



**DE – 40 (EE)**  
**PROJECT REPORT**  
**“SEMANTIC NEURAL TRANSLATION DEVICE”**

Submitted to the Department of Electrical Engineering  
In partial fulfillment of the requirements for the degree of  
**BACHELOR IN ELECTRICAL ENGINEERING**

Submitted By:

1. Muhammad Minhaj Khokhar
2. Ahmad Shams
3. Malik Muhammad Usama Khan
4. Muhammad Ashfaq

**Project Supervisor:**  
Asst. Prof. Sobia Hayee

**Co-Supervisor:**  
Dr. Muwahida Liaqat

## **DEDICATION**

We are thankful to our Creator Allah Subhana-Watala to have guided us throughout this work at every step of the way. Indeed, we could have done nothing without your priceless help and guidance. Whosoever helped us throughout the course of our Project, whether our parents or any other individual was Your will, so indeed none be worthy of praise but You.

We are profusely thankful to our parents; their efforts and all the prayers were with us during the project. It is also dedicated to our teachers and all the technical staff who really helped us for the completion of this project.

We would also like to express special thanks to my supervisor “Asst. Prof Sobia Hayee” and Co-supervisor “Dr. Muwahida Liaqat” for their help throughout our project.

Finally, we would like to express our gratitude to all the individuals who have rendered valuable assistance to our study.

# CERTIFICATE OF APPROVAL

It is to certify that the project “SEMANTIC NEURAL TRANSLATION DEVICE” was done by NC **Muhammad Minhaj Khokhar**, NC **Ahmad Shams**, NC **Malik Muhammad Usama Khan**, NC **Muhammad Ashfaq** under the supervision of “**Asst. Prof Sobia Hayee**” and co-supervisor “**Dr. Muwahida Liaqat**”.

This project is submitted to **Department of Electrical Engineering**, College of Electrical and Mechanical Engineering (Peshawar Road Rawalpindi), National University of Sciences and Technology, Pakistan in partial fulfilment of requirements for the degree of Bachelors of Engineering in Electrical engineering.

## Students:

### 1- Muhammad Minhaj Khokhar

NUST ID: 265439

Signature: \_\_\_\_\_

### 2- Ahmad Shams

NUST ID: 269251

Signature: \_\_\_\_\_

### 3- Malik Muhammad Usama Khan

NUST ID: 249691

Signature: \_\_\_\_\_

### 4- Muhammad Ashfaq

NUST ID: 268424

Signature: \_\_\_\_\_

## APPROVED BY:

Project Supervisor: \_\_\_\_\_

Date: \_\_\_\_\_

**Asst. Prof Sobia Hayee**

# DECLARATION

We hereby declare that this project report entitled “Semantic Neural Translation Device” submitted to the “Electrical Engineering Department”, is a record of an original work done by us under the guidance of Supervisor “Asst. Prof. Sobia Hayee” and Co-supervisor “Dr. Muwahida Liaqat” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Electrical Engineering.

## Students:

### 1- Muhammad Minhaj Khokhar

NUST ID: 265439

Signature: \_\_\_\_\_

### 2- Ahmad Shams

NUST ID: 269251

Signature: \_\_\_\_\_

### 3- Malik Muhammad Usama Khan

NUST ID: 249691

Signature: \_\_\_\_\_

### 4- Muhammad Ashfaq

NUST ID: 268424

Signature: \_\_\_\_\_

## **COPYRIGHT STATEMENT**

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

## **ACKNOWLEDGEMENTS**

It is a great pleasure of writing a report is acknowledging efforts of many people whose names may not be appear on the title page, but their hard work, friendship, cooperation, and understanding were with us to the production of this report.

We would like to acknowledge the efforts of both our supervisor, Asst. Prof Sobia Hayee and Dr. Muwahida Liaqat for being helpful throughout the project. They guided us throughout and bestowed upon us their knowledge to keep the project streamlined. Their availability and prompt responses to our issues aided us in successfully finishing this job. Our department (Department of Electrical Engineering) is responsible for enabling us to complete this project and produce outcomes.

## ABSTRACT

Machine translation is a challenging task that traditionally involves large statistical models developed using highly sophisticated linguistic knowledge. Neural machine translation is the use of deep neural networks for the problem of machine translation. The use of Machine Translation for the linguistic processing is taking up pace throughout the world. The demand of broader communication devices is increasing considerably since the world has become a single global hub of information. Bearing this in mind, the need for an AI driven solution, to provide this service, has risen exponentially.

“Semantic Neural Translation Device” provides on a semantic neural networks-based device which will use contextual data to interpret the complete meaning in text and translate it from one language to another, in this project, from English Language to German Language due to the availability of datasets online. Translators based on classical Natural Language Processing methods already exist, but the improved translation method, based on what context the text is being written, is still scarce.

This approach is quite new and only a handful of organization and institutions are implementing it on a large scale. The market leaders in this work at the moment would be Google, Facebook, Amazon and Microsoft, but their work is private and not accessible for learning or other purposes. Work still needs to be done to build a state-of-the-art semantic machine which translates a text based on the context it is being used in. The device can be extended to provide translations from any language of the world given the dataset needed for training. This device proves extremely useful to the global organizations and also to the local ones as it will aid in expanding new horizons of market and communication across the globe.

# TABLE OF CONTENTS

<b>DEDICATION.....</b>	<b>ii</b>
<b>CERTIFICATE OF APPROVAL .....</b>	<b>iii</b>
<b>DECLARATION.....</b>	<b>iv</b>
<b>COPYRIGHT STATEMENT.....</b>	<b>v</b>
<b>ACKNOWLEDGEMENTS .....</b>	<b>vi</b>
<b>ABSTRACT.....</b>	<b>vii</b>
<b>TABLE OF CONTENTS .....</b>	<b>viii</b>
<b>TABLE OF FIGURES.....</b>	<b>xi</b>
<b>Chapter 1 .....</b>	<b>1</b>
<b>INTRODUCTION.....</b>	<b>1</b>
1.1 Summary .....	1
1.2 Novelty .....	1
1.3 Project Workings.....	2
<b>Chapter 2 .....</b>	<b>3</b>
<b>LITERATURE REVIEW .....</b>	<b>3</b>
2.1 Natural Language Processing and Machine Learning.....	3
<b>Chapter 3 .....</b>	<b>6</b>
<b>RESEARCH AND TECHNICAL BACKGROUND.....</b>	<b>6</b>
3.1 Machine Learning .....	6
3.2 Deep Learning.....	8
3.2.1 Deep Learning vs Machine Learning.....	8
3.2.2 Deep Learning vs Machine Learning vs Artificial Intelligence.....	9
3.2.3 Deep Layered Network.....	10
3.3 Neural Networks .....	11
3.3.1 Recurrent Neural Networks .....	11
3.4 Natural language Processing .....	15
3.4.1 NLP Tasks.....	15
3.4.2 Neural Machine Translation .....	17
3.5 Python Programming Language.....	19



3.6	Libraries .....	20
3.6.1	NumPy .....	20
3.6.2	PyTorch.....	20
3.6.3	Tensor Flow .....	21
3.7	Transformers .....	22
3.7.1	Pipeline Transformers.....	22
3.8	Cloud Computation .....	23
<b>Chapter 4</b>	<b>.....</b>	<b>25</b>
<b>SOFTWARE IMPLIMENTATION.....</b>		<b>25</b>
4.1	Software Development.....	25
4.1.1	Data Set Selection & Training .....	25
4.1.2	NLP Models .....	27
4.2	Software Complication.....	34
4.2.1	Version Compatibility.....	34
4.2.2	Coding Errors.....	34
4.2.3	Model Compatibility.....	34
4.2.4	Transformers Algorithm .....	35
4.2.5	Finding the perfect libraries for data extraction.....	35
<b>Chapter 5</b>	<b>.....</b>	<b>36</b>
<b>HARDWARE METHODOLOGY AND IMPLIMENTATION .....</b>		<b>36</b>
5.1	Server Development.....	36
5.1.1	Software and Tools Used.....	37
5.1.2	Developing API with Flask and Python.....	38
5.1.3	Inferencing the Software Model for Processing Output .....	41
5.1.4	Local Server Running and Response .....	47
5.2	Remote Client.....	50
5.2.1	Establishing Connection with Server.....	51
5.2.2	Graphical User Interface Data for User Interaction .....	55
5.3	Raspberry Pi Integration.....	57
5.3.1	Use and Specifications of the Raspberry Pi.....	58
5.3.2	Setting Up an Environment for User Interface on Hardware .....	59
5.3.3	Starting User Interface on Bootup .....	61

5.4	Displaying and Evaluation of Translation Results .....	62
5.4.1	Equipment/Hardware Required .....	62
5.4.2	User Input Parameters .....	62
5.4.3	Realtime Translation Response.....	64
<b>Chapter 6</b>	.....	<b>65</b>
<b>RESULTS</b>	.....	<b>65</b>
6.1	Marian MT Model Results .....	65
6.1.1	English to German Results.....	65
6.1.2	German to English Results.....	65
6.2	T5 Model Results .....	66
6.2.1	English to German Results.....	66
6.2.2	German to English Results.....	67
6.3	MBART-50 Model Results .....	67
6.4	Observations.....	67
<b>Chapter 7</b>	.....	<b>68</b>
<b>CONCLUSION AND FUTURE WORK</b>	.....	<b>68</b>
7.1	Future Works and Recommendations .....	68
7.2	Conclusion.....	70
<b>REFERENCES</b> .....		<b>71</b>

# TABLE OF FIGURES

Figure 1 – Project Diagram and Summary .....	2
Figure 2 – Steps of NLP Method .....	5
Figure 3 – Machine Learning and Artificial Intelligence .....	7
Figure 4 – Machine Learning vs Deep Learning .....	8
Figure 5 – Artificial Intelligence vs Machine Learning vs Deep Learning .....	9
Figure 6 – Deep Layered Network Architecture.....	10
Figure 7 – A Simple Neural Network.....	11
Figure 8 – Natural Language Processing circular diagram.....	16
Figure 9 – Neural Machine Translation Architecture .....	18
Figure 10 – Features of Python programming language.....	19
Figure 11 – Cloud Computation Architecture .....	24
Figure 12 – Flow Chart of Software section.....	26
Figure 13 – Data set collection .....	27
Figure 14 – Detailed chart of working of T5 Model.....	29
Figure 15 – Steps of Marian MT model.....	30
Figure 16 – Detailed explanation of MBart 50 .....	31
Figure 17 – Auto tokenizer Explanation.....	32
Figure 18 – T-5 Model.....	32
Figure 19 – Marian MT model.....	33
Figure 20 – Auto tokenizer Model.....	33
Figure 21 – Hardware Implementation pipeline block diagram .....	36
Figure 22 – Basic working diagram of an API .....	38
Figure 23 – Starting a basic Flask Application.....	39
Figure 24 – A minimalistic flask application response.....	39
Figure 25 – Deep Learning Inference Example .....	41
Figure 26 – A machine learning inference system diagram .....	43
Figure 27 – Deep Learning Training Example .....	44

