently, a scoring algorithm is applied to the retrieved Sinhala sentences to identify the most frequently occurring Sinhala phrase, deemed the most likely translation candidate (Inter-Language Matching).

In another study [7], impressive accuracy in English-to-Sinhala translation was demonstrated, achieving an 89% success rate in tests with a diverse set of two hundred sentences. This system also exhibited commendable efficiency in morphological generation, effectively handling 85 grammar rules for Sinhala nouns and 36 for verbs. However, it faced challenges when confronted with more complex language elements, such as multi-word expressions, idioms, and compound sentences. Additionally, it grappled with limitations tied to its reliance on limited lexical resources.

Source [8] introduced a machine translation system tailored to convert grammatically accurate Sinhala sentences into English. It also featured valuable components like a built-in dictionary, a Sinhala based grammar checker, and more. Employing the Transfer-based machine translation approach, the study achieved a commendable 75% accuracy rate based on a meticulously selected set of 150 well-structured Sinhala sentences.

Furthermore, a rule-based machine translation system [9] stands out with bidirectional translation capabilities between Sinhala and English. This pioneering system includes unique features like a Sinhalese font translator and an English grammar checker. Users input translations by providing Sinhala in Singlish and English in English. This system aims to eliminate language barriers by providing smooth translations for both locals and foreigners. With an 87% accuracy rate, it translated 500 well-structured Sinhala sentences into English and 150 English sentences into Sinhala. It can process around 70 sentences per minute, showcasing its efficiency in bridging language gaps.

Moreover, another study [10] introduces a novel approach to addressing Sinhala-English translation challenges by leveraging Evolutionary Algorithms (EA). Unlike traditional methods, EA focuses on identifying the correct meaning of Sinhala text and subsequently translating it into English. Given the limited digital text available in Sinhala, EA proves to be a promising solution for deriving accurate translations. The methodology involves passing Sinhala text through the EA to discern its meaning and then executing the translation into English. The translated text undergoes a grammatical refinement process to ensure linguistic accuracy, showcasing promising outcomes in terms of translation precision. This innovative approach has the potential to overcome the challenges posed by the unique linguistic characteristics of Sinhala, opening new avenues for effective and accurate machine translation.

Existing literature consistently highlights the importance of accommodating diverse sentence structures. Various tools, such as the Unicode Converter [11] and the Google Transliteration IME [12], shed light on the complexity of language conversion, particularly the intricate transition from Singlish to Sinhala.

Comprehensive dictionaries, exemplified by the Madura dictionary [13], play a pivotal role in enriching the translation landscape. These dictionaries serve as invaluable resources, offering English meanings for Sinhala words and vice versa. Their contributions extend beyond mere translation, significantly enhancing language comprehension and overall usability.

Recent strides have introduced sophisticated models like the sequence-to-sequence Transformer, showcasing their proficiency in language translation tasks, including English-to-Spanish translation [2]. These models, grounded in the Transformer architecture, excel at handling sequential data, rendering them ideal for machine translation. They leverage attention mechanisms to heighten accuracy and fluency. The machine translation process involves several critical steps, encompassing text vectorization with the Keras Text Vectorization layer, the implementation of Transformer Encoder and Transformer Decoder layers within the Transformer architecture, meticulous data preparation (including tokenization), and the deployment of these trained models

for real-world translation tasks. In English-to-Spanish translation, these advanced models promise higher translation quality, encompassing the understanding of sentence structures, idiomatic expressions, and contextual nuances for linguistically accurate and contextually meaningful translations.

ChatGPT [14], a leading conversational AI model also based on the Transformer architecture, processes natural language sequentially, capturing word relationships through self-attention mechanisms. In language translation, it efficiently uses an encoder-decoder framework. During pre-training on diverse internet texts, the model predicts the next word, acquiring a broad understanding of language. Incorporating natural language processing techniques like lemmatization and stemming enhances its ability to handle language variations. Lemmatization reduces words to their base form, refining comprehension while stemming captures common linguistic roots. In fine-tuning, ChatGPT adapts to tasks like language translation, excelling in generating context-aware responses. Ongoing research and improvements, including curriculum learning and reinforcement learning, drive the continuous evolution of models like ChatGPT, enhancing their versatility in conversational AI systems.

3. Methodology

A sequence-to-sequence transformer neural network model [15] implemented with Keras [2] was used to design the Singlish-to-English language translator. The system uses encapsulation to hide its complexity from the users. The entire task is done on behalf of the user with a single button click. What the user wants to know is what the input format is that the user should enter, and which button should be clicked to get the output. To address this concern, the input format for Singlish sentences was defined, utilizing [11], which effectively converts Sinhala sentences into their corresponding English text. The work was carried out using the Google Co-Lab environment and the Python Flask framework for building the interface. The Singlish-to-English language translator consists of five main categories:

1. Convert active voice from Singlish to English.
2. Transform passive voice from Singlish to English.
3. Change interrogative Singlish to English.
4. Adapting non-living subject Singlish sentences into relevant English sentences.
5. Convert double-entendre Singlish to English.

The Transformer architecture, introduced in the ‘Attention is all you need’ paper [15], does not appear to have been written with the intent of serving as the foundation for Bert GPT or language models. The emphasis in this paper has been less on the architecture itself and more on achieving the specific task of language translation. In this paper, attention will also be directed toward the same architecture for language translation from a language called Sinhala to a language called English.

The transformer architecture is comprised of two parts: an encoder and a decoder. During training, Singlish words from the sentence are simultaneously taken by the encoder, and word vectors are generated concurrently. Word vectors are eventually used as context before continuing to train the Transformer, owing to the attention mechanism. The decoder handles English words simultaneously while also receiving a start token to indicate the beginning of the sentence and an end token to signify the sentence's conclusion.

The Singlish vectors that were generated by the encoder are fed into the decoder, and the translation is shifted to the left by the forward-labeled output of the decoder.

In the context of translating the Singlish to English. A batch size of 30 is assumed, whereby 30 sentences are passed once through the network simultaneously to update the weights of the entire network at once. It is noted that both languages have a shared maximum sentence length of 50 words.

