

Translation of Hindi in Roman Script into English: Use of Transformer

by

Parth Modi
202111030

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY

in

INFORMATION AND COMMUNICATION TECHNOLOGY

to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

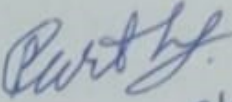


May, 2023

Declaration

I hereby declare that

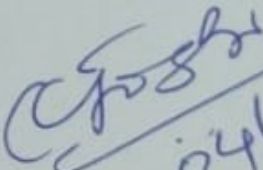
- i) The thesis comprises my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.


04/09/23

Parth Modi

Certificate

This is to certify that the thesis work entitled Translation of Hindi in Roman Script into English: Using Transformer has been carried out by Parth Modi for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.


04/09/23

Prof. Manjunath V. Joshi
Thesis Supervisor

Acknowledgments

I want to express my sincere gratitude to my research Guide, Professor Manjunath V. Joshi, Dhirubhai Ambani Institute of Information and Communication Technology, for allowing me to work in Natural language processing and providing his invaluable guidance throughout this work. He helped and motivated me to explore new ideas throughout the research. It was a privilege and honor to work and study under his guidance.

Also, thanks to all my peers, who helped me in one way or another throughout my research. I also want to thank my institute for providing me with the necessary resources for the research.

Lastly, I thank my family for their help and support in this unprecedented quest.

Contents

Abstract	v
List of Principal Symbols and Acronyms	vi
List of Tables	vii
List of Figures	viii
1 Introduction	1
2 Literature Survey	5
3 Transformer	8
3.1 Encoder stack and Decoder Stack	9
3.1.1 Encoder	9
3.1.2 Decoder	10
3.2 Attention	10
3.2.1 Attention using Dot Product	11
3.2.2 Multi-Headed Attention	11
3.3 Usage of attention in Machine Translation	13
3.4 Position-wise Feed Forward Networks	13
3.5 Embeddings and softmax operation	14
3.6 Position wise Encoding	14
3.7 Working of attention	14
3.7.1 Computation of score for Encoder State	15
3.7.2 Computation of the Attention Weights	15
3.7.3 Computation of the Attention Vector	15
3.7.4 Decoder Output	16
4 Proposed Approach	17
4.1 Pre-processing	17

4.2	Tokenizer	17
4.3	Transformer Model Training	19
4.4	Hyper-parameters	19
4.5	Hardware Specifications	20
5	Experimental Results	21
5.1	Dataset	21
5.2	BLEU score Evaluation	21
6	Conclusion and Future Work	25
	References	26

Abstract

Translation from one language to another is a complex problem in machine learning and one in which the machine still cannot achieve satisfactory results. The recent focus for solving this challenge has been on neural machine translation (NMT) techniques using architectures such as recurrent neural network (RNN) and long short-term memory (LSTM). Even though they give slightly better results than the previously available conventional techniques, the transformer can outperform these NMT techniques. To the best of our knowledge work is yet to be carried out in translating Hindi language sentences written in Roman (English) letters into English. In this report, we discuss how the architecture of transformer that uses attention mechanism is used to translate Hindi language sentences written in Roman letters into English sentences. Since there was no dataset available till now, our work also involves creating a dataset for training and testing. Our results are compared with other approaches using BLEU score as a measure.

List of Principal Symbols and Acronyms

BLEU Bilingual Evaluation Understudy

BP Brevity Penalty

GRU Gated Recurrent Units

HMT Hybrid Machine Translation

k Key

LSTM Long Short-Term Memory

MT Machine Translation

NLLB No Language Left Behind

NMT Neural Machine Translation

q Query

RBMT Rule-Based Machine Translation

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

Seq2Seq Sequence to Sequence

SMT Statistical Machine Translation

v Value

List of Tables

5.1 BLEU scores comparison 23

List of Figures

1.1	Illustrations of translations from Hindi written in Roman (English) script to English	4
3.1	Model Architecture of the Machine translation system using transformer [21]	8
3.2	Transformer sub-layer architecture [21]	9
3.3	Look ahead mask	10
3.4	Process for forming self-attention vector	12
4.1	Block Diagram	17
4.2	Tokenized sentences	18
4.3	Sub-worded sentences	18
5.1	Input sentence and its translated sentence using proposed method	22
5.2	Snapshot of translated sentences using NLLB [20]	23
5.3	Snapshot of one of the above translated sentences using NLLB [20]	24

CHAPTER 1

Introduction

Machine translation (MT) is a sub-field of artificial intelligence and computational linguistics that aims to translate text or speech from one language into another automatically. In order to process and comprehend the meaning of the source text and producing an equivalent translation in the target language involves using computer algorithms and models.

Statistical machine translation (SMT), rule-based machine translation (RBMT), and neural machine translation (NMT) are some of the different methodologies that can be used for machine translation. Statistical machine translation uses statistical models like dictionary matching trained on substantial bilingual corpora. In contrast, Rule-based systems use linguistic rules and dictionaries created manually. Neural machine translation (NMT) has grown in popularity recently and uses deep learning methods and neural networks to enhance translation quality. NMT models require only a fraction of the memory of the more conventional SMT models.

Text is a time series data that is sequential. Recurrent neural network (RNN) is used to handle and process time series data. Recurrent neural networks (RNNs) are made for handling such sequential data. RNNs contain an internal memory that enables them to keep track of prior inputs or timesteps, unlike standard feed-forward neural networks that process inputs individually. Because of this memory, RNNs can recognize temporal connections and the context in sequential data. The concept of machine translation dates back to the early days of computing when scientists and researchers began exploring ways to automate the translation process.

There are many benefits of machine translation. It helps in communication and knowledge exchange across languages, which benefits businesses, organizations, and individuals to access information and connect with people from different backgrounds.

In addition, domain adaptation has received attention recently, which involves

training machine translation models on certain domains or topic areas to increase translation accuracy in specialized contexts. This has shown to be helpful in industries like legal, medical, and technical translation, where specialized terminology and nuanced expressions are essential.

However, there are still many difficulties with machine translation. Accurate translations can be challenging due to cultural context, informal idioms, and language complexities. Low-resource languages with limited training data also create unique challenges. Nonetheless, ongoing research and advancements push the boundaries of machine translation, aiming for more accurate and natural-sounding translations.