Figure 28 – Deep Learning Training vs Inference Example.....	45
Figure 29 – HTTP request/response .....	47
Figure 30 – Request/Response between User and Web Server .....	49
Figure 31 – A Client-Server System .....	50
Figure 32 – Run-time process of a Server-Client Connection.....	52
Figure 33 – Tkinter based User Interface of the project. ....	56
Figure 34 – Raspberry Pi Board.....	57
Figure 35 – Raspberry Pi Model 3 B+ Hardware .....	58
Figure 36 – Raspberry Pi 3 Model B+ Specifications .....	59
Figure 37 – Raspberry Desktop .....	60
Figure 38 – User entering Input Parameters in the Interface. ....	63
Figure 39 – Translation result on the Interface obtained after the response. ....	64
Figure 40 – Marian MT: English to German translation result .....	65
Figure 41 – Marian MT: English to German translation result .....	66
Figure 42 – T-5 Model: English to German translation result.....	66
Figure 43 – T-5 Model: English to German translation result.....	67
Figure 44 – MBART-50 Model: English to German translation result .....	67

# Chapter 1

## INTRODUCTION

Artificial intelligence is one of the most remarkable technologies that will change people's lives in practically every aspect of modern life. Many enterprises and companies have already seized considerable development prospects provided by this technology. Artificial intelligence (AI) is being used by many businesses to save costs, enhance efficiency, increase revenue, and improve customer experience. Artificial Intelligence is likely to be one of the main developments in retail in the next years, with a market worth \$126 billion and a forecast of \$3 trillion by 2024.

### 1.1 Summary

On the basis of a virtual machine, a Semantic Neural network is presented to develop models of artificial neuron networks. Virtual connections between neurons represent an astral body, whereas the bodies and signals passing via the neurons' connections represent a physical body. Meaning preservation and handling data sparsity (many sentences have a single meaning) are two of the primary reasons that semantic representations are useful for machine translation. Creating this initiative was a top priority for us since we wanted to improve communication and erase language barriers. Using textual data, our semantic Neural network device translates from one language to another and interprets the full meaning of the text.

### 1.2 Novelty

The semantic neural translation model, that is a device, as a means to provide accurate translation results and operated remotely with much hassle. The currently available technologies are available online but their workings are private and not open for public to view and learn from. Our solution gives access to the user to obtain meaningful translation without harming the context of the original message. This device is translating between English and German language because of the availability of online labelled data sets. This model can be used for other languages as well along with regional languages if the labeled data set is available on which the model will train itself from and operate just like for English and German language with good precision and accuracy.

### 1.3 Project Workings

The project workings can be seen in the figure, which is describing the steps involved in this project and its completion.

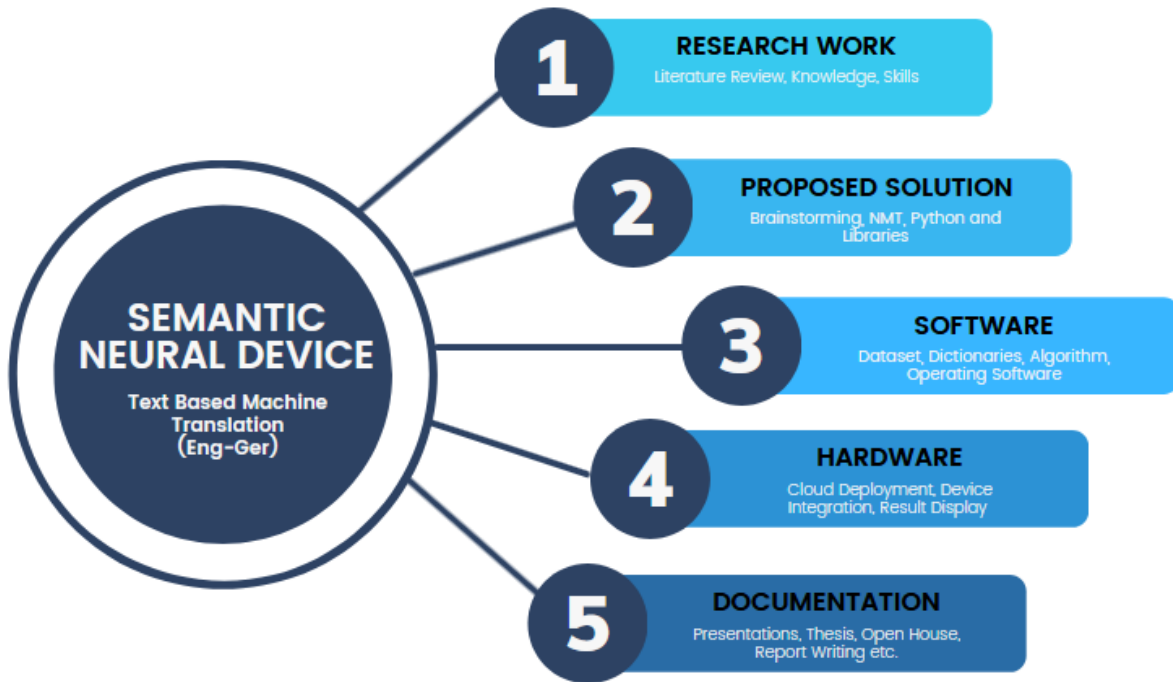


Figure 1 – Project Diagram and Summary

## Chapter 2

### LITERATURE REVIEW

#### 2.1 Natural Language Processing and Machine Learning

Human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data. Homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, variations in sentence structure—these just a few of the irregularities of human language that take humans years to learn, but that programmers must teach natural language-driven applications to recognize and understand accurately from the start, if those applications are going to be useful.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

- Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.
- Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies ‘make’ as a verb in ‘I can make a paper plane,’ and as a noun in ‘What make of car do you own?’
- Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in

the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).

- Named entity recognition, or NEM, identifies words or phrases as useful entities. NEM identifies 'Kentucky' as a location or 'Fred' as a man's name.
- Co-reference resolution is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).
- Sentiment analysis attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.
- Natural language generation is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

If we see, NLP and Machine learning are the exploding fields in section of engineering, there is not a lot of work done on it before. This field is vesting due to its potential and large field of work associated to it.



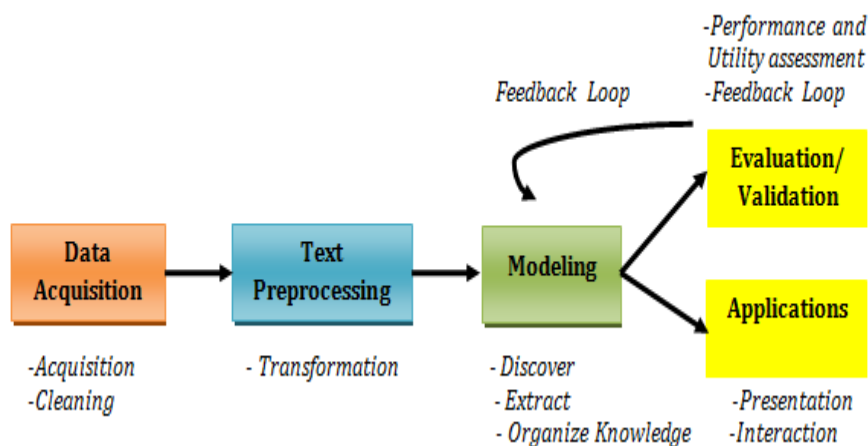


Figure 2 – Steps of NLP Method

### Pervious Work done on Sematic Neural translator

Semantic Neural Translator has been worked on by several people before. The semantic representations ought to be relevant to machine translation, given that the task is to produce a target language sentence with the same meaning as the source language input. Semantic representations formed the core of the earliest symbolic machine translation systems, and have been applied to statistical but non-neural systems as well.

Another German professor Linfeng Song has worked on the AMR based translation on German language itself. According to him, “It is intuitive that semantic representations can be useful for machine translation, mainly because they can help in enforcing meaning preservation and handling data sparsity (many sentences correspond to one meaning) of machine translation models. On the other hand, little work has been done on leveraging semantics for neural machine translation (NMT). In this work, we study the usefulness of AMR (short for abstract meaning representation) on NMT. Experiments on a standard English-to-German dataset show that incorporating AMR as additional knowledge can significantly improve a strong attention-based sequence-to-sequence neural translation model.”

## Chapter 3

# RESEARCH AND TECHNICAL BACKGROUND

### 3.1 Machine Learning

It's called machine learning when a computer does something without being explicitly instructed. It has been a decade since machine learning has made it possible for self-driving cars, a more accurate understanding of the human genome, and a more successful web search. It's likely that you utilize machine learning hundreds of times a day without ever realizing it. It has been widely accepted as the most successful method for getting closer to human-level artificial intelligence.

In the field of machine learning, there are three main categories:

1. Supervised learning (parametric and non-parametric algorithms, support vector machines, kernels, and neural networks).
2. Unsupervised learning (clustering, dimensionality reduction, recommender systems, deep learning).
3. Reinforced Learning (bias/variance theory; machine learning and AI innovation process)

#### Evolution of Machine Learning

Now that machine learning is so widely used, you probably utilize it millions of times per day without even realizing it! Because of developments in computing technology, machine learning is no longer the same as it was in the past. Pattern recognition and the thought that computers could learn to perform specific tasks without being explicitly trained to do so motivated researchers in the field of artificial intelligence to investigate this possibility. As new data is fed into the models, the iterative nature of machine learning becomes increasingly important. They build on previous computations to produce predictable results and judgments. It's not a new science, but it's picking up steam.

## Requirements for a Strong Machine Learning System

1. Capabilities for data preparation
2. Algorithms; basic and advanced
3. Iterative processes and automation
4. Scalability
5. Ensemble modelling

## Machine Learning and Artificial Intelligence

The fields of artificial intelligence and machine learning have a lot in common. When it comes to creating intelligent systems, these two technologies are the most often used.