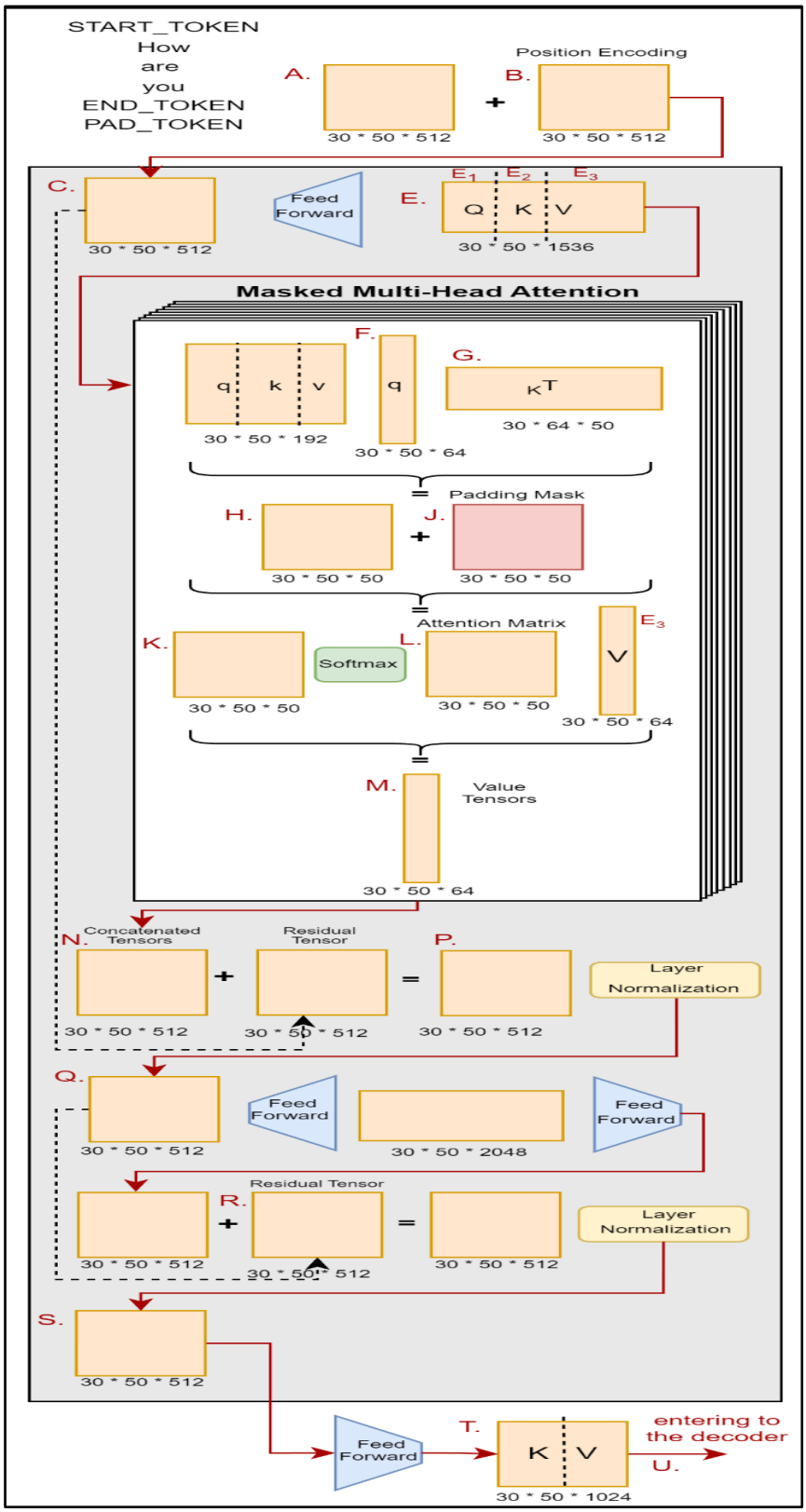

Fig. 1 The Encoder of the transformer

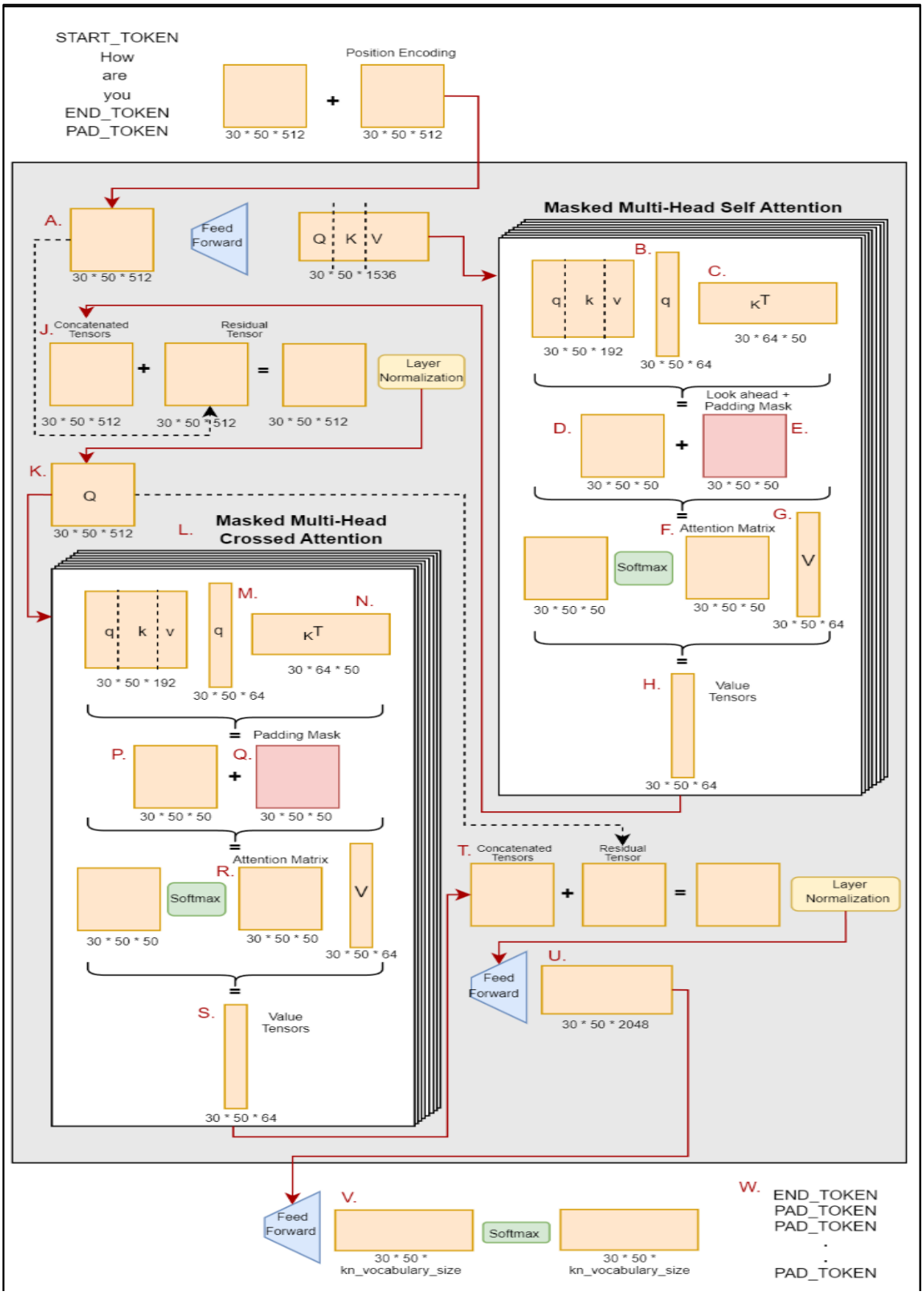

Fig. 2 The Decoder of transformer

The words are transformed into embeddings; specifically, embeddings are represented as vectors, each consisting of 512 dimensions. The cube structure, denoted as (A in "Fig. 1"), is essentially a large tensor that will be of size 30*50*512. (Includes a batch dimension of 30, a word count of 50 words in the sentence, and 512 dimensions for each word representation.) Then a positional encoding of the same shape is added (B in "Fig.1").

These positional encodings are generated using sine and cosine functions involving numbers ranging from -1 to +1. The existence of positional encoding is due to the simultaneous intake of words by the encoder, where the order of these words holds significance. The ordering, or positional encoding, is defined by a positional encoder. Subsequently, the final tensor (C in "Fig. 1") is obtained after positional embeddings and passed through a feedforward network to derive Query vectors, Key vectors, and Value vectors. Each word generates three vectors: a Query vector, a Key vector, and a Value vector.

Every word is represented by three vectors. The 512-dimensional word vector is converted into a 512-times-three matrix, resulting in a 1536-dimensional representation. Then a breakdown is undertaken to facilitate multi-head attention. A set of large, stacked white blocks denoted as (D in "Fig. 1") has been implemented according to transformer architecture. Eight of these stacked blocks are placed on top of each other, demonstrating extensive multi-head self-attention.

Self-attention is employed to analyze the context and establish context within the same sentence. Self-attention is now transformed into a multi-head structure with eight layers. Additionally, it has been masked because many sentences do not have a length of 50 words, and as a result, padding tokens need to be added. The logic observed within one grid (D in "Fig. 1") is intended for a single attention head, and similarly, eight parallel processes are concurrently conducted.

The 1536-dimensional word (E in "Fig. 1") vector for every single word is going to be divided into eight pieces. The query part (E1 in "Fig. 1"), key part (E2 in "Fig. 1"), and value part (E3 in "Fig. 1") are divided into eight pieces as well. Each piece is essentially transformed into a 64-dimensional vector. The 64 dimensions of the Query, Key, and Value are then stacked in a certain way to obtain a 192-dimensional vector representing every word for one head.

Then the query vector (F in "Fig. 1") is taken along with the key vector (G in "Fig. 1") and multiplied by them. (i.e., 30*50*64 and 30*50*64). After the matrix multiplication is completed, a 30*50*50 matrix (H in "Fig. 1") is obtained. It is observed that every word in a Singlish sentence that constitutes the Query vector interacts with every word in the same Singlish sentence's Key vectors, forming a self-attention matrix.