Different metrics and evaluation techniques have been developed to assess and compare the quality of machine translation systems. Manual measurements like Bilingual Evaluation Understudy (BLEU) , which evaluates the overlap between machine-generated translations and reference translations, and human evaluation, where human judges evaluate the quality of translations, are common approaches.

Many NMT models generally target a single language pair translation, where the input is in the form of one language, and the generated translated output is in the desired language. Whereas models like multilingual neural machine translation MNMT utilize a single NMT model to enable translation among multiple languages instead of training a single model for every language pair. However, No research has been done on the ability of NMT models to translate sentences of a language written in letters of another language into the desired target language, which needs to be done. This kind of translation can be helpful for the translation of chats in communication applications. This thesis tries to establish an NMT model's ability based on a transformer's architecture to translate sentences in the Hindi language written in Roman letters into English, which is a more challenging task than translating Hindi sentences written in Hindi into English.

Problem statement - Given a dataset with sentences in the Hindi language written in Roman letters, our aim is to translate these sentences into English using the transformer as illustrated in figure 1.1.

Some applications output words in Hindi letters when given input of Hindi words written in Roman letters, and some applications translate those Hindi words into English. But none of them perform the whole work of translating directly to English. Moreover, it faces difficulty when an entire sentence is given as the input, as it only works on words and not the entire sentence.

The motivation behind choosing this topic is that in daily conversation, everyone uses their mother tongue, or what they are fluently comfortable in, but to do that, one needs to download/install another linguistic software on their device. Moreover, the person on the other end also needs to download/ install that software, and if the receiver is not comfortable in that language, it creates another difficulty. To tackle this problem, we are working on this thesis topic.

Our work has not been attempted by anyone to the best of our knowledge. Hence, no dataset was readily available for us to work upon. So, we have created our dataset consisting of sentence pairs of Hindi sentences written in Roman letters and their respective English translation, as shown in figure 1.1, which we will look into in detail in Chapter 5.

Source Sentence	Translated Sentence
in vedon ke antim bhaag ko upanishad kaha jaata hai, hamaara vaastav mein matalab yah hai ki ve dhyaan na dene mein bure hain	The ending portion of these vedas is called upanishad what we really mean is that they are bad at not paying attention
isamen aapake saamane logon kee paristhitiyaan nihit hain aur ham kaun hote hain yah kahane vaale bhee ki ve galat hain	In this lies the circumstances of people before you and who are we to say even that they are wrong
parivaar niyojan viyatanaam mein shuroo hua aur ve chhote parivaaron ke lie gae	The family planning started in vietnam and they went for smaller families
klinikal pareekshan abhee bhee kie jaane kee aavashyakata hai	Clinical trials still need to be carried out

Figure 1.1: Illustrations of translations from Hindi written in Roman (English) script to English

CHAPTER 2

Literature Survey

let us look at some machine translation models. Statistical machine translation (SMT), developed during the 1980s, was designed based on a statistical model. Given a large parallel corpus of translated text data, SMT takes a series of symbols provided in the source language with its related vocabulary and converts them into a series of symbols in the target language formed with its corresponding vocabulary [13]. It faces challenges when the vocabulary of both languages is of significantly different strength. [12] explains the basic idea behind SMT and sheds light on the challenges faced. It shows the classification of various methods in this area as well.

Next, we look into rule-based machine translation (RBMT) , introduced in 1985. RBMT is designed on the definition of rules for syntax and morphology of a language. In RBMT, the collection of rules and vocabulary of both the source and target languages are required as the resources. It faces challenges when the grammatical rules of both languages are too different. [19] explains the basic idea behind RBMT and compares the statistical and rule-based approaches to machine translation with the help of a case study from the perspective of Indian languages.

Methods discussed so far are inadequate for machine translation tasks, leading to the invention of hybrid machine translation (HMT) . It combines and uses various machine translation methods in a single system. The most popular and frequently used combination is SMT and RBMT. Work in [2] explains some of the popular hybridization methods and how they try and integrate the principal attributes of the various individual techniques. It also discusses the application of these HMT methods.

We now look at the types of machine translation techniques explored till now using neural networks. Earlier recurrent neural networks were used to handle sequential data. The key feature of an RNN is its recurrent connection, which forms a loop, allowing information to persist and be passed from one time-step to the next. At each time step, an RNN receives an input and produces an output

and a hidden state. The hidden state serves as the memory of the network, storing information from previous time-steps and influencing the processing of future inputs [18].

RNNs can be seen expanding through time, with each time-step denoting a distinct network instance connected to its preceding and subsequent time-steps. The sequential aspect of the data processing and the information flow through the recurrent connections are both highlighted by this unfolding picture. RNNs are Sequence to Sequence (Seq2Seq) type of models, which takes second word as the input only after the first word has been processed. Which is one of the drawbacks of the RNN.

There are four types of RNNs based on the number of inputs and outputs in the network.

- one to one : behaves as normal neural network, takes one input, gives one output.
- many to one: many inputs, single output, widely used in sentiment analysis.
- one to many: one input, multiple outputs, most widely used for image captioning.
- many to many: multiple input, multiple output, this is used in the machine translation task.

RNNs have disadvantages like vanishing or exploding gradients, so it faces difficulty processing longer sequences. It takes the output of the previous step as the input at the next step, so it can't be parallelised.

Long Short-Term Memory (LSTM), the most popular type of RNN, addresses some of the drawbacks of conventional RNNs by introducing gating mechanisms that regulate the flow of data into and out of the hidden state. When handling sequences with pauses or delays between important events, LSTMs excel at learning long-range dependencies [5]. However, RNNs must overcome difficulties such as vanishing or exploding gradients, where it becomes challenging to learn how distant time-steps affect the current time-step. Variants like Gated Recurrent Units (GRUs) and other cutting-edge architectures have been created to overcome these problems.

A machine translation system that uses an artificial neural network to increase the fluency and accuracy of the machine translation model is classified as a neural machine translation (NMT) model. NMT models comprise encoder and decoder structures and use recurrent neural networks (RNNs). RNN is cyclic, enabling it

to learn the repeated sequences much more efficiently than other networks. In models such as ConvS2S [3], Extended Neural GPU [8], and Byte-Net [9], conventional neural networks are used as the building blocks. All these models aim to compute the hidden representation simultaneously for all input and output positions.

The issue with such models mentioned above is that when the distance between two arbitrary words/tokens in the input or output positions increases, so do the computational resources required to build a connection between them. It rises linearly in the case of ConvS2S and logarithmically in the case of ByteNet. This presents a problem since it becomes challenging to learn the dependencies between positions [6] that are not close together. Any machine translation model must learn about these dependencies.