No matter how often they are used interchangeably, they are two distinct technologies, despite the fact that they are related.

On a broad level, we can distinguish AI and ML. Artificial Intelligence (AI) is a subset of machine learning that enables machines to learn from data without being explicitly instructed.

The goal of artificial intelligence (AI) is to create machines that can think and act like humans.

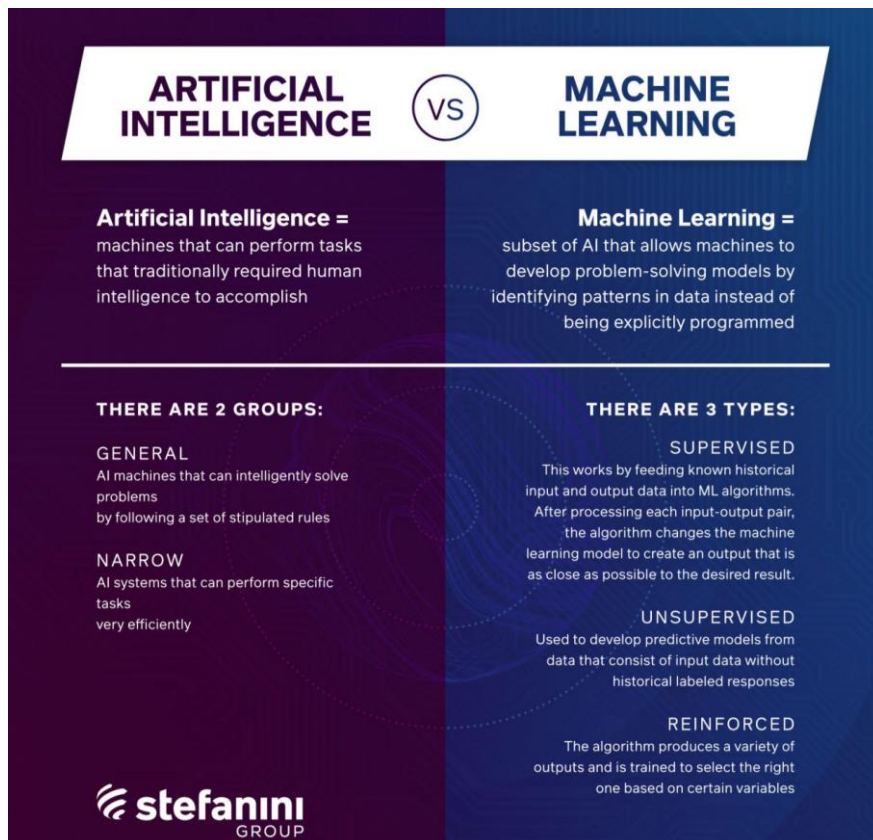


Figure 3 – Machine Learning and Artificial Intelligence

## 3.2 Deep Learning

The ability to learn several levels of abstraction for data representations is enabled by deep learning in computational models with numerous processing layers. A range of domains, including speech recognition, visual object recognition, object detection, and drug development and genomics have greatly benefitted from these techniques. Using backpropagation, deep learning shows how a machine should alter its internal parameters that are required to compute the representation in each layer from the representation in the previous layer, showing subtle structure in vast data sets. Recurrent networks have shed light on sequential data like text and speech while deep convolutional nets revolutionized picture, video, voice, and audio processing.

It is common to apply deep learning (DL) in these types of applications. The term "representation learning" refers to DL (RL). Research in deep and distributed learning has been spurred by both the unpredictability of data availability and hardware advancements, such as High Performance Computing (HPC). As with its predecessors, Deep Learning is built on top of a standard neural network. In addition, DL makes use of both transformations and graph technology to build multi-layer learning models. A wide range of applications, including audio and speech processing, visual data processing, and natural language processing (NLP), have benefited from the most recent deep learning techniques.

### 3.2.1 Deep Learning vs Machine Learning

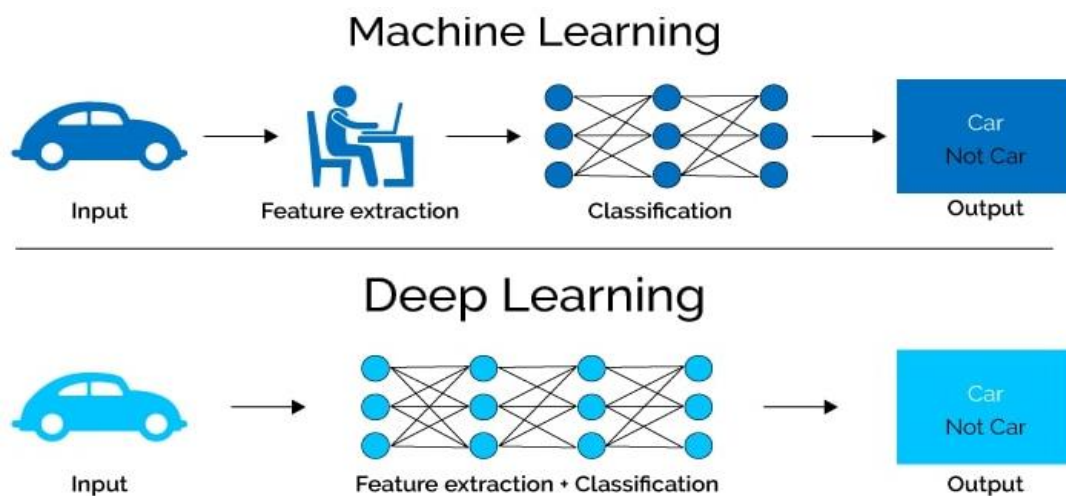


Figure 4 – Machine Learning vs Deep Learning

### 3.2.2 Deep Learning vs Machine Learning vs Artificial Intelligence

In computing, the term "artificial intelligence" (AI) refers to the process of making machines intelligent. A subset of AI known as Machine Learning (ML) is often used in conjunction with AI. ML refers to an AI system that can self-learn using an algorithm. Automated systems that learn over time without human intervention are referred to as machine learning (ML). There are many other types of machine learning (ML) techniques, but deep learning (DL) is the most commonly employed to analyze large volumes of data. Most AI projects employ ML because intelligent behavior demands a lot of knowledge.

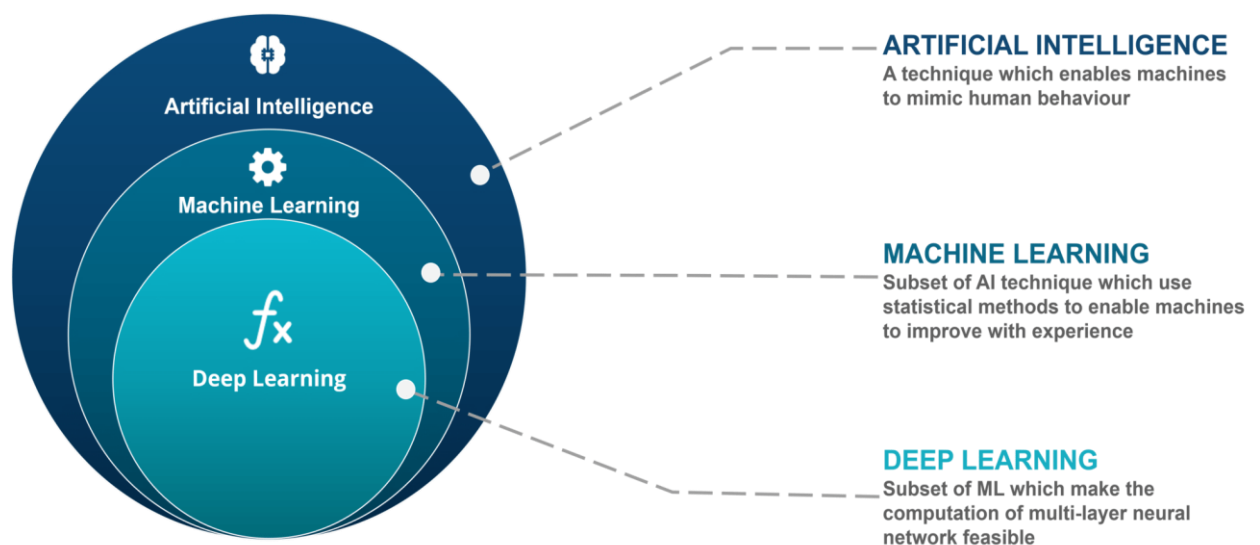


Figure 5 – Artificial Intelligence vs Machine Learning vs Deep Learning

### 3.2.3 Deep Layered Network

Deep neural networks combine the results of many layers of processing. Denoting the output of hidden layers by  $h^{(l)}(x)$ , the computation for a network with  $L$  hidden layers is:

$$f(\mathbf{x}) = f [ \mathbf{a}^{(L+1)} ( \mathbf{h}^{(L)} ( \mathbf{a}^{(L)} ( \dots ( \mathbf{h}^{(2)} ( \mathbf{a}^{(2)} ( \mathbf{h}^{(1)} ( \mathbf{a}^{(1)} (\mathbf{x})))))) ) ) ]$$

Each pre-activation function  $\mathbf{a}^{(l)}(x)$  is usually a linear operation with a matrix  $\mathbf{W}^{(l)}$  and a bias  $\mathbf{b}^{(l)}$  that may be integrated into a parameter  $\boldsymbol{\theta}$ ;

$$\mathbf{a}^{(l)}(\mathbf{x}) = \mathbf{W}^{(l)}\mathbf{x} + \mathbf{b}^{(l)},$$

$$\mathbf{a}^{(l)}(\hat{\mathbf{x}}) = \boldsymbol{\theta}^{(l)}\hat{\mathbf{x}}, \quad l = 1$$

$$\mathbf{a}^{(l)}(\hat{\mathbf{h}}^{(l-1)}) = \boldsymbol{\theta}^{(l)}\hat{\mathbf{h}}^{(l-1)}, \quad l > 1$$

The "hat" notation  $\hat{x}$  denotes the addition of 1 to the vector  $x$ . The form of hidden-layer activation functions  $h^{(l)}(x)$  is generally the identical at each level, but it is not required.

An example network is shown in Figure. Unlike graphical models like Bayesian networks, where hidden variables are random variables, the hidden units in this model are intermediate deterministic calculations, which is why they aren't shown as circles. However, the output variables are drawn as circles because they can be formulated probabilistically.