According to architecture, the attention matrix should be scaled and masked. Typically, before the padding mask is added (J in "Fig. 1") some sort of scaling is performed to prevent values there (H in "Fig. 1") from experiencing excessive multiplication, resulting in either excessively high or very low numbers. This scaling also serves to stabilize the training process.

Scaling simply means dividing every value in the self-attention tensor by a constant value. The value that has been used in the main paper [15] is the square root of the key dimension size vector for one head. It ensures that the activation values are neither too large nor too small.

After scaling, the padding mask should be added. A padding mask will prevent the padding tokens from propagating values. After applying the padding mask (K in "Fig. 1"), the SoftMax activation is applied, and then an attention matrix is generated (L in "Fig. 1"). The attention matrix is a *30*number of words*number of words* matrix and serves as a probability distribution for each row. Each value in the matrix quantifies how much attention should be paid by each word to every other word.

Subsequently, the Value matrix computed at the beginning (E3 in "Fig. 1") is applied to obtain 30*50*64-dimensional value tensors, denoted as (M in "Fig. 1") (output of just one attention head). These value tensors exhibit high contextual awareness. The output comprises not only a single attention head but also includes eight attention heads. Upon concatenating them sequentially, a total of 64 times 8 of these value tensors (N in "Fig.

1") are generated, resulting in 512 value tensors. The concatenated tensor is now a highly contextually aware tensor.

Now, a residual tensor is introduced. During the process of backpropagation, the loss value is propagated in a backward direction to update the weights. The most significant changes are observed towards the end of the network. Skip connections or residual connections assist in improving the propagation of inputs in the forward direction and the loss in the backward direction. It has been observed extensively in research involving deep convolutional neural networks, as demonstrated in [16].

After the addition of the residual tensor, a new tensor is obtained (P in "Fig. 1") which is expected to carry out the activations and the weight updates. Subsequently, layer normalization is performed. The objective of normalization is to ensure stable training, ensuring that activations during the forward phase and gradient updates during the backpropagation phase do not exhibit excessive magnitudes. Mathematically, normalization involves subtracting the values from the mean and dividing them by the standard deviation.