Utilizing the transformer reduces the above-mentioned problems, and the amount of operations needed to discover dependencies between distant locations becomes constant and independent of distance. Here the idea of self-attention can be used to calculate the representation of a sequence. It can also be called intra-attention. It is used to relate the diverse positions of a single sequence. The concept of self-attention has been successfully utilized in performing various tasks, which include abstractive summarization [16], reading comprehension [1], learning task-independent sentence representations [11] and textual entailment [15]. We will look at the transformer and the attention in the next chapter. As there was no dataset available prior to our work, there was no tokenizer available for our dataset also. Here we also generated a tokenizer for our dataset, which we will look at in Chapter 4.

In this chapter, we looked at some of the machine translation methods and models and their shortcomings. Now, let us look at the architecture of the transformer in more detail in the next chapter.

CHAPTER 3

Transformer

This machine translation model using transformer follows an encoder-decoder structure as shown in Fig 3.1. The job of the encoder is to map an input sequence (a_1, \dots, a_n) where a_i signifies a word embedding onto $c = (c_1, \dots, c_n)$, where c_i is a representation of a word assigned to it by the encoder and c (Context Vector) is a sequence of continuous representations. The decoder uses it to generate a sequence (b_1, \dots, b_n) as the output of words one word at a time. The nature of the model is auto-regressive. The previously generated symbols are taken as input for generating the next symbol.

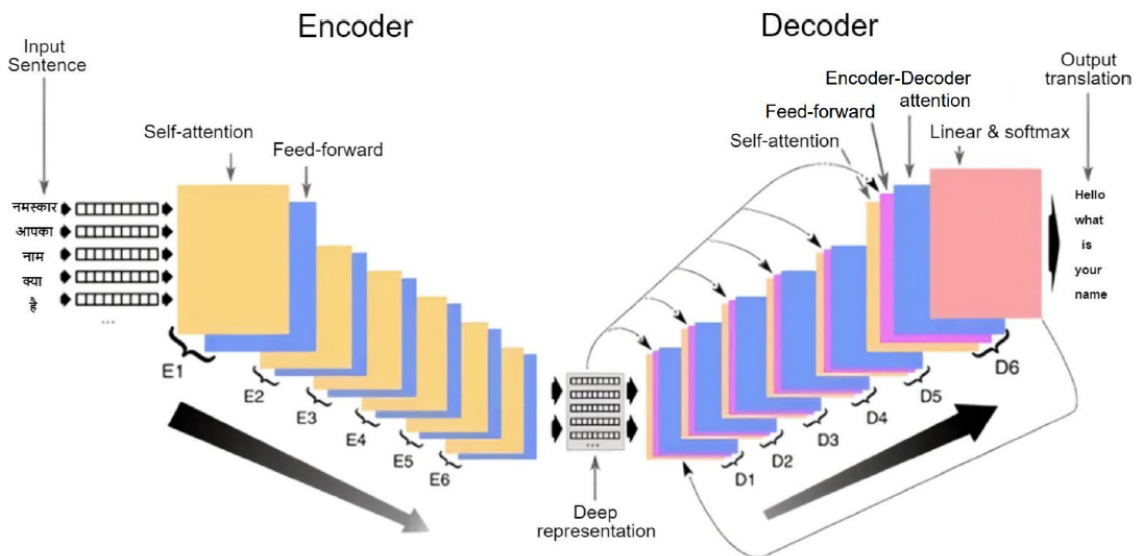


Figure 3.1: Model Architecture of the Machine translation system using transformer [21]

Now let us look at the pictorial representation of the architecture of a transformer sub-layers as shown in Fig 3.2. It consists of self-attention layers shown in the left part of Fig 3.2 and point-wise fully connected layers shown in the right part of Fig 3.2. The left half of the transformer is used in each encoder layer as

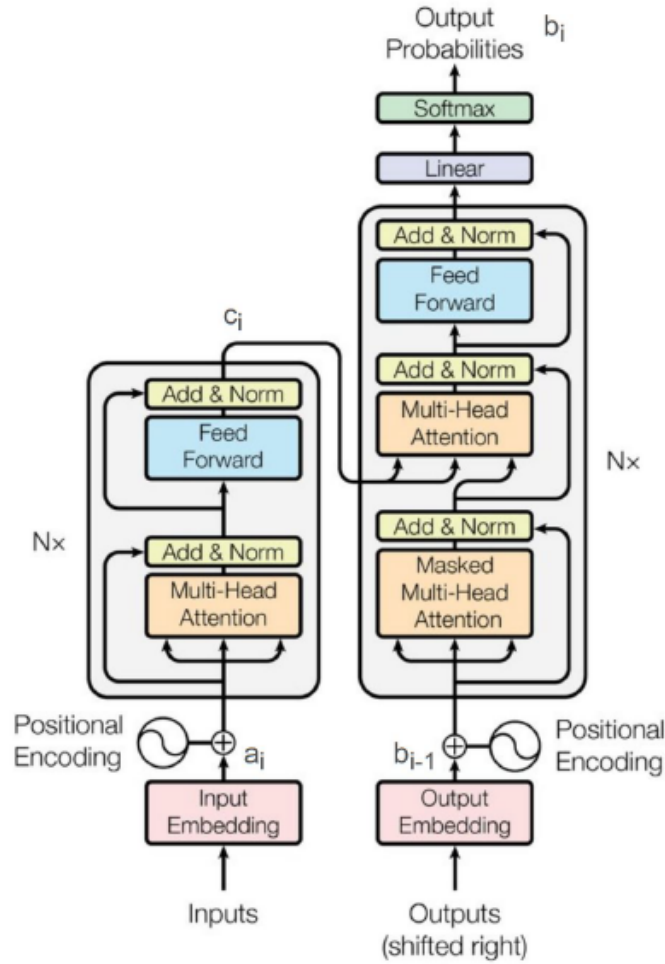


Figure 3.2: Transformer sub-layer architecture [21]

shown in Fig 3.1 using two sub-layers in each layer, and the right half of the transformer is used in each decoder layer as shown in Fig 3.1 using three sub-layers in each layer.