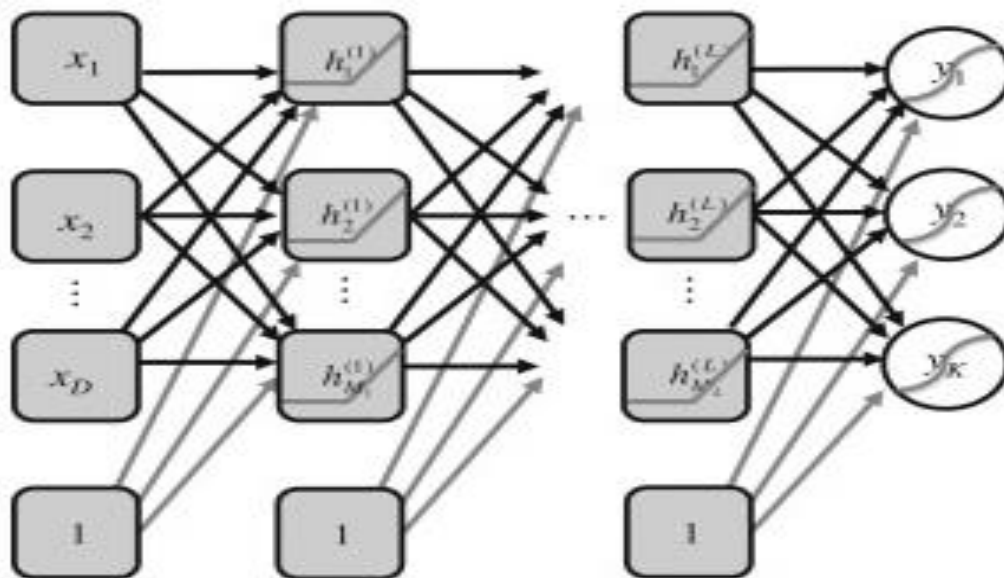


Figure 6 – Deep Layered Network Architecture

### 3.3 Neural Networks

There are many algorithms that may be used to create neural networks, which are designed to imitate the human brain's ability to recognize patterns in large amounts of data. Neuronal networks are systems of neurons that can be either organic or artificial in nature, depending on the context of use.

So long as the output criteria remain the same, neural networks produce the best possible results without having to change them. Neural networks, a concept derived from artificial intelligence, are becoming increasingly popular in the design of trading systems.

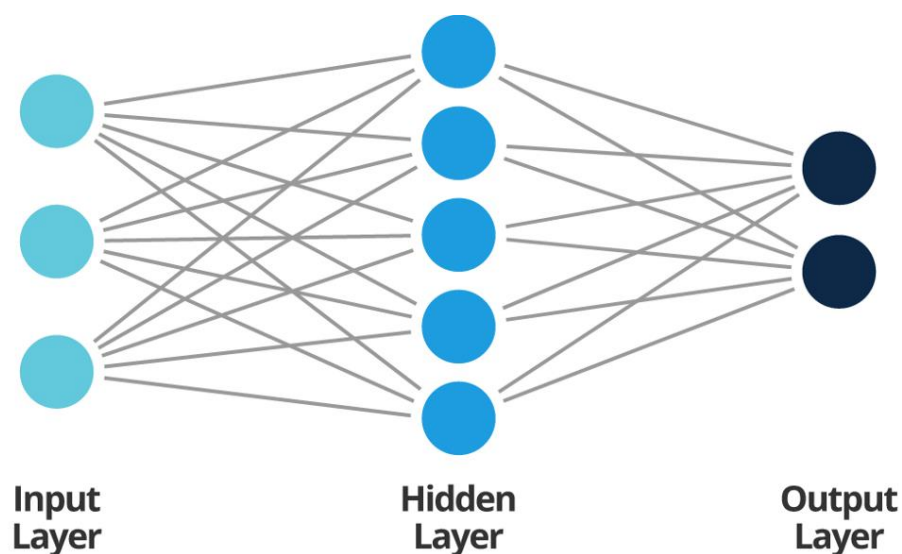


Figure 7 – A Simple Neural Network

#### 3.3.1 Recurrent Neural Networks

Time series or sequential data are the focus of RNNs, a type of artificial neural network. Many prominent apps, such as Siri, voice search, and Google Translate, use these deep learning algorithms for ordinal or temporal difficulties including language translation, natural language processing (nlp), speech recognition, and image captioning. Feedforward and convolutional neural networks (CNNs) are examples of recurrent neural networks. "memory" permits them to influence current input and output by incorporating knowledge from prior experiences. The output of recurrent neural networks is dependent on the previous elements of the sequence, whereas standard deep neural networks assume that inputs and outputs are independent of one another. Unidirectional recurrent neural networks cannot account for future occurrences in their forecasts, even if they may be useful in defining the sequence's output.

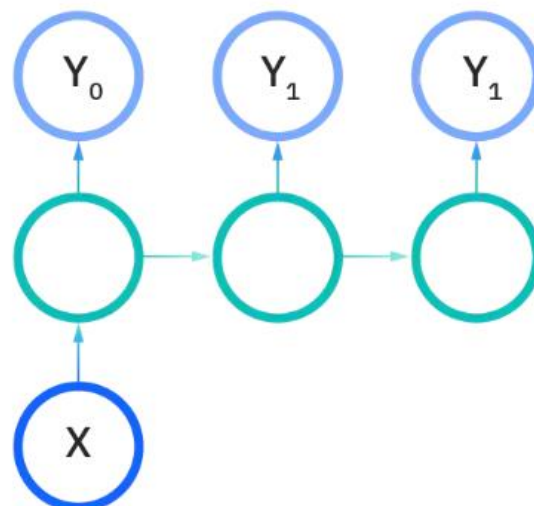
There are a few types of RNNs, they are described below:

Recurrent neural networks, which are illustrated in the diagrams above, do not have this constraint, although feedforward networks do. RNNs are used for a wide range of tasks, including music composition, sentiment classification, and machine translation, and their input and output lengths can vary. A variety of RNN diagrams are often used to illustrate the many sorts of RNNs:

1. One-to-One;

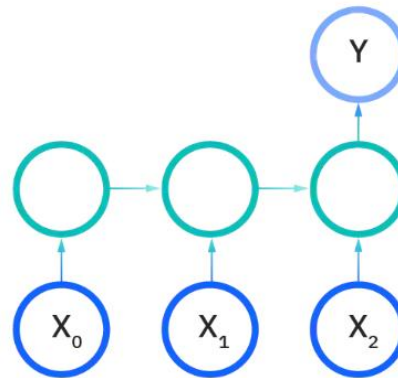


2. One-to-Many;

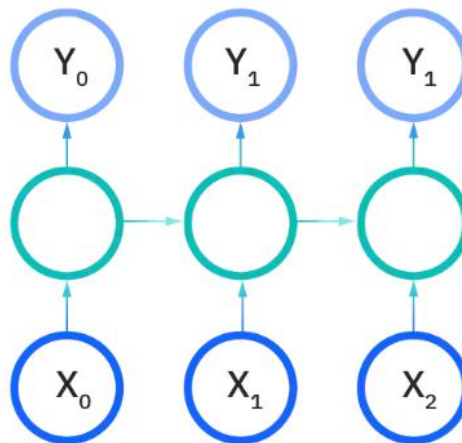




3. Many-to-One;



4. Many-to-Many;



## **Recurrent Neural Networks vs Neural Network**

In reality, neural networks (NN) aren't even algorithms in and of themselves. For the most part, these are a collection of algorithms that can be used together to address a certain problem.

Neurons and connections are the basic building blocks of a neural network. Functions like neurons have several inputs and a single output. When it receives an input, it performs a function on all of the numbers, then returns the result.

Similar to the NN, the RNN has a similar design. As a result of the inclusion of memory, RNNs are distinct from other neural networks. Unlike a feedforward NN, certain levels feed back into the inputs of the preceding layer. For example, it is possible to analyze sequential data such as music, text, or speech files using this functionality instead of a traditional neural network. The RNN, on the other hand, is not constrained by this limitation.

Feedback loops can be used to help students acquire concepts based on context because of the presence of reverse-direction links across layers. What this can do for a deep learning application is highly reliant on the requirements (or goal) and the data.

Time-series data with recurrent patterns can be recognized and exploited by the RNN. Patterns in which prior and subsequent tokens are recognized are given more weight, rather than examining them individually. Consider, for instance, a system that is learning to recognize spoken English. It would be tremendously helpful for such a system if it could predict the next sentence based on recently spoken words.

## 3.4 Natural language Processing

Natural language processing tries to construct robots that interpret and respond to text or voice input in the same manner that people do—and respond with text or speech of their own. Natural language processing (NLP) is an area of artificial intelligence (AI) that deals with the ability of computers to comprehend text and spoken words in the same way that humans can. Statistical, machine, and deep learning models are integrated with computational linguistics (rule-based human language modelling) in NLP. These technologies, when used together, allow computers to analyze human language in the form of text or speech data and 'understand' its full meaning, including the speaker's or writer's intent and sentiment.

NLP is used to power computer programmes that translate text from one language to another, respond to spoken requests, and swiftly summarize vast amounts of material—even in real time. NLP can be seen in voice-activated GPS systems, digital assistants, speech-to-text dictation software, chatbots for customer service, and other consumer conveniences. However, NLP is rapidly being employed in corporate solutions to assist firms streamline operations, enhance employee productivity, and simplify mission-critical business processes.

### 3.4.1 NLP Tasks

Several NLP operations help the machine grasp what it is absorbing by breaking down human text and vocal input in ways that the computer can interpret. These are some of the duties you'll be expected to perform:

1. **Speech recognition** often known as "speech-to-text," is a process that converts spoken input into text. Speech recognition is required for any programme that follows voice commands or answers to spoken enquiries. Slurring, varying emphasis and intonation, as well as incorrect grammar make it difficult for voice recognition software to understand people's speech.
2. **Grammatical Tagging** or part of speech tagging, is a method of determining the part of speech of a word or piece of text based on its use and context. A noun and a verb are used interchangeably in 'I can create a paper plane,' and 'What make of car do you own?' in the context of speech.

3. **Word Sense Disambiguation**, when a word has multiple meanings, disambiguation is the act of identifying which word makes the most sense in the current context by utilizing a semantic analysis technique. Through word sense disambiguation, one may tell the difference between “make the grade” (achieve) and say, “make a bet” (put a wager), for example (place).
4. **NEM (named entity recognition)** is a mechanism for identifying words or sentences as useful entities. According to NEM, 'Kentucky' is a state, and 'Fred' is a person's name.
5. **Co-Reference Resolution:** When two terms are used to describe the same thing, the task of co-reference resolution is called in. Pronouns are the most basic example of this, but it can also include spotting an idiom or metaphor in the text (e.g., when the word "bear" is used to describe a human rather than an animal).
6. **Sentiment Analysis:** Attitudes and feelings, such as sarcasm and confusion, are all part of the subjectivity that can be extracted from text using sentiment analysis.
7. **Natural language generation** is the antithesis of voice recognition or speech-to-text in that it converts structured data into human language.