In batch normalization, values are normalized across the batch, which has a dimension of 30. However, in layer normalization, values are normalized across the feature layer, which has a dimension of 512. For layer normalization, each value of the tensor is subtracted from the layer mean and divided by the layer standard deviation.

Now, the output tensor (Q in "Fig. 1") is taken and passed through a feedforward layer. It is then passed back through another feedforward layer to capture additional information. Next, the same addition of the residual tensor (R in "Fig. 1") and layer normalization is performed to finally obtain a set of 512-dimensional tensors for every single word (S in "Fig. 1") (S is the output of the entire encoder architecture). Each of them will be highly contextually aware. Then, the contextually aware items are going to be passed through a feedforward network, and Key and Value vectors will be extracted (T in "Fig. 1").

For decoder input, sentences will be passed with a start token followed by the sentence values, an end token, and then a bunch of padding tokens following the same procedure as before. Then positional encoding will be added. Once the positional encoding has been added, the tensor (A in "Fig. 2") will be passed through a feedforward network to obtain Query, Key, and Value vectors. Next, Mass multi-head self-attention will be applied to the same dimensions. The Query, Key, and Value vectors will be split into 8 different heads, and the Query (B in "Fig. 2") will be multiplied by the Key (C in "Fig. 2") tensor. An attention matrix will then be created (D in "Fig. 2"). That tensor will be scaled before a mask is added.

However, in this case, there is no need to add a padding mask. A look-ahead mask also needs to be added. The look-ahead mask will ensure that the decoder is not cheating during the generation phase. The decoder is required to start translating sentences without having access to future English words or future target language words.

Therefore, a mask needs to be applied during training to ensure that it does not look ahead of its current self. The third word in the sentence cannot be considered the fourth word to determine what it can attend to. Any contextual information cannot be derived during the training phase.

The look-ahead mask will be added along with the padding mask (E in "Fig. 2"). Then, SoftMax will be applied to obtain attention values and probability distribution-like values for every single word, indicating how much attention it needs to pay to every other word in that sentence. Then the attention matrix (F in "Fig. 2") will be multiplied by the value matrix. (G in "Fig. 2"). Then a tensor will be obtained (H in "Fig. 2"), which can be concatenated across all eight batches to get the final concatenated tensor (J in "Fig. 2"). Next, a residual tensor is added to ensure that information will be propagated, and layer normalization is applied.

Now the batch is obtained (K in "Fig. 2"). Referred to as 'Q,' a set of Query tensors will now be used as input for the Mass Multi-Head Cross-Attention layer (L in "Fig. 2") to perform cross-attention, which involves connecting every word in the target English sentence to every source word in the Singlish sentence. The Query represents essentially "What am I looking for?" That is kind of what would want to output. The English words

will be referred to as the Query tensors. An arrow is observed (U in "Fig.1"), indicating its origin from the encoder component, which supplies concatenated tensors (S in "Fig. 1").

A feed-forward network will be employed to map 512-dimensional tensors (S in "Fig. 1") to 1024-dimensions. Each word will be represented by 512 and 512-dimension Key and Value tensors (T in "Fig. 1"). The information encoded in Singlish (T in "Fig. 1") will be incorporated into the English vectors. Subsequently, an appropriate English translation will be generated based on the information provided by the Singlish translation. The Query will originate from the English sentence, while the Key and Value vectors will be derived from the Singlish sentence. Multi-head cross-attention will then be performed. Resembling the self-attention mechanism was observed in other cases. But here, clearly, the source of the Query is different, and the Key and Values are different as well.

To obtain an attention matrix (P in "Fig. 2") the Query (M in "Fig. 2") and Key (N in "Fig. 2") vectors are multiplied, which will be scaled as before to ensure numerical stability. A padding mask (Q in "Fig. 2") is also needed. Every single English word is permitted to be exposed to the entire Singlish word sentence. Because, during the translation phase, everything is contained within the encoder and all of the Singlish words have already been translated. A padding mask needs to be added solely to zero out any padding information from excessive tokens. Afterwards, a SoftMax operation is performed to obtain a probability distribution of how much attention each English word should be paid to a Singlish context (R in "Fig. 2").

Subsequently, similar value tensors (S in "Fig. 2") are obtained by concatenating them across the eight heads. Results in a 512-dimensional vector for every single English word (T in "Fig. 2"). Each English word now has some Singlish context embedded in it as information. A residual tensor is added to ensure that extra propagated information is present throughout the network because it is a very deep network. After performing some layer normalization to stabilize the values and gradients, the feed-forward layer will end up with a 512-dimensional tensor (U in "Fig. 2").

The final English tensors, which have Singlish context embedded, can be passed into a feedforward layer to expand the size of the English vocabulary. The vocabulary represents the number of possible words that can be seen and predicted by the model. Transformer architecture has been designed for predicting words.

In this case, a vector of the size of the English dictionary will be generated for every single batch and every single word. After applying a SoftMax function to it (V in "Fig. 2"), a probability distribution across all English words is obtained. To select the word most likely to align with the prediction, it will be compared with the labels (W in "Fig. 2"). Based on those labels and predictions, a cross-entropy loss will be computed, and then backpropagation will be performed throughout the network.

The methodology comprises several key steps, including data preprocessing, model architecture, training, deployment, and the development of a user-friendly web interface for translation. The following subsections provide a comprehensive explanation of each step.

3.1. Data Collection and Preprocessing

3.1.2. Data Collection

The foundation of any machine translation model is the quality and quantity of the training data. In this research, a Singlish-to-English parallel corpus for training was acquired. The data collection process involved obtaining a Singlish text corpus and aligning it with its corresponding English translation. The Singlish text corpus was sourced from a file containing Singlish-English sentence pairs separated by a period.

3.1.2. Data Preprocessing

3.1.2.1. Sentence Segmentation

Sentences were split using the period (".") as a delimiter to extract both the Singlish and English sentences.The Singlish sentence was stripped of leading and trailing spaces.The English sentence was preprocessed by adding

special tokens "[start]" and "[end]" to denote the beginning and end of the sentence, respectively.The processed sentence pairs were stored as (Singlish, and English) tuples.

The dataset was then randomly shuffled to ensure an unbiased distribution. A portion of the dataset was set aside for validation and testing, with approximately 15% of the data used for validation and the rest for training.