3.1 Encoder stack and Decoder Stack

3.1.1 Encoder

The job of the encoder is to convert the input word tokens into an embedding format that can be further used by the decoder while providing the translation. As seen in Fig 3.1, six layers comprise the encoder, where every layer is divided into two sub-layers. The first of these sub-layers is a mechanism that uses multi-headed self-attention; which will be covered in section 3.2.2. The second layer

is a feed-forward network, which is fully connected position-wise. A residual connection [4] is used in both sublayers, subjected to layer normalization [7]. All sub-layers in the model as well as the embedding layers, produce outputs of $d_{model} = 512$.

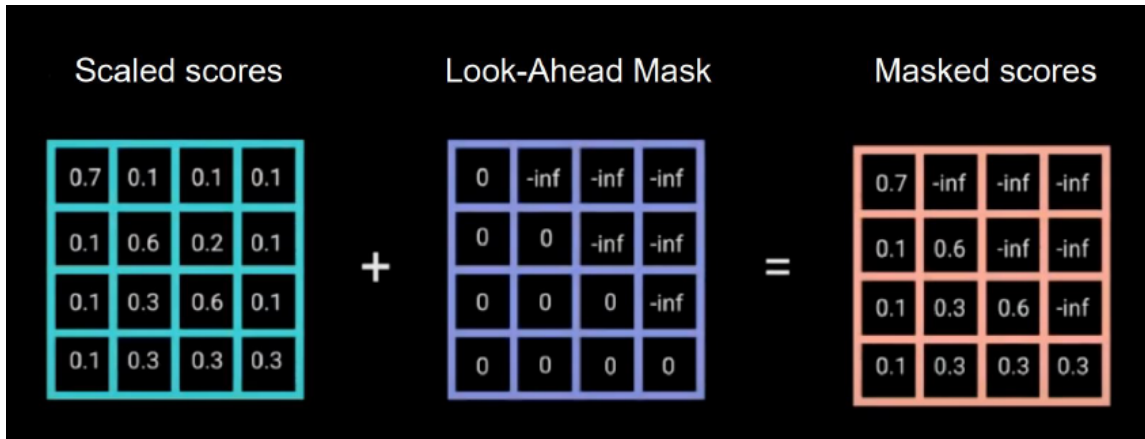


Figure 3.3: Look ahead mask

3.1.2 Decoder

The decoder's job is to take the embedded input from the encoder and convert them into their corresponding translations, and provide the word in the target language as the final output. As seen in Fig 3.1, the decoder also consists of 6 layers, each further divided into three sub-layers. The initial two sub-layers are similar to those used in the encoder; the third sub-layer performs encoder-decoder attention on the output generated by the encoder. As in the case of the encoder, a residual connection is used here, and this output undergoes layer normalization. However, there is a subtle modification to the self-attending sub-layer in the decoder; the combination of the look-ahead mask, as shown in Fig. 3.3, and the offset of output embeddings by one position makes sure that the prediction for the symbol in the position I depend only on the previously generated outputs.

3.2 Attention

When a sentence is composed in any language, words within it have some inter-relationship. To capture this relationship between words within a sentence, the concept known as "attention" was formed. It decides which word in the sentence pays how much attention to another word in that sentence. Attention [21] may be understood as a relationship between queries and a set of pairs, whose nature

is that of a key and value to the output. Here, $\text{query}(q)$, $\text{key}(k)$ and $\text{value}(v)$ are all vectors. For each value, a weight will be assigned. This weight is computed using a query's similarity function with its respective key. Now the output can be calculated as a weighted sum of these values. Two kinds of attention are used in the machine translation system, which is discussed below.

3.2.1 Attention using Dot Product

For calculating attention, we need an input that is made up of queries q and keys k , which have dimensions of d_k . We also need values v , which has a dimension of d_v . Queries in the machine translation model can be understood as the input word vector; keys are the input word vectors for all the other words, and values are the collection of positional input embedded vectors shown in the flowchart in Fig 3.4. The mathematical representation of attention is given as follows :

$$AT(q, k, v) = S\left(\frac{qk^T}{\sqrt{d_k}}\right)v \quad (3.1)$$

The above equation generates an attention matrix AT , S is the soft-max activation function, and q , k , and v are queries, keys, and values, respectively. These attention vectors are computed for each word during training. Upon computation, they will contain the information regarding which word is being paid the most attention by the encoded-word.

3.2.2 Multi-Headed Attention

Our method uses multi-headed attention. Using multi-headed attention [21] allows the model to simultaneously attend to various representation sub-spaces at various locations instead of using a single attention function. The mathematical representation of multi-headed attention is given as:

$$M(q, k, v) = [head_1, \dots, head_h]A^o \quad (3.2)$$

M is multi-headed attention computed for a specific query, key, and value vector set. Here $[\]$ represents the concatenation operation of the multiple heads taken from 1 to h , where the value of h is taken as eight as per [21]. It is necessary to take multiple heads as it increases the model's ability to focus on various positions. It also prevents the word from dominating the encoding when an attention vector is calculated.

Each $head_i$ is computed as:

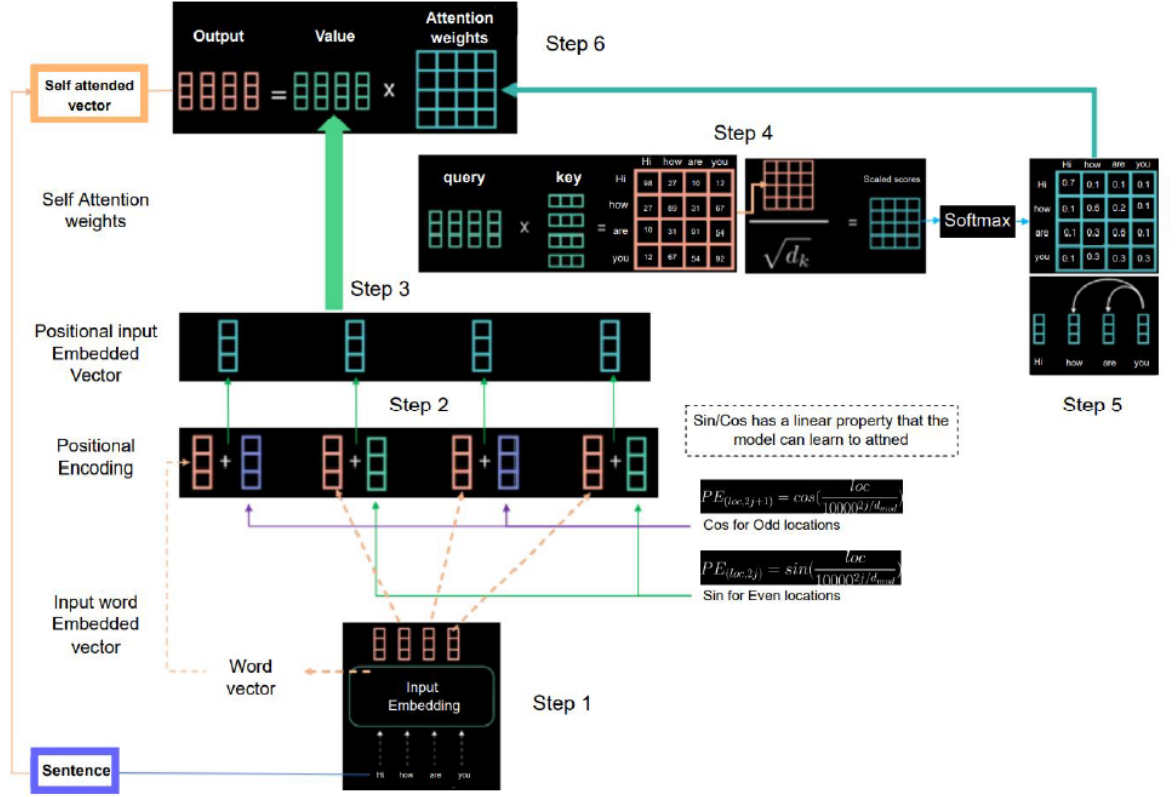


Figure 3.4: Process for forming self-attention vector

$$head_i = AT(qA_i^q, kA_i^k, vA_i^v) \quad (3.3)$$

The projections A_i^q , A_i^k , A_i^v , and A^o used above are parameter matrices, and these parameters will be learned as the model is trained on the desired data and will be adjusted accordingly using weights. Their dimensions are given as follows:

$$A_i^q \in \mathbb{R}^{d_{mod} * d_q}$$

$$A_i^k \in \mathbb{R}^{d_{mod} * d_k}$$

$$A_i^v \in \mathbb{R}^{d_{mod} * d_v}$$

$$A^o \in \mathbb{R}^{hd_v * d_{mod}}$$

Here query (q), key (k), and value (v) are projected h times linearly while using various linear projections that can be learned. The dimensions d_k and d_v are set to 64, and d_{mod} is set to 512 according to [21]. When the attention function is utilized

parallelly on each of the projected versions of q , k , and v , they will generate output values of dimension d_v . These outputs are concatenated and then projected once more, giving us the final attention vector of each word, as shown in figure 3.4.

3.3 Usage of attention in Machine Translation

In machine translation using the transformer, attention is used in different ways:

- To enable each position within the encoder to pay attention to all the positions within the previous layer, self-attention is used within the encoder layers; in these layers, q , k , and v are all taken from the output generated by the encoder in the previous layer.
- The decoder also contains self-attention layers, which enable each position of the decoder to pay attention to the positions in the decoder. If the flow of information is leftward, it will interfere with the auto-regressive property. As shown in Fig. 3.3, a look-ahead mask is utilized to prevent this. It is used to pay attention till the current position, including the current position.
- In the encoder-decoder attention layer, each position in the decoder is enabled to pay attention to each position in the input sequence. The q is taken from the output of the previous decoder layer, and the k and v are taken from the output generated by the encoder.

3.4 Position-wise Feed Forward Networks

A fully connected layer accompanies each layer in the encoder and the decoder. It is applied identically and separately for each position. It is made up of Rectified Linear Unit (ReLU) activation between two linear transformations [21]. For position x it is given by:

$$F(x) = \max(0, xA_1 + c_1)A_2 + c_2 \quad (3.4)$$

The output $F(x)$ is obtained by applying the ReLU activation to the result of the first linear transformation, followed by another linear transformation with bias terms.

Where A_i is a projection, and c_i is a bias.

3.5 Embeddings and softmax operation

For converting the input and output tokens into vectors of dimension, d_{mod} embeddings are used. A softmax function is combined with the learned linear transformation to help predict the probabilities of the next word [21].

3.6 Position wise Encoding

This translation model does not contain any recurrence or convolution operations. This creates a need to include information regarding the relative or absolute positions of the words in the sequence, which will help the model use the order of the sequence. This is where positional encodings are used; they are added to the input embeddings in the encoder and decoder stacks. The dimensions of both these encodings are the same d_{mod} . For this purpose, sine and cosine functions with different frequencies are used; they are given below:

$$PE_{(loc, 2j)} = \sin\left(\frac{loc}{10000^{2j/d_{mod}}}\right) \quad (3.5)$$

$$PE_{(loc, 2j+1)} = \cos\left(\frac{loc}{10000^{2j/d_{mod}}}\right) \quad (3.6)$$

Here, loc is the position of the vector, and j is the vector's dimension, as seen in Fig 3.4. The positional encoding vector will follow a particular pattern. The model learns this pattern through which it will help establish the location of each word or the separation between different words in the sentence. These positional encoding vectors are added to every input encoding. By doing this, the encodings are provided with relevant separation of the encoding vectors after they are projected into the query, key, and value vectors to compute the attention vector.

3.7 Working of attention

In traditional Seq2Seq models comprising an encoder-decoder structure using RCNN or LSTM, the encoder will process and encode the input sequence into a context vector of fixed length. This context vector is fed to the decoder as an input; using this, the decoder starts predicting the output. However, the problem associated with a fixed-length context vector is that it cannot remember longer sequences. It tends to forget the beginning part of the sequence once the entire sequence is processed. This is the motivation for the development of the atten-

tion mechanism. To understand how the attention mechanism works, let us walk through an example where a Hindi sentence is converted into English, as shown below.

3.7.1 Computation of score for Encoder State

Each encoder state $E_1, E_2, E_3, E_4,$ and E_5 stores the local information of the input sequence. The objective is to predict the first word, but the decoder has no initial state, so we consider the last encoder state, E_5 , as the previous decoder state. We now train a feed-forward network using all the encoder states and the current decoder state. The information to predict the first word in Hindi is stored in the encoder states E_1 and E_2 . Therefore we need the decoder to pay more attention to these states than the others. This is the reason for training a feed-forward network to learn to assign a higher score to the states that require more attention and assign a lower score to the states that are meant to be ignored. Let $S_1, S_2, S_3, S_4,$ and S_5 be the scores generated for each encoder state. Since the information is in E_1 and E_2 , their scores in S_1 and S_2 will be higher compared to the others.