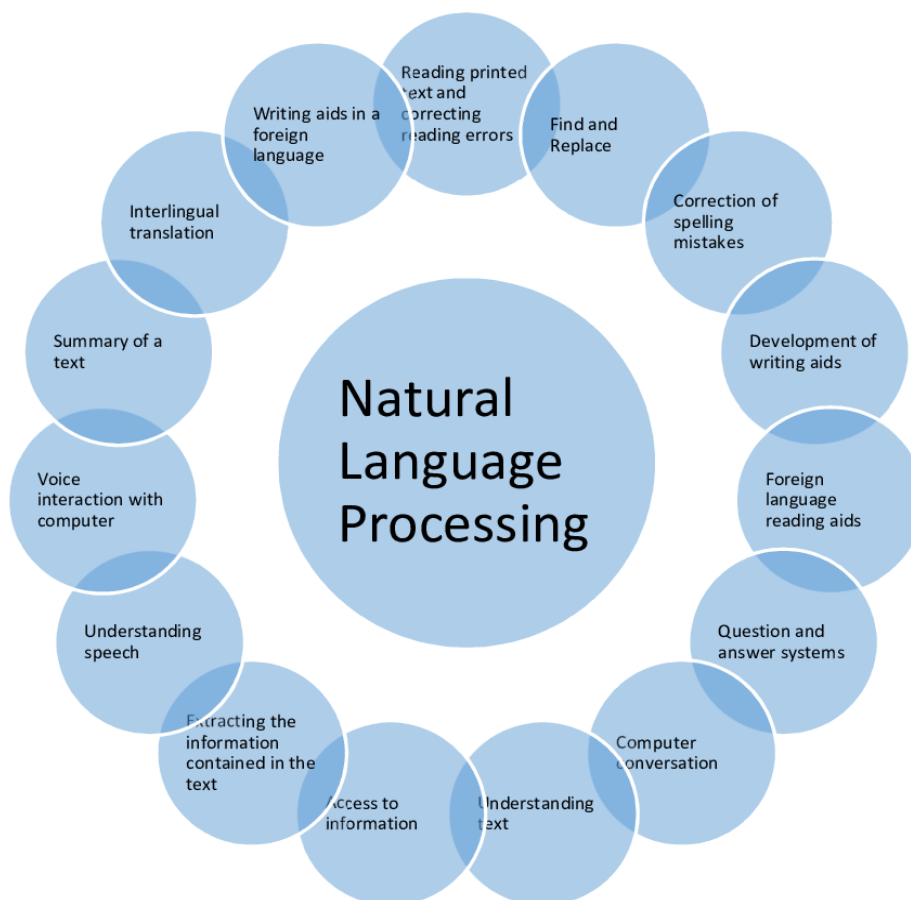


Figure 8 – Natural Language Processing circular diagram

## **3.4.2 Neural Machine Translation**

NMT is a sort of machine translation that uses an artificial neural network to predict the likelihood of a sequence of words, often modelling whole sentences in a single integrated model. Using deep learning for speech recognition was a pioneering achievement in the late 1980s and early 1990s. Neural networks have been used in machine translation since 2014, when the first scholarly paper on the topic was published. For the first time, an NMT system competed in WMT'15; the next year, 90% of NMT systems won. [1]

To make information from the worldwide patent system more accessible, the European Patent Office has been utilizing neural machine translation since 2017. Together with Google, the technology has been created to work with 31 different languages and over nine million papers have been translated as of 2018. [2]

### **3.4.2.1 Workings of NMT**

Individually produced subcomponents distinguish NMT from phrase-based statistical approaches. As with statistical machine translation, NMT does not represent a significant change from previous approaches (SMT). The essential shift is to express words and internal states using vector representations ("embeddings," "continuous space representations"). Phrase-based models have a more complex structure. For any language, there is only one sequence model that predicts the order of a single word. However, this prediction is based on the entire source sentence as well as the previously generated target sequence. Several NMT models employ deep learning and representational learning techniques.

A recurrent neural network (RNN) was initially used to represent word sequences (RNN). Using a bidirectional recurrent neural network, the neural network encodes a source phrase to be decoded by another RNN, which is called a decoder. Recurrent neural networks have a hard time encoding long inputs into a single vector because of this. For example, a decoder can focus on different parts of the input while producing each word of the output to compensate for this. Other coverage models address difficulties with attention approaches, such as ignoring past alignment information, which results in over- and under-translations.

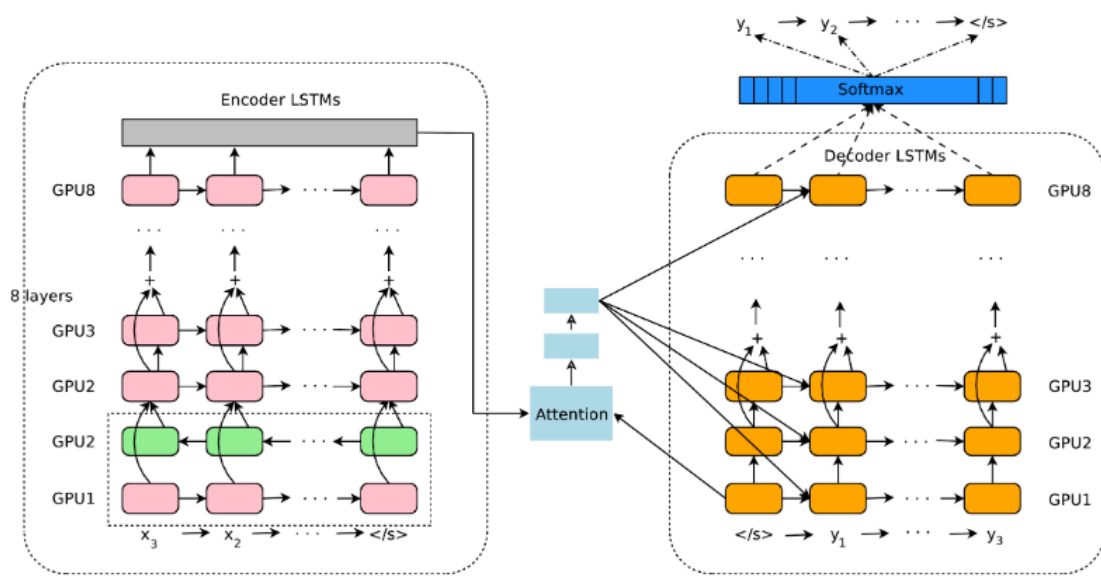


Figure 9 – Neural Machine Translation Architecture [1]

### 3.5 Python Programming Language

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. Some of the major applications of Python include;

- Web Development (server-side)
- Software Development
- Mathematics
- System Scripting

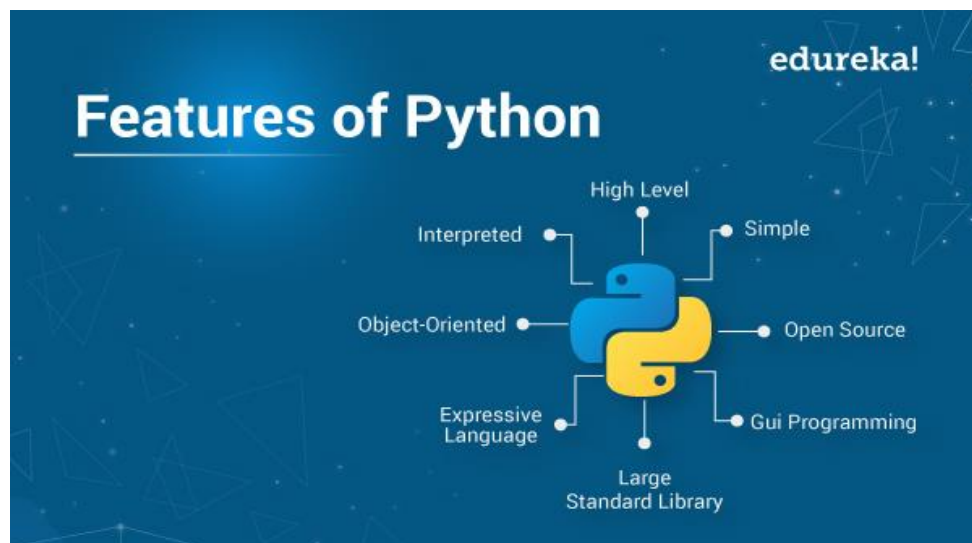


Figure 10 – Features of Python programming language

#### Python Merit Points

1. Python can run on a wide range of platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
2. There are several similarities between the grammar of Python and that of English.
3. In comparison to other programming languages, Python's syntax allows programmers to write programmes in less lines.
4. As an interpreter language, Python allows for immediate execution of newly written code. Prototyping can be done fast as a result of this.
5. Procedural, object-oriented, and functional approaches are all viable options for learning Python in its many forms.

## **Python Syntax Comparison**

1. Python was designed with readability in mind and has a mathematical flavor with some resemblance to English.
2. Instead of semicolons or parentheses, Python uses new lines to complete commands, unlike other programming languages.
3. Python makes use of indentation and whitespace to indicate the scope of things like loops, functions, and classes. For this reason, curly brackets are frequently employed in different programming languages.

## **3.6 Libraries**

A library can be any of the following:

An object-oriented programming library is a collection of data structures and methods that may be accessed through a program's source code. It's common for these libraries to be generated by the same developers that developed the programming language.

### **3.6.1 NumPy**

NumPy is a Python module for creating powerful arrays. NumPy splits and manipulates data conveniently for machine learning. In other words, machine learning uses a range of techniques to analyze data sets. A single loop through each item in these data sets is complex and time intensive due to the fact that they can contain tens of thousands of values.

### **3.6.2 PyTorch**

Machine learning framework PyTorch, created primarily by Meta AI for applications like computer vision and NLP, is an open-source machine learning framework based on the Torch library. The Modified BSD license is used to distribute this open-source software. C++ is also supported by PyTorch. However, the Python interface has been deemed the most stable and efficient.