3.2. Text Vectorization

To input the textual data into the model, text vectorization should be employed. Two separate Text Vectorization layers were used for the Singlish and English text:

3.2.1. Source Vectorization (Singlish)

This layer tokenizes and converts Singlish sentences into sequences of integers. It has the following characteristics:

- Maximum vocabulary size is 500.
- Output mode is set to ‘int’ for integer sequence output.
- Output sequence length set to 50 to limit sequence length.

3.2.2. Target Vectorization (English)

This layer performs similar tokenization and integer sequence conversions for English sentences. It includes additional standardization to remove punctuation and special tokens. The characteristics are:

- Maximum vocabulary size is 500.
- Output mode is set to ‘int’ for integer sequence output.
- Output sequence length is set to 51 (one token longer than the source) for sequence prediction.
- Both vectorization layers were adapted to the training data to ensure consistent tokenization.

3.2.2.1. Custom Standardization

To standardize and preprocess English text, a custom standardization function based on [17] was utilized. This function converts text to lowercase.

3.3. Model Architecture

3.3.1. Transformer Architecture

The model is built on the Transformer architecture, a groundbreaking neural network architecture introduced in ‘Attention Is All You Need’[15]. This architecture has revolutionized sequence-to-sequence tasks, including machine translation. The model comprises two main components:

1. Transformer Encoder
2. Transformer Decoder

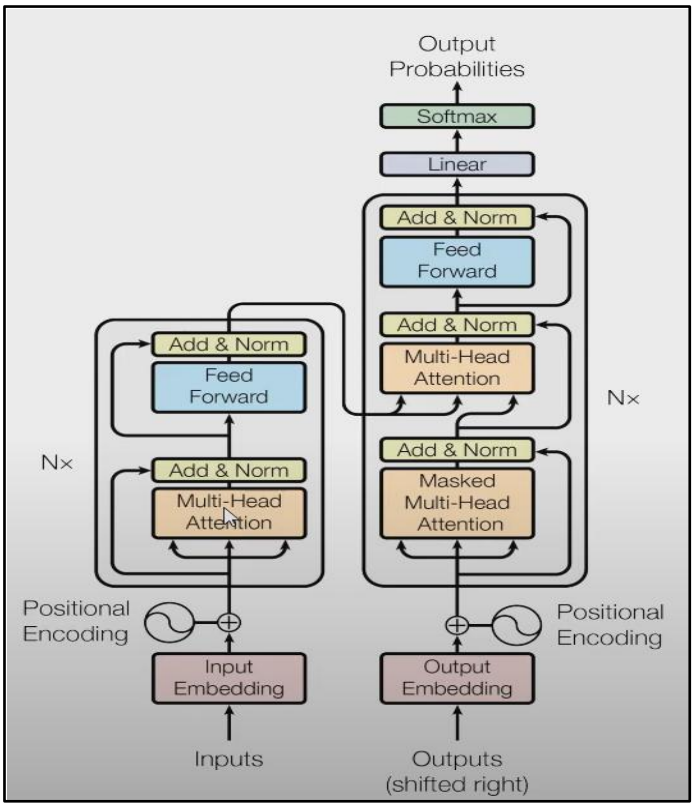

Fig. 3 The transformer neural network architecture [18]

3.3.2. Transformer Encoder

With a transformer encoder, it is possible to pass all the words of the Sentence (which were previously vectorized) simultaneously to determine word embeddings. The encoder's primary function is to capture meaningful representations of the source language sentences. The Transformer Encoder is responsible for encoding the input Singlish sentences. It consists of the following components:

3.3.2.1. Multi-Head Self-Attention

The concept of multi-head self-attention is fundamental to the Transformer Encoder. This mechanism, as described in [15], enables the model to capture contextual information effectively by attending to different parts of the input sequence simultaneously. It involves answering what part of the input should be focused on.

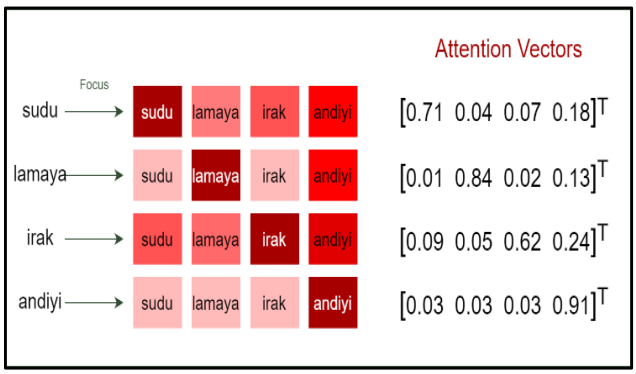

Fig. 4 The attention vector

When translating Singlish to English, it is necessary to perform self-attention, which involves paying attention to oneself. The relevance of the 'i^{th}' word in a Sinhala sentence to other words in the same Sinhala sentence is computed within the attention block. An attention vector is generated for every word, capturing contextual relationships between words in the same sentence.

By using attention every word can have its vector better incorporate the context both before and after it. Essentially, each input word to Transformer will be associated with three vectors: a Query vector, indicating 'What am I looking for'; a Key vector, specifying 'What can I offer'; and a Value vector, denoting 'What I actually offer.'

Query (Q), Key (K), and Value (V) are abstract vectors that extract different components of an input word. Q, K, and V vectors are computed for each word, and these vectors are used to calculate the attention vectors for every word using the following formula (1).

$$self\ attention = softmax\left(\frac{Q.K^T}{\sqrt{d_k}} + M\right)v \quad (1)$$

'T' denotes transpose , and 'd_k' signifies dimensionalKey vector

This product ($Q.K^T$) results in the generation of an input sequence length by input sequence length matrix. The extent of its proportionality is determined by the level of attention that is wished to be allocated to each word. The square root of the dimension of Q and K is employed to reduce variance and stabilize the values within the Q.K transpose vector.

An input sentence is transformed into an embedding to convey its meaning and a positional vector is added to contextualize each word within the sentence. Subsequently, the attention block computes attention vectors for each word. While it is true that the attention vector for each word can be strong, it becomes less valuable when the word heavily weights its relationship with itself.