3.7.2 Computation of the Attention Weights

A softmax is applied to the scores generated to produce attention weights $w_1, w_2, w_3, w_4,$ and w_5 . In this example, the values of w_1 and w_2 will be higher than the others to help predict the first word.

3.7.3 Computation of the Attention Vector

After the generation of the attention weights, a context vector will be generated that will be used by the decoder to predict the next word in the sequence. The context vector is calculated as follows:

$$\text{contextvector} = w_1 * E_1 + w_2 * E_2 + w_3 * E_3 + w_4 * E_4 + w_5 * E_5 \quad (3.7)$$

In this case, the values of w_1 and w_2 are higher than the others. Therefore, it contains more information from the states E_1 and E_2 . Concatenate context vector with an output of the previous time step: The decoder uses this context vector and the output word generated from the previous time step to predict the next word in the sequence. Since the first time step, there is no output from the previous time step, a unique token $\langle \text{START} \rangle$ is given to the decoder.

3.7.4 Decoder Output

Concatenate context vector with an output of the previous time step: The decoder uses this context vector and the output word generated from the previous time step to predict the next word in the sequence. Since the first time step, there is no output from the previous time step, a unique token $\langle \text{START} \rangle$ is given to the decoder. The decoder predicts the first word of the sequence; a hidden state, d_1 , is also generated along with this output.

Decoding at Time step 2: To predict the next word in the sequence, the internal state d_1 , along with all the encoder states E_1, E_2, E_3, E_4 , and E_5 , are given to the feed-forward network, which then generates new S_1, S_2, S_3, S_4 , and S_5 scores which are used to compute new attention weights, and a new context vector is generated. This context vector is concatenated with the previous time step output and given to the decoder to predict the next word, producing a new internal state d_2 .

This process is repeated till the decoder generates the $\langle \text{END} \rangle$ token. After the generation of this token, the process is terminated.

The main thing to be noted is that in the case of traditional Seq2Seq models, a fixed context vector was used for every decoder time step. In contrast, in the case of the attention mechanism, a new context vector is computed every time by using newly generated attention weights.

CHAPTER 4

Proposed Approach

As shown in the figure 4.1 this chapter describes our proposed approach:



Figure 4.1: Block Diagram

4.1 Pre-processing

First, we need to check if the data is in the form of a parallel corpus. If the data is misaligned, we need to align it accordingly. Then we remove empty lines, if any. Then we divide the dataset into the train, validation, and test dataset.

This pre-processed data will be fed to the tokenizer.

4.2 Tokenizer

Although the dataset contains Hindi words, they are written in Roman letters, so neither a tokenizer for Hindi nor English can be used. Therefore we had to create our custom tokenizer for our dataset using sentence-piece tokenizer [10], which can be seen in figure 4.2.

```
src-test-tokenized.txt
File Edit View
_aur _phir _main ne _har _jagah _bade - bad e _post ar _chipaka _die
_ise _ek _majaboot _aur _su gat hit _tred _yooneyan _aandolan _ka _shakti shaalee _samarthan _praapt _tha _aur _phir
_jagah _bade - bad e _post ar _chipaka _die
_ise _ek _majaboot _aur _su gat hit _tred _yooneyan _aandolan _ka _shakti shaalee _samarthan _praapt _tha _nae _saal _k
_dauraan _bhaarat _ke _lie _chaaval a _kee _ant im _yaatra _jab _vah _aur _unake _pati _parivaar _ke _saath _samay _bit
_bhaarat _ke _lie _chaaval a _kee _ant im _yaatra _nae _saal _kee _chhutt ee _ke _dauraan _jab _vah _aur _unake _pati
_saath _samay _bita ate _hain , _praty ek _pariyojana _ek _philm _hotee _hai
_praty ek _pariyojana _ek _philm _hans ee _hai
_hans ee _shuroo _se _hee _uchchaar an _kee _saamaany _shailee _mein _antar _tha a sh ab don anu shthaan _karane _ke n
_uchchaar an _kee _saamaany _shailee _mein _shuroo _se _hee _antar _tha _saty avat ee _vyaas ajee _kee _aag ya anusaar
_karane _ke _niyam on _ne _a mb ika _ko _dr ut a raasht r _aur _a mb aalika _ne _paandu _ko _janm _diya
_kaaph ee _badee _sankhya _mein _hinduon _ne _islaam _ko _gale laga liya _aur _yah aan _tak _ki _jo _log _nahin _the
_saty avat ee _vyaas ajee _ke _aadesh _ke _anusaar _musalamaan on _ke _prati _behatar _vyavahaar _karate _the , _dr ut
_janm _dene _ke _lie _a mb ika _ko _banaaya _aur _a mb aalika _ne _paandu _ko _janm _diya
_kaaph ee _badee _sankhya _mein _hinduon _ne _islaam _ko _apanaaya _aur _jin h on ne _nahin _apanaaya _ve _bhee _ab _mus
_prati _behatar _vyavahaar _kar _rahe _the
_abhya gat _chale _gae
_aur _unh on ne _aadesh _ke _puja ar iyon _ko _abhee _bhee _ise _dhaarmik _samarp an _ke _saath _chalaaya
_aadesh _ke _puja ar ee _abhee _bhee _ise _dhaarmik _samarp an _ke _saath _chalaate _hain
_jama sa ad apur _2 _jo on _pha ij aabaad _kort _ne _yah _aadesh _diya _hai
_yah _gupha _mandir _shrnkhala _mein _ek _amaatr _udaaharan _hai _jis amen _mandap _ke _agra bhaag _ke _don on _chhor
_on _kee _ek _baahar ee _jod ee _hai _jama sa ad apur
_mera _maan a _hai _ki _aphreeka _ek _vi bh ak ti _bindu _par _pahunch _gaya _hai
_vah _jo on _aur _sit ambar _ke _beech _maanason _kee _baar ish _ke _ka ee _kshetr on _mein _baadh _ke _lakshy _bhee
_karegee , _jin amen _se _mumbee _mein _sabas e _adhik _varsha _hotee _hai
_aur _bhaarat _mein _par idhaan _kaarakhaane _aur _phir _ve _un _vichaaron _ko _kriya an v it _kar _rahe _the _jo _log
```