Just a few examples of deep learning applications developed on PyTorch include Tesla Autopilot, Uber Pyro, Hugging Face Transformers, PyTorch Lightning, and Catalyst. High-



level capabilities of PyTorch includes Deep neural networks with an independent tape differentiation approach for tensor computation (like NumPy) and high GPU acceleration.

### **3.6.3 Tensor Flow**

A Google framework called TensorFlow makes it easy to build and train neural networks and machine learning models. Using this data, you may build and train machine learning models. Machine learning platform TensorFlow is free and open source. Machine learning researchers and developers alike can take advantage of a wide range of open-source tools, frameworks, and community resources that make it easy to build and deploy ML systems.

## **3.7 Transformers**

Using the self-attention technique, a transformer is a deep learning model that assigns a distinct weight to each piece of incoming data. It is widely used in computer vision and natural language processing (NLP).

In tasks like translation and text summarization, transformers such as recurrent neural networks (RNNs) can be used to analyze sequential input data, such as natural language. RNNs, on the other hand, only process a portion of the input at a time. The attention mechanism provides context at any point in the input stream. The transformer does not have to parse each word if the incoming input is a natural-language phrase. Compared to RNNs, this allows for more parallelization, which speeds up training time.

In 2017, a team at Google Brain produced Transformers, which have now replaced RNN models like long short-term memory for NLP challenges (LSTM). The ability to train on larger datasets is made possible by the enhanced parallelization of training. Thus, methods like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) were developed and may be fine-tuned for specific purposes using large language datasets such as Wikipedia Corpus and Common Crawl. [3]

### **3.7.1 Pipeline Transformers**

When it comes to using models, inference pipelines are a simple and effective method. They provide an easy-to-understand API for tasks including Named Entity Recognition (NER), Masked Language Modeling (MLM), Sentiment Analysis (SA), Feature Extraction (FE), and Question Answering (QA).

## 3.8 Cloud Computation

One way that numerous services can be delivered online is by using cloud computing. Just some of the tools and applications accessible include data storage, servers, databases, networking, and software.

"Cloud computing" refers to the fact that the data is stored in a remote location in the cloud or a virtual environment. Clients can store their files and apps on faraway servers, then access them over the Internet via cloud services. Allowing users to work from any location, this indicates that they don't need an internet connection. Cloud computing removes all of the data crunching and processing work from the device you carry about or sit at. Computer clusters in cyberspace handle all of the work. The Internet has made it possible for you to access all of your data, work, and apps via the cloud.

There is the option of a public or a private cloud. Public cloud service companies offer their products and services through the Internet in exchange for a fee. Unlike public cloud providers, private cloud service providers limit the number of users who can access their services. Hosted services are provided through a network-based infrastructure. There is also a hybrid option that combines public and private services.

There are a few important things to know about cloud computing.

1. The distribution of various services through the Internet, such as data storage, servers, databases, networking, and software, is known as cloud computing.
2. Cloud-based storage makes it possible to store and retrieve files from a remote location.
3. There are both free and paid public services available online, with the latter being housed on a network for specific customers.

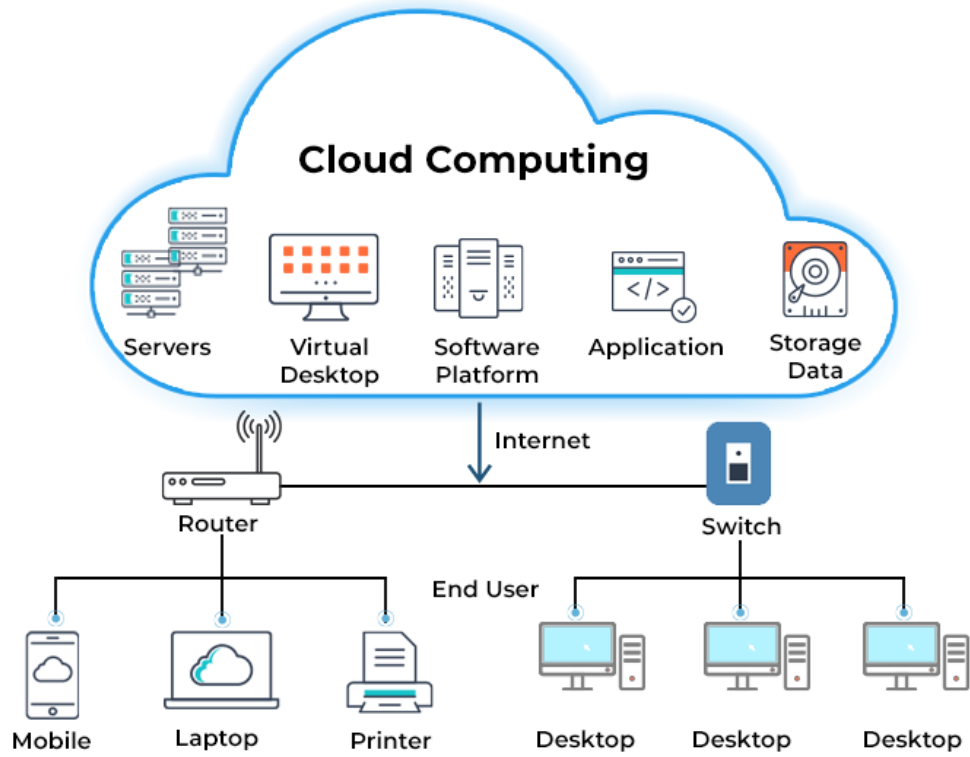


Figure 11 – Cloud Computation Architecture

## Chapter 4

# SOFTWARE IMPLEMENTATION

### 4.1 Software Development

The software part includes the coding and ML-Based data training to execute our program.

The software part is subdivided into further part as per series of actions, which are mentioned as follows.

1. Data Set Selection & Training
2. NLP models
3. Algorithm Designing

#### 4.1.1 Data Set Selection & Training

A dataset in machine learning is, quite simply, a collection of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes. This means that the data collected should be made uniform and understandable for a machine that doesn't see data the same way as humans do. For this, after collecting the data, it's important to preprocess it by cleaning and completing it, as well as annotate the data by adding meaningful tags readable by a computer [4].

For choosing a dataset of our own very program we have chosen a pre-made data set of most common 1500 words in both languages, German and English. These words are most common words used in both languages, while the data set is extracted from an open source available at web and accessible for everyone.

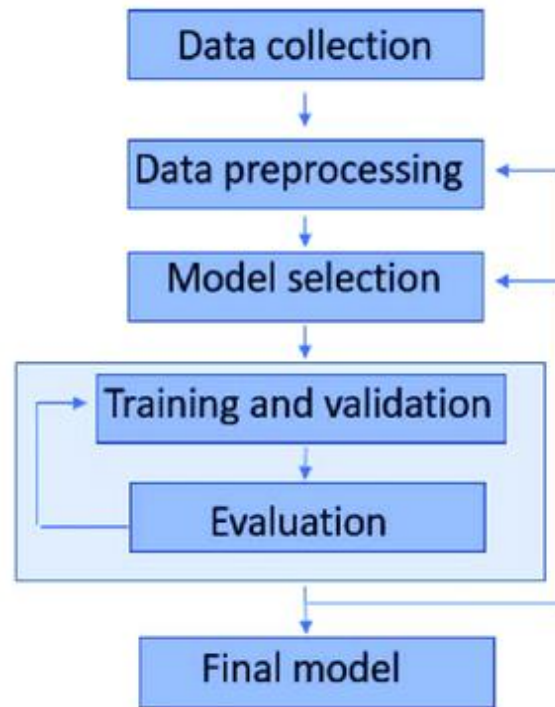


Figure 12 – Flow Chart of Software section

Figure 12, explains the steps which are been followed in software part in order to get the desired output. For our program we are using pre-established data set to save the time and execute our program more vigorously.

Large data sets are avoided to tackle the difficulty of data training and to avoid massive running time. The large data would require a hard development desktop and processor for smooth performance, we are right now trying to make it slim and more convenient for users across cloud with lower space of program. The lower size program would also benefit the low-end computers and laptops to use the translator more conveniently.

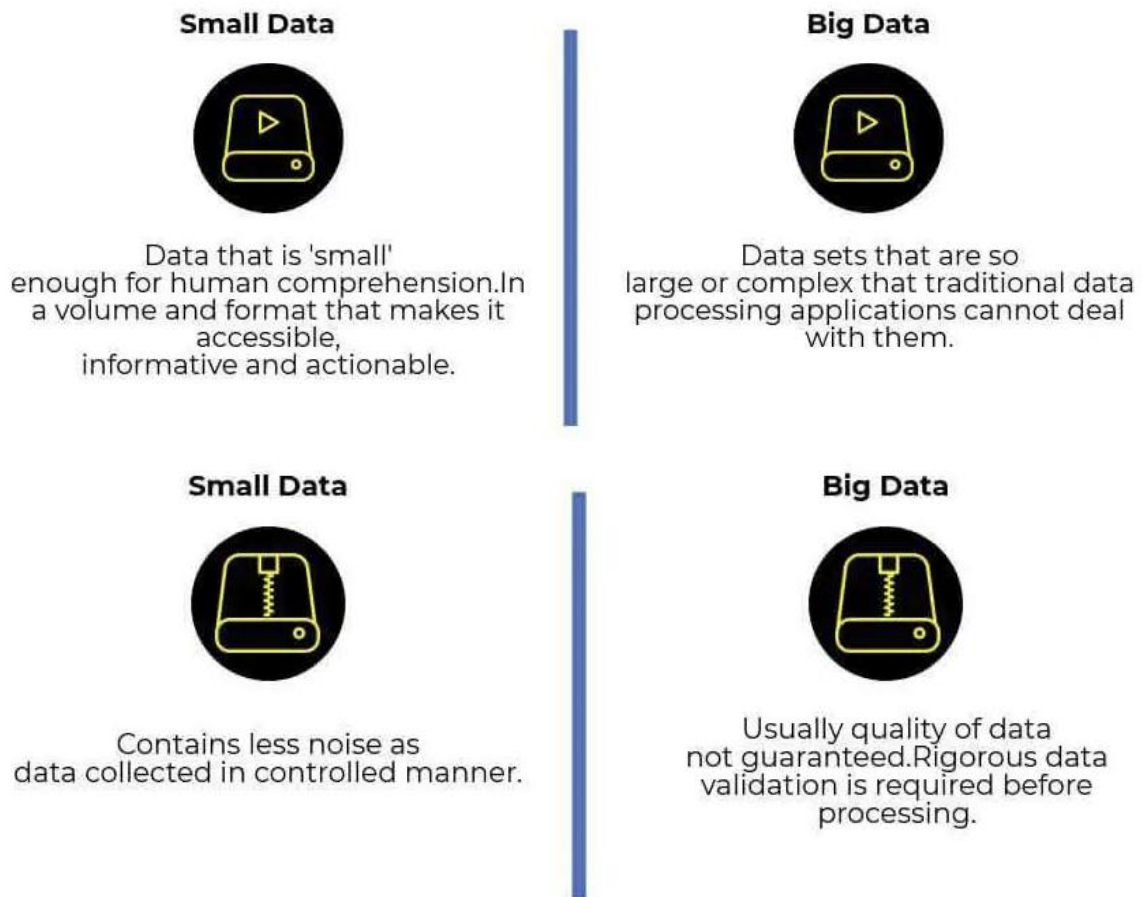


Figure 13 – Data set collection [5]

#### 4.1.2 NLP Models

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment. [6]

The NLP models, we used for this translator are in quantity of 5, the models are listed below:

1. Marian MT German to English
2. Marian MT English to German
3. T-5 Model
4. MBart 50
5. Auto Tokenizer

We will be going through in detail for the above listed models one by one.