Attention vectors exhibit a higher interest in interacting with different words and are utilized to derive approximately 8 such vectors per word. These vectors are then subjected to a weighted averaging process to calculate the final attention vector for each word. Since it has been used multiple attention vectors, name it as multi-headed attention block.

The word vector is located here (A in "Fig.5 "). It could be considered one of the words in the sentence "sudulamayairakandiyi." The net vector for the word 'lamaya' could be introduced by it. It is a 512-dimensional vector that is broken down into three component vectors. Each word is assigned a Query Vector, a key vector, and a Value vector. Each vector is divided into eight parts (B in "Fig.5"), and each part contributes to the creation of an attention head.

There are eight attention heads in total. Each attention unit then processes one, along with other words as well. All other words in the sentence are broken down in a very similar way and passed to an attention unit. For each head, an attention matrix is generated (C in "Fig.5 "), which is a sequence of the same length as the input sequence. Each row of these matrices sums up to one because they represent a probability distribution.

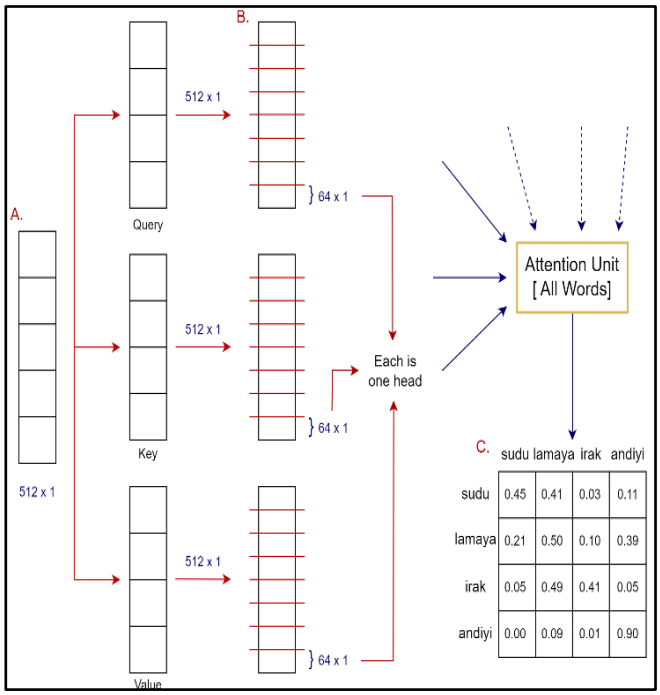

Fig. 5 The Multi-Headed Attention

There will be eight such attention matrices because there are eight attention heads in this multi-headed attention system. Then, other output vectors are generated, which are concatenated in order to produce vectors that possess a high level of contextual awareness.

3.3.2.2. Dense Projections and Layer Normalization

The use of dense projections and layer normalization after each sub-layer in the Transformer Encoder is in line with best practices for stabilizing training, as detailed in [15].

Batch normalization is typically applied to smooth out the loss surface, making it easier to optimize when using larger learning rates. However, layer normalization can be employed, which normalizes across each feature instead of each sample, offering better stabilization.

The attention networks are processed through a feedforward network one vector at a time. Each attention network operates independently, allowing for efficient parallelization. This facilitates the simultaneous processing of all words within the encoder block, resulting in a set of encoded vectors for each word.

3.3.3. Transformer Decoder

During the training phase for Singlish to English, it is necessary to feed output English sentences to the decoder. Computers do not get languages they get numbers vectors and matrices. Therefore, before process input embedding is required to get the vector form of the word. Then it is need to add a positional vector to get the position of context of the word in a sentence. Finally, pass the vector into a decoder block.

The Transformer Decoder takes the encoded Singlish sentences and generates the target English sentences. It includes the following components:

3.3.3.1. Causal Attention Mask

The inclusion of a causal attention mask in the Transformer Decoder, inspired by [15], ensures that during decoding, each token attends only to previous tokens, preventing information leakage from the future. This mechanism is critical for autoregressive sequence generation.

3.3.3.2. Multi-Head Self-Attention (Decoder Input)

Multi-head self-attention for the decoder input allows the model to attend to itself while generating the output sequence, as explained in [15].

3.3.3.3. Multi-Head Attention (Encoder-Decoder Attention)

The multi-head attention mechanism between the encoder and decoder, as outlined in [15], enables the model to focus on relevant source information during the decoding process.

Mask multi-headed attention block of decoder generates attention vectors for each word in the English sentence to represent how much each word is related to every word in the same sentence.These attention vectors and vectors from the encoder are passed into another attention block.

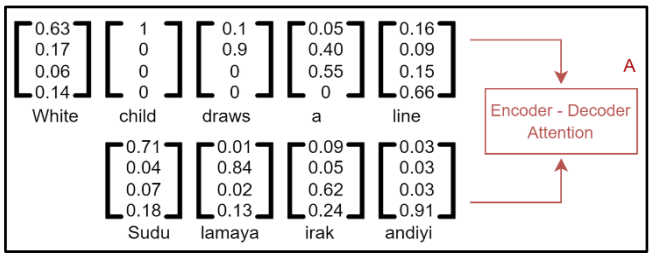

Fig. 6 The encoder-decoder Attention

Overall, the next word is predicted by the decoder, and execution occurs over multiple time steps until the end of the sentence token is generated.

Since each word in the English and Singlish sentences is represented by a vector, the determination of the degree of relatedness between each word vector with respect to each other is accomplished by the attention block (A in "Fig.6"), where the primary Singlish-to-English word mapping takes place. The output of this block consists of attention vectors for every word in the English and Singlish sentences, with each vector representing the relationships with other words in both languages.

3.3.3.4. Dense Projections and Layer Normalization

Similar to the encoder, two dense layers project the attention output to the desired embedding dimension. Layer normalization is applied after each sub-layer to stabilize training. A dense layer with a Softmax activation function produces the output probabilities for each token in the vocabulary.

3.3.4. Positional Embedding

To provide positional information to the model, as described in[15], plays a crucial role in providing the model with information about the sequential order of tokens within sentences. A Positional Embedding layer is applied to both the encoder and decoder input sequences.