Figure 4.2: Tokenized sentences

We have also used a tokenizer to make sub-worded sentences. Sub-words are generated by dividing one word into two or more than two words or mixing two words to create another word, which can be seen in figure 4.3. This can be useful when a word is encountered while translating not in the training vocabulary to predict the nearest meaningful word.

```
whatsappengt1.subword
whatsappsrc1.subword
> Users > Parth > whatsappsrc1.subword
This document contains many non-basic ASCII unicode characters. Disable Non ASCII Highlight
1 | _main _aapako _aise _hee _ek _baal _raajaneetigy _ke _baare _mein _bataa n _chaahata _hoon , _jis e _karane _kee _anuma ti
2 | _yah _pratishat _bhaarat _mein _pratishat _se _bhee _adhik _hai _aaedee _aapako _aise _hee _ek _bachche _ke _baare _mein
3 | _in _ved on _ke _a ntim _bhaag _ko _upanishad _kaha _jaata _hai , _hamaar a _vaastav _mein _matal ab _yah _hai _ki _ve _dhy
4 | _in _ved on _ke _a ntim _bhaag _ko _upanishad _kaha _jaata _hai
5 | _is amen _aapake _saaman e _log on _kee _paristhitiy aan _nihil _hain _aur _ham _yah _kahane _vaale _kaun _hote _hain _ki _v
6 | _aap _chaahate _hain _ki _aapak a _bachcha _kise e _aise _skool _mein _jæ _jo _kise e _gair - rakharakhaav _vaale _vishesh
7 | _krpaya _su _nishchit _karen _ki _aap _upa yukt _pra patr _shrenee _dhaarmik _paath _ka _upayog _karate _hain
8 | _shrenee _dhaarmik _paath
9 | _is alie _kise e _prakaar _ka _nyaay _hai
10 | _pahale _do _ko _a vishvasaneey _paaya _gaya _aur _abhiyoja n _ka _maamal a _mukhy _roop _se _shesh _paanch _anumodak on _ke
11 | _aur _ab _varta maan _mein _nepaal _mein _sarakaar _ke _maadhyam _se _praakrtik _upachaar _aayurved ik _aur _aadhunik _upach
12 | _sansad _kee _samay _seema _varsh on _hai _aur _yah _us ase _pahale _bhag _ho _jaega _aur _ab _varta maan _mein _nepaal _m
13 | _ii _nyaayaadheesh on _dvaara _adhik rt _kie _jaane _par _rs _se _adhik _nahin _hone _vaale e _raash iyon _ke _kaaran on _ke
14 | _inamen _se _la hadee _ek _lokapriy _hai _yah aan _tak _ki _paanee _mein _haidrojan _salph aid _kee _peepee em _kee _sa and
15 | _yah _dheere - dheere _badala _islaam _eesaee _dharm _ke _baad _duniya _ka _doosara _sabas e _bada _dharm _hai
16 | _yah _dheere - dheere _badal _gaya
17 | _vitt eey _sansthaon _se _su juk ee _elaarabee _ke _paksh _mein _hissedaar ee _ko _utaarane _kee _ummeed _hai , _varta maan
18 | _vitt eey _sansthaon _dvaara _su juk ee _elaarabee _ke _paksh _mein _varta maan _mein _pratishat _hissedaar _aaraarabee _aur
19 | _naren _ne _vaanijy _dootaavaas _ke _saath _teen _ya _chaar _baithak en _ke en _lek in _paaya _ki _vah _ko ee _pragati _nahi
20 | _k uchh _maamalon _mein _uchh _nyaayaalay on _ke _paas _sarvochh _nyaayaalay _kee _tulana _mein _pratha m _drshta ya _adaa
21 | _k uchh _maamalon _mein _uchh _nyaayaalay on _ke _paas _sarvochh _nyaayaalay _kee _tulana _mein _pratha m _drshta ya _ke
22 | _adhninyam _ke _tatah _ek _se _adhik _kle m _tribyoon al _sthaap it _kie _ja _sakate _hain _aur _tribyoon al _ka _gathan _k
23 | _roshan ee _chaal oo _karane _ke _lie _ya _use _ek _gil aas _paanee _men _laane _ke _lie
24 | _f aarasee _darshan _mein _phir ada us _udyaan _ko _mugal _saahity _mein _aadarsh _ke _roop _mein _prastut _kiya _gaya _hai
25 | _f aarasee _darshan _mein _phir ada us _udyaan _ko _mugal _saahity _mein _aadarsh _ke _roop _mein _prastut _kiya _gaya _hai
```

Figure 4.3: Sub-worded sentences

These tokenized sentences are used to train the transformer.

4.3 Transformer Model Training

As described in Chapter 3 on transformer, a transformer model with six encoder and decoder layers comprising multi-headed attention is created to train on the dataset. All the hyper-parameters that can be tuned are mentioned in the config file for fine-tuning.

4.4 Hyper-parameters

We trained the transformer model on our system with the below tuned hyperparameters:

- Train Steps: 60,000
- Valid Steps: 500
- Warmup Steps: 8000
- Seed: 3435
- Word vector size: 512
- layers: 6
- Hidden state size: 512
- Self-attention head: 8
- Optimizer: Adam
- adam beta1: 0.9
- adam beta2: 0.998
- Decay method: noam
- Learning rate: 2.0
- max gradient normalization: 0.0
- Batch size: 4096
- Dropout:0.1
- Label smoothing: 0.1

- max generator batches: 2
- initial parameter: 0.0
- param init glorot: True

4.5 Hardware Specifications

We trained our transformer model on a machine with eight Intel (R) Xeon (R) E5-2609 v4 CPUs with 1.7 GHz frequency with 64 GB RAM. which took two days to complete training.

CHAPTER 5

Experimental Results

As no datasets containing Hindi sentences written in Roman letters existed before our work, we have created the parallel dataset.

5.1 Dataset

The dataset required for a translation model should be a parallel corpus in text format. In one file, the sentences are given in the source language; in the other, the translated equivalents of the first file are shown in the target language.

The dataset used for training the model is an original dataset created by us inspired by "Samanantar" [17]. The Samanantar dataset is the most extensive parallel corpus of Indic languages publicly available. The languages available in the dataset are Assamese, Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Oriya, Punjabi, Tamil, and Telugu. We have created more than 30,000 sentence pairs in "Hindi written in Roman-English" for now. With train, validation, and test split = 20000, 5000, 8000, respectively. To assess the accuracy of a machine translation system, a metric known as BLEU score [14] is used, which is the standard method for the evaluation of any machine translation model.