#### **4.1.2.1 T-5 Model**

Recently, there are lots of transfer learning techniques for NLP. But some techniques may work almost identical — just with different datasets or optimizers — but they achieve different results, then can we say the technique with better results is better than the other? Given the current landscape of transfer learning for NLP, Text-to-Text Transfer Transformer (T5) aims to explore what works best, and how far can we push the tools we already have. [7]

Many tasks are cast into this framework: machine translation, classification task, regression task (for example, predict how similar two sentences are, the similarity score is in range 1 to 5), other sequence to sequence tasks like document summarization (for example, summarizing articles from CNN daily mail corpus). [7]



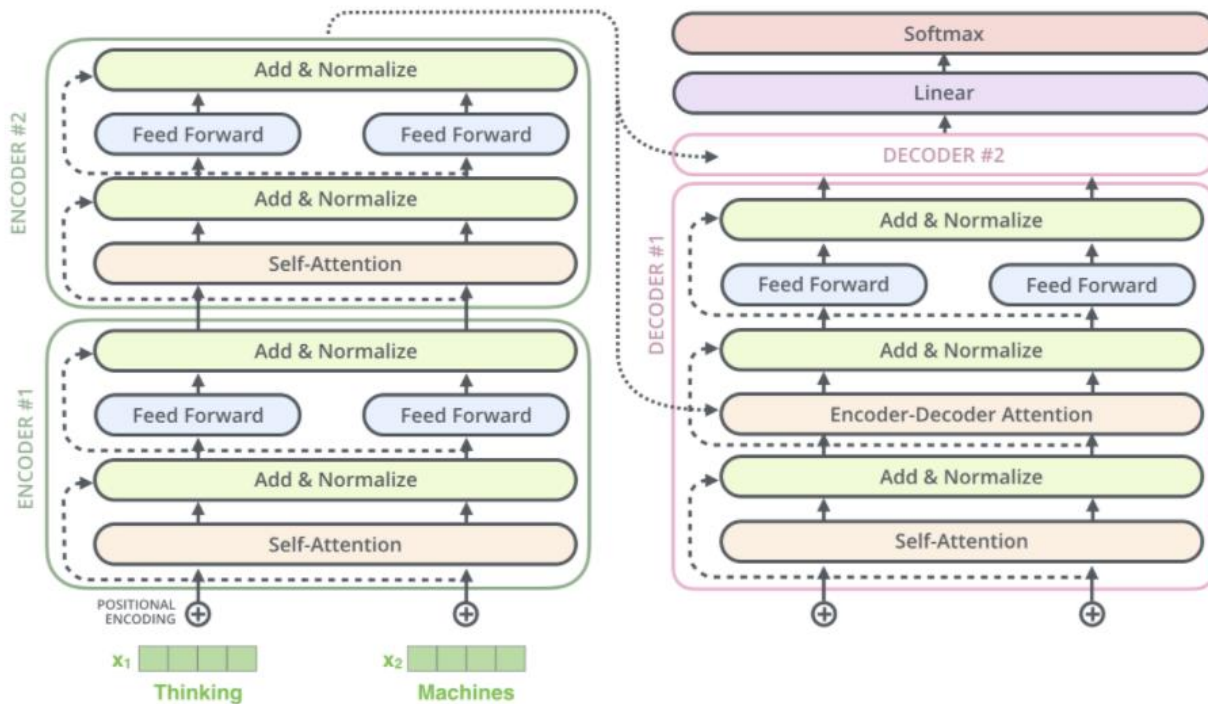


Figure 14 – Detailed chart of working of T5 Model

#### 4.1.2.2 Marian MT Models

This is the framework that will be used for translating the original text to a target language and vice-versa as stated in the first two steps of the previous section.

Marian MT is an efficient Machine Translation framework. It uses the MarianNMT engine under the hood, which is purely developed in C++ by Microsoft and many academic institutions such as the University of Edinburgh, Adam Mickiewicz University in Poznań. The same engine is currently behind the Microsoft Translator service. [8]

The NLP group from the University of Helsinki open-sourced multiple translation models on Hugging Face Transformers. Marian MT is one of those models previously trained using Marian on parallel data collected at Opus [8].

All model names from Hugging Face use the following format Helsinki-NLP/opus-mt-{src}-{tgt} where {src} and {tgt} correspond respectively to the source and target languages. Their value is a two digits code representing the language. You can find here the list of the language codes. For our case, we will have two candidate models.

The configuration of each model requires two main classes, Marian Tokenizer used to load the right tokenizer and Marian MT Model used to load the pre-trained model.

1. Helsinki-NLP/opus-mt-en-de the first model translating German to English.
2. Helsinki-NLP/opus-mt-en-de, the first model translating English to German

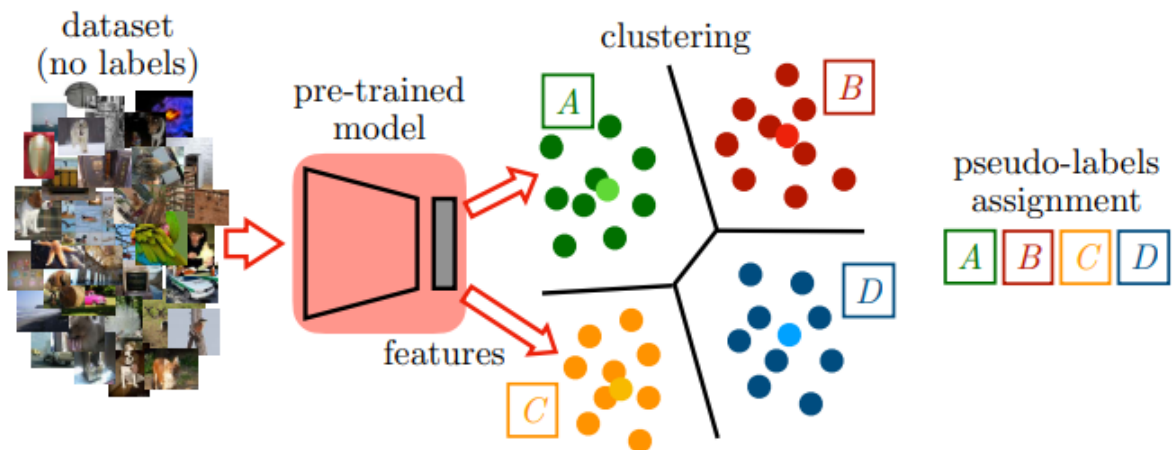


Figure 15 – Steps of Marian MT model

#### 4.1.2.3 MBART 50

MBART is useful also to be fine-tuned on language pairs with at least one unseen language. The results, though, are not as good as for the pretraining languages. The degradation is particularly bad when the unseen language is on the source side.

Thus, the pretraining languages constrain the possible downstream tasks to those languages. Then, in order to use MBart effectively with new languages, it is important to have more monolingual data that can be added to the pretrained model [9]. However, given the computational and time resources needed for training it, it would be nice not to start it from scratch.

To overcome this problem, the authors propose to extend the final mBART25 checkpoint with 25 additional languages, effectively doubling their amount. The procedure is conceptually simple but effective. They start by adding 25 randomly-initialized language tokens in the

embedding layers, one for each new language. Then, they concatenate the monolingual training data of the original 25 languages with the data of the 25 new languages and use the resulting training set to continue the training of the saved checkpoint. They also reuse the same sentence piece model for word segmentation, as it was trained on 100 languages and not only on the original 25. Also, the authors perform some data filtering for both preventing any overlap between train and dev/test sets, and filtering out sentences that are not recognized as in the correct language by fast text [9]

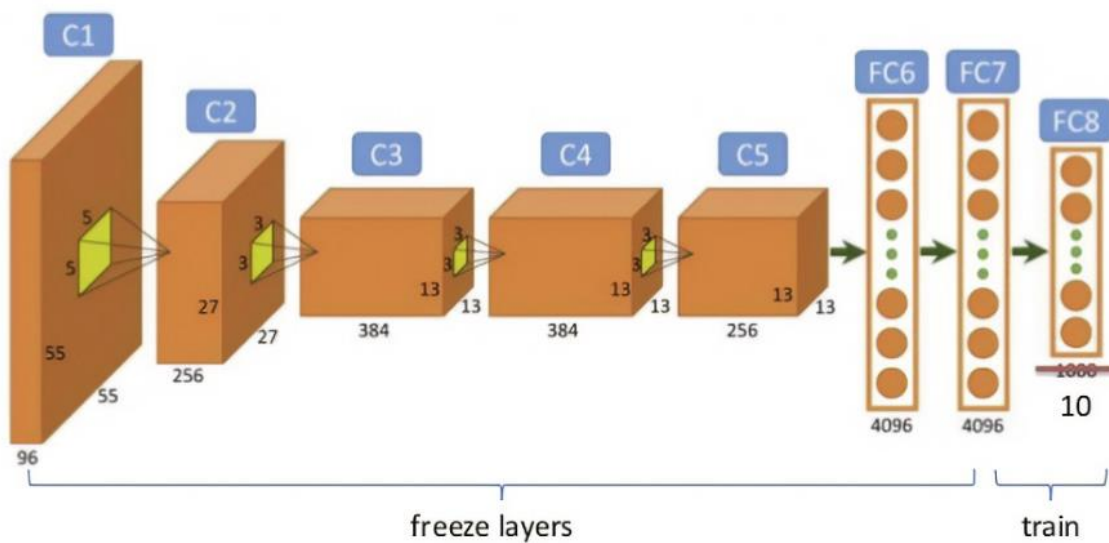


Figure 16 – Detailed explanation of MBart 50

#### 4.1.2.4 Auto Tokenizer

A tokenizer is in charge of preparing the inputs for a model. The library contains tokenizers for all the models. Most of the tokenizers are available in two flavors: a full python implementation and a “Fast” implementation based on the Rust library for Tokenizers. The “Fast” implementations allow:

1. A significant speed-up in particular when doing batched tokenization
2. Additional methods to map between the original string (character and words) and the token space (e.g., getting the index of the token comprising a given character or the span of characters corresponding to a given token). [10]

# Tokenization



Figure 17 – Auto tokenizer Explanation

## T5 En to De

```

> text='The world is not Fair.The sense that I can derive from the world is that it is ever changing and yet all the same throu
model_name = 't5-small-finetuned-en-to-de'
tokenizer = AutoTokenizer.from_pretrained('t5-small')
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
translated = model.generate(**tokenizer(text, return_tensors="pt", padding=True),max_length=1024)
trans_text=[tokenizer.decode(t, skip_special_tokens=True) for t in translated]
print(*trans_text)

```

Die Welt ist nicht Fair.Ich kann deren Gefühl vermitteln, dass die Welt immer verändert und zugleich immer wieder ist.

Figure 18 – T-5 Model

## Preprocessing Text and Tokenizing text data for FineTuning

```
from transformers import AutoTokenizer,MBart50TokenizerFast

#Available models for En-De NMT
model_dict ={'marianMT_En_to_De': "Helsinki-NLP/opus-mt-en-de",
             'mBART50': 'facebook/mbart-large-50-one-to-many-mmt',
             't5': "t5-small",
             'marianMT_De_to_En': "Helsinki-NLP/opus-mt-de-en"}

def preprocessing(data,tokenizer,model_name,max_input_length,max_target_length,from_lang,to_lang):
    if model_name=='t5':
        if from_lang=='en' and to_lang=='de':
            prefix= "translate English to German:"
        elif from_lang=='de' and to_lang=='en':
            prefix= "translate German to English:"
        else:
            prefix=""
        input_text=[prefix + i[from_lang] for i in data['translation']]
        target_text=[i[to_lang] for i in data['translation']]
        input_tokenizer=tokenizer(input_text,max_length=max_input_length,truncation=True)
        with tokenizer.as_target_tokenizer():
            labels=tokenizer(target_text,max_length=max_target_length,truncation=True)
            input_tokenizer['labels']=labels['input_ids']
        return input_tokenizer
```

Figure 19 – Marian MT model

## Initialising Tokenizer for different models

```
#MarianMT model tokenizer for English to German translation
tokenizer_marianMT_En_De=AutoTokenizer.from_pretrained(model_dict['marianMT_En_to_De'],use_fast=False,)
#MarianMT model tokenizer for German to English translation
tokenizer_marianMT_De_En=AutoTokenizer.from_pretrained(model_dict['marianMT_De_to_En'],use_fast=False)
#T5 small model
tokenizer_t5=AutoTokenizer.from_pretrained(model_dict['t5'],use_fast=False)
#mBART50 model tokenizer for English to German translation
tokenizer_mbart_En_De= MBart50TokenizerFast.from_pretrained(model_dict['mBART50'],src_lang="en_xx",tgt_lang = "de_DE")
#mBART50 model tokenizer for German to English translation
#tokenizer_mbart_De_En= MBart50TokenizerFast.from_pretrained(model_dict['mBART50'],src_lang="de_De",tgt_lang = "en_XX")
```

Figure 20 – Auto tokenizer Model

## **4.2 Software Complication**

The operations of software part can involve many complications, which we as well faced. Some of them were

1. Version compatibility
2. Coding errors
3. Models' compatibility with each other
4. Transformers algorithm
5. Finding the perfect libraries for data extraction

### **4.2.1 Version Compatibility**

The versions of Python and all running operations should be kept into considerations, in order to run the program smooth and error-free. Few versions of python don't support MBart 50, and it need to run at previous models due to compatibility issues.

### **4.2.2 Coding Errors**

The coding errors involve the syntax and other functional errors. The changes in functional traits of Python have been fast and some functions don't coordinate with syntax of old versions, which makes us get few errors and days to resolve them.

### **4.2.3 Model Compatibility**

The models Marian M5 and MBart 50 aren't well coordinated by side to each other, which was a big issue at development end, which forced us to add additional functions and transformers for smooth running.

#### **4.2.4 Transformers Algorithm**

Transformers itself are complicated and need time to solve their errors which was a hard task itself.

#### **4.2.5 Finding the perfect libraries for data extraction**

Finding most common library for dataset training is not that easy, how it seems. The most common English library is still accessible but getting a same translated German library took the team into much more complications, which was later solved by hugging face where both data sets were pre-trained for us.

## Chapter 5

# HARDWARE METHODOLOGY AND IMPLEMENTATION

The hardware implementation portion of this project is divided into whole project is majorly divided into three parts:

1. Server Development
2. Remote Client Access
3. Raspberry Pi Integration
4. Translation Evaluation

The overall pipeline can be seen in the figure.

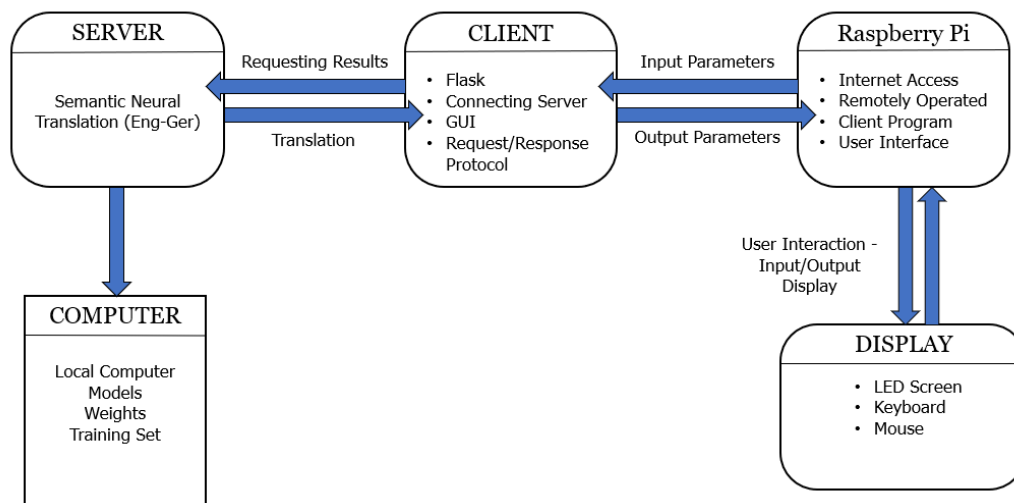


Figure 21 – Hardware Implementation pipeline block diagram

### 5.1 Server Development

In general terms, a server can be defined as a program, setup or device which is used as a resource or service for some other computer program such as a user also termed as a client. A server could be a physical computer, CPU or GPU on which the server programs operate and runs, this is referred as a server. We can have categories of servers such as a server which only contains the concerned data i.e., unique to a single task or organization etc. is a dedicated server or a non-dedicated server is one which is hosted in an environment which is shared and other separate tasks, organization, data is using it and is not unique to a single data or it might be used for other purposes. [11]



A client and server are a programming model, where a server generally operates all the time and doesn't stop and awaits for the request from the user i.e., client program and then fulfils that request. A client programs might be operating on the same device or any other computers or operating devices. In simpler terms, an application in one computer can be fulfilling the role of a client which would be requesting server on the same device or other device for the services to be obtained from other programs on the server. So, a client will be requesting for the service and the server will be waiting for the request from the client and providing the appropriate response accordingly.

The development of a server is comprising of a several parts in this project and generally overall. It is divided into three parts:

1. Developing an API with Flask and Python
2. Inference of the Software Model for Output Processing
3. Local Server Running, Request and Response fulfillment

### **5.1.1 Software and Tools Used**

The software and tools used in this part of the project are as follows:

1. Computer with decent RAM and GPU to be operated as a host server.
2. Python3
3. Anaconda Notebook
4. Jupyter Notebook
5. Flask API
6. Computer or Operating Device for the client program to be operated on (Raspberry Pi – in this project).
7. Tkinter for GUI development

## 5.1.2 Developing API with Flask and Python

### What is meant by an Application Program Interface (API)?

Any well-defined interface that identifies the service that one component, module, or application delivers to other software elements is known as an Application Program Interface (API). The API Server is a lightweight Web application that lets users construct and expose data APIs without having to do any custom programming.

In a nutshell, API stands for Application Programming Interface, and it is the means through which you interface with a service provider, which can be a server, a locally saved application, or even a physical device. In the context of the web, an API server is a server that performs particular activities in response to requests that are received by API.

An API server is where an API is stored. The server is usually a cloud-based computer that executes the programming that powers the API. As a result, when you state the request reaches the API, you're also implying that the client makes touch with the API server.

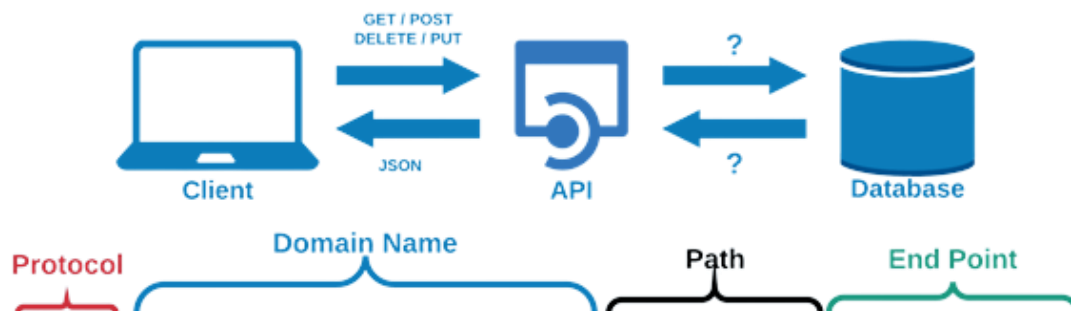


Figure 22 – Basic working diagram of an API

### What is Flask and how does it work?

Flask is a Python microframework based