3.3.4.1. Input Embedding/Token embedding

The idea is to map every word to a point in space where similar words and meanings are physically closer to each other. The space in which they are present is called embedding space. Embedding space map, a word to a vector. The concept involves the mapping of each word to a point in space where similar words and meanings are physically closer to each other. The space in which they are located is referred to as the embedding space, where a word is mapped to a vector.

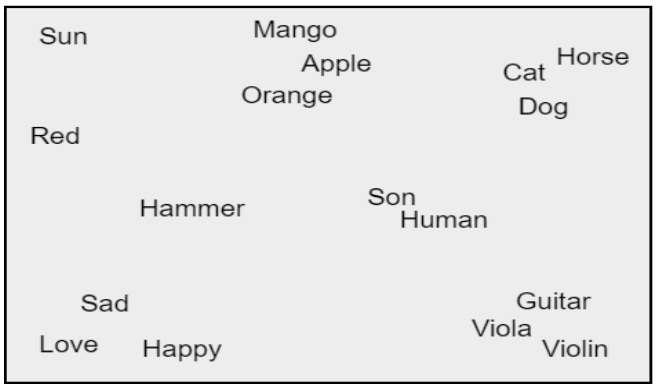

Fig. 7 Input Embedding

But the same words in different sentences may have different meanings. Positional encoders come into play.

Positional Encoder provides a context-based vector indicating the position of words in a sentence. It is a vector that contains information about the distances between words and the sentence. Sin and cosine functions are employed to generate this vector (A in "Fig.8"). After the Singlish sentence is passed through the input embedding and positional encoding is applied, a vector with positional information is obtained, which represents the context.

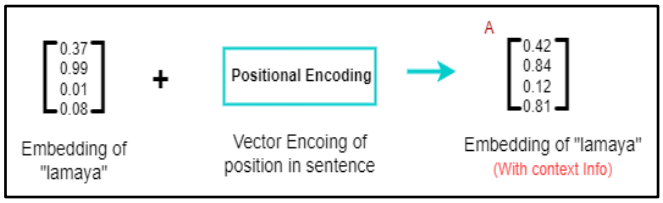

Fig. 8 Position-encoded vector

3.3.4.2. Positional Encoding

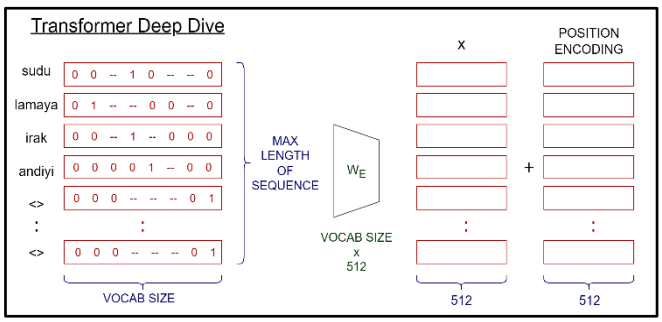

Fig. 9 Positional Encoding

Transformer architectures will generate word pieces, sub-words, or byte pair encodings, which are similar to broken-down versions of words rather than complete words. However, in this research paper, a word-level language model is employed.

First, the sentence intended for input in Singlish is, 'sudulamayairakandiyi.' To ensure a consistently passed fixed-length matrix, the remainder of words not present is padded with a dummy character or dummy sequence input (the maximum length of the sequence is determined by the maximum allowable number of words in the transformer). Each of these words is then one-hot encoded. The vocabulary size represents the number of words in the dictionary, i.e., the number of possible input words.Next, the data is passed into the feed-forward layer, where each vector is mapped to a 512-dimensional vector, and the parameters are learned through backpropagation. The number of parameters to be learned is the vocabulary size multiplied by 512. The output

consists of a set of 512-dimensional vectors, one for each input in the sequence. Additionally, positional encoding of the same size is added to the output.

The formulas for computing positional encoding as (2) and (3) are presented here.

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{2i/d_{model}}}) \quad (2)$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d_{model}}}) \quad (3)$$

'*pos'* represents the position of the word in the sequence, '*i'* denotes the index of the dimension, and 'd_{model}' signifies the dimensional length, which is set at 512. The rationale behind the formulation of positional embedding in this manner is now being explored.

3.4. Dataset Preparation

The dataset preparation pipeline follows the best practices outlined in [19], which include batching, parallelization, and prefetching to optimize data loading and processing. To train the model, it has been created a dataset pipeline using TensorFlow. The pipeline involves the following steps:

- Data pairs (Singlish and English) were formatted using the previously defined source and target vectorization layers.
- Batches of data were created, each containing a fixed number of samples (*batch_size = 50*).
- Parallelization and prefetching were used to optimize data loading and processing.

3.5. Model Training

3.5.1. Optimizer and Loss Function

The choice of the RMSprop optimizer and sparse categorical cross-entropy loss function aligns with established practices for training sequence-to-sequence models [20] [21].

3.5.2. Training Epochs

The decision to train the model for 60 epochs is based on empirical experimentation and convergence analysis. During this, the model learned to map Singlish sentences to English sentences. The validation dataset was used to monitor model performance and prevent overfitting.

3.6. Model Deployment

The trained model was deployed for practical use through a web-based interface. The deployment involved the following steps:

3.6.1. Google Drive Integration and Flask web application with Ngrok Integration

The model was saved to Google Drive for easy access and sharing, allowing for easy version control, and sharing. The development of a Flask web application for model deployment is inspired by the framework's simplicity and flexibility [22]. It provides a user-friendly interface for translation. Ngrok was selected for exposing the Flask app to the internet due to its ease of use and reliability [23]. It ensures that the translation service is accessible online.

3.6.2. Loading the Model

The model was loaded using TensorFlow's 'saved_model.load' function.

3.7. User-Friendly Web Interface

The user-friendly web interface design and functionality draw inspiration from best practices in web development. It aims to provide a seamless and intuitive experience for users seeking translations. The web

interface enables users to input Singlish sentences and receive English translations. It includes the following features:

- Input Form: Users can enter Singlish sentences for translation.
- Translation: The model generates English translations and displays them in the output text area.
- Exit Option: Users can terminate the translation session by entering "exit."

3.8. Evaluation

3.8.1. Evaluation Metrics

The choice of evaluation metrics, including BLEU score, METEOR, and ROUGE follows industry standards for assessing machine translation model performance [6]. These metrics assess the quality of the generated translations by comparing them to reference translations.

4. Results

Acquiring knowledge of and improving sequence-to-sequence Transformer models relies heavily on performing statistical analyses and visualizations. Loss curves and learning curves help evaluate model convergence and identify potential overfitting, while attention heatmaps reveal where the model concentrates its attention during sequence generation. Sequence length analysis exposes how the model handles varying input and output lengths, while BLEU and ROUGE scores provide quantitative evaluation metrics. Word frequency distributions and error analysis can both be used to identify weak points in a model. Using the right statistical tests and comparisons with baselines provides context for performance increases. ROC and Precision-Recall curves are helpful for classification tasks while embedding visualizations provide information about word representations [4].

5. Discussion

Sequence-to-sequence Despite being quite good at many different tasks involving natural language processing, transformer models have many difficulties. The model needs a large amount of parallel data set for training and takes a large amount of computer resources [2]. When getting efficient output from this model, it could be used for a long duration. Due to the quadratic complexity, overfitting can happen with small datasets, and they may have trouble with very long sequences. The Sinhala language has various kinds of patterns of character, words, and complex structured approaches in that situation to provide accurate output from using all things in Sinhala which is a much more complex task. Sinhala is spread over a large area. Furthermore, extending these models to multimodal data and coping with lengthy training times add to the complexity of working with sequence-to-sequence Transformers.

6. Conclusion

The study provides a thorough methodology for creating and using a deep neural network sequence-to-sequence transformer model-based Singlish-to-English translator, which helps remove the language barrier between Sinhala and English. The system consists of an English present, past, and future tense translator as well as a Singlish-to-English language translator that can translate sentences in active voice, passive voice, and questionnaire sentences. The first sequence transduction model is based solely on attention, using multi-headed self-attention instead of the conventional recurrent layers in encoder-decoder designs. In the Singlish-to-English translation challenge, the intention is to increase the training dataset from 20,000 to 100,000 sentences to enhance the model's accuracy. It has been demonstrated that expanding the dataset has a considerable positive impact on the model's performance. Anticipate that the model will produce more precise and trustworthy translation results as the data is scaled up. Research findings indicate that this Singlish-to-English translator can create reliable translations that are pertinent to the situation, which will help language learners and those who want to interact more successfully in a global setting.

References

[1] D. De Silva, A. Alahakoon, I. Udayangani, V. Kumara, D. Kolonnage, H. Perera and S. Thelijjagoda, "Sinhala to English Language Translator," in *4th nternational Conference on Information and Automation for Sustainability*, Colombo, 2008.

[2] F. Chollet, "Keras documentation: English-to-Spanish translation with a sequence-to-sequence Transformer," 26 May 2021. [Online]. Available: https://keras.io/examples/nlp/neural_machine_translation_with_transformer/.

[3] "Machine Learning Glossary," [Online]. Available: https://developers.google.com/machine-learning/glossary.

[4] "Neural machine translation with a Transformer and Keras," [Online]. Available: https://www.tensorflow.org/text/tutorials/transformer.

[5] R. Weerasinghe and A. M. Silva, "Example based machine translation for english-sinhala translations.," in *Proceedings of the 09th International IT Conference*, 2008.

[6] K. Papineni, S. Roukos, T. Ward and W. J. Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, Philadelphia, 2002.

[7] B. Hettige and A. S. Karunananda, "Computational tra of grammar for english to sinhala machine translation.," in *2011 International Conference on Advances in ICT for Emerging Regions (ICTer)*, Colombo, 2011.

[8] D. I. De Silva, P. K. D. A. Alahakoon, P. V. I. Udayangani, D. Kolonnage, M. H. P. Perera and S. Thelijjagoda, "Application of Transfer based Machine Translations from Sinhala to English," in *4th SLIIT Research Symposium*, Malabe, 2008.

[9] L. Wijerathna, W.L.S.L. Somaweera, S. L. Kaduruwana, Y. V. Wijesinghe, D. I. De Silva, K. Pulasinghe and S. Thellijjagoda, "A translator from sinhala to english and english to sinhala (sees).," in *International Conference on Advances in ICT for Emerging Regions (ICTer2012)*, Colombo, 2012.

[10] A. Nugaliyadde, J. K. Joseph, W. T. Chathurika and Y. Mallawarachchi, "Evolutionary algorithm for sinhala to english translation.," in *2019 National Information Technology Conference (NITC)*, Colombo, 2019.

[11] "Type in Sinhala," Gishan Networks, 2023. [Online]. Available: https://sinhalaunicode.gishan.net/write.

[12] "Google Input Tools," [Online]. Available: https://www.google.com/inputtools/try/.

[13] M. Kulatunga, "Madura English-Sinhala Dictionary - Online Language Translator," 2008. [Online]. Available: https://www.maduraonline.com/.

[14] "ChatGPT," [Online]. Available: https://chat.openai.com/.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, K. Łukasz and I. Polosukhin, "Attention is all you need.," in *Advances in neural information processing systems*, Long Beach, 2017.

[16] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition.," in *Proceedings of the IEEE conference on computer vision and pattern recognition.*, 2016.

[17] "Built-in Types," Python Software Foundation, [Online]. Available:

https://docs.python.org/3/library/stdtypes.html.

[18] X. Xu, L. Wang, R. Liu and T. Xu, "Deep learning based news text classification software design," *Journal of Physics: Conference Series*, vol. 2031, no. 1, pp. 12 - 67, Aug. 2021.

[19] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. A. Patwary, Y. Yang and Y. Zhou, "Deep learning scaling is predictable, empirically.," *arXiv preprint arXiv:1712.00409*, 2017.

[20] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning.," 2017.

[21] G. Klambauer, T. Unterthiner, A. Mayr and S. Hochreiter, "Self-normalizing neural networks," *Advances in neural information processing systems*, 2017.

[22] "Welcome to Flask," 2010. [Online]. Available: https://flask.palletsprojects.com/en/2.1.x/.

[23] "ngrok documentation," [Online]. Available: https://ngrok.com/docs.