5.2 BLEU score Evaluation

The quality of machine-generated translations or text outputs is frequently assessed using the BLEU (Bilingual Evaluation Understudy) score. It measures the similarity between a machine-generated text and one or more reference texts (typically human-generated) [14].

The BLEU score is based on the principle that good translations should contain similar n-grams (contiguous sequences of n words) as the reference translations. The metric calculates precision by comparing n-grams from the machine-generated output with those in the reference translations [14]. The equation for

calculating the BLEU score is as follows:

$$BLEU = BP * exp\left(\sum_{n=1}^N w_n \log P_n\right) \quad (5.1)$$

Where:

BP (Brevity Penalty) : It penalizes the translation if it is much shorter than the reference translation.

$$BP = \begin{cases} 1 & , \text{if candidate length} \geq \text{reference length,} \\ exp\left(1 - \frac{\text{referencelength}}{\text{candidatelength}}\right) & , \text{if candidate length} < \text{reference length,} \end{cases} \quad (5.2)$$

N: The maximum n-gram order considered for precision calculation.

P_n : The precision of n-grams between the candidate translation and the reference translation.

w_n : The weight assigned to each precision score. Typically set to $1/N$, which gives equal importance to all n-grams.

Figure 5.1 below shows an example of the translated sentence. The BLEU scores for different models are shown in the table 5.1. It shows that our proposed method gives a better BLEU score than the other methods.

Source Sentence	Translated Sentence
kya aap aaj raat ka khaana khaenge?	Will you have dinner tonight?
sir, main aapako kal subah uttar doonga	Sir, I will reply you tomorrow morning
main tumhen kal 7 baje lene aaoonga	i will pick you up tomorrow at 7

Figure 5.1: Input sentence and its translated sentence using proposed method

To compare our results, we trained another Bi-LSTM model. However, the main thing to notice here is when we tried to use Facebook’s latest No Language Left Behind (NLLB) model [20], which claims it can translate a sentence from any language to any other language; it fails to generate translated output for our dataset which is evident from figure 5.2 and 5.3, that resulted in the same input as shown in figure 5.2 or in some cases it generates random sentence as shown in figure 5.3 which in no way is connected to the original sentence which means it

can not translate these sentences.

The screenshot shows a web-based translation interface. It has three main sections: 'Source' with a dropdown menu set to 'Hindi', 'Target' with a dropdown menu set to 'English', and 'Input text' with a text area containing the Hindi sentence 'main apko aise hee ek raajaneetigy ke baare mein bataana chaahata hoon'. Below the input text, there are two buttons: '{ } output' and 'copy to clipboard'. The output area displays a JSON object:

```
{
  inference_time: 5.126469135284424,
  source: "hin_Deva",
  target: "eng_Latn",
  result: "main apako aise hee ek baal raajaneetigy ke baare mein
  bataana chaahata hoon"
}
```

Figure 5.2: Snapshot of translated sentences using NLLB [20]

Model	Bleu Score
NLLB [20]	0
Bi-LSTM	8.23
Proposed Model	10.64
Subword + Proposed Model	10.82

Table 5.1: BLEU scores comparison

Source

Hindi

Target

English

Input text

kya aap aaj rat ka khana khaenge?

{-} output copy to clipboard

```
{
  inference_time: 2.753345489501953,
  source: "hin_Deva",
  target: "eng_Latn",
  result: "What is the meaning of the word "happiness"?"
}
```

Figure 5.3: Snapshot of one of the above translated sentences using NLLB [20]

CHAPTER 6

Conclusion and Future Work

The work we have carried out has not been done before, to the best of our knowledge, for which we have even created our dataset. Hence, to compare the result of our method we trained another simple transformer and Bi-LSTM as well. Looking at the BLEU scores in table 5.1, it is evident that using a transformer for machine translation will give better results and using subwords improves accuracy.

The task here shows that the transformer can translate Hindi in Roman letters into English sentences, which is more challenging than translating Hindi sentences written in Hindi translated into English.

One possible application of the work carried out in here is chat applications.

We have created a dataset of around 30000 sentence pairs. However, one can generate more sentence pairs and train the model to get better results.

Similarly, We can generate a dataset for the Gujarati language and train on that.

References

- [1] J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. 1 2016.
- [2] M. R. Costa-jussà and J. A. Fonollosa. Latest trends in hybrid machine translation and its applications. *Computer Speech Language*, 32(1):3–10, 2015. Hybrid Machine Translation: integration of linguistics and statistics.
- [3] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 06–11 Aug 2017.
- [4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.
- [6] F. Informatik, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*, 03 2003.
- [7] N. Jiang, J. Tang, W. Yu, and J. Zhou. Local feature normalization. In *Knowledge Science, Engineering and Management: 14th International Conference, KSEM 2021, Tokyo, Japan, August 14–16, 2021, Proceedings, Part II*, page 228–239, Berlin, Heidelberg, 2021. Springer-Verlag.
- [8] L. u. Kaiser and S. Bengio. Can active memory replace attention? In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

- [9] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. *ArXiv*, abs/1610.10099, 2016.
- [10] T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, 2018.
- [11] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio. A structured self-attentive sentence embedding. 3 2017.
- [12] A. Lopez. Statistical machine translation. *ACM Comput. Surv.*, 40(3), aug 2008.
- [13] M. Osborne. *Statistical Machine Translation*, pages 912–915. Springer US, Boston, MA, 2010.
- [14] 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*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.
- [15] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. 6 2016.
- [16] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. 5 2017.
- [17] G. Ramesh, S. Doddapaneni, A. Bheemaraj, M. Jobanputra, R. AK, A. Sharma, S. Sahoo, H. Diddee, M. J, D. Kakwani, N. Kumar, A. Pradeep, K. Deepak, V. Raghavan, A. Kunchukuttan, P. Kumar, and M. Khapra. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages. 04 2021.
- [18] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.
- [19] S. Sreelekha, P. Bhattacharyya, and D. Malathi. Statistical vs. rule-based machine translation: A comparative study on indian languages. In S. S. Dash, S. Das, and B. K. Panigrahi, editors, *International Conference on Intelligent Computing and Applications*, pages 663–675, Singapore, 2018. Springer Singapore.

- [20] N. Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, and J. Wang. No language left behind: Scaling human-centered machine translation, 2022.
- [21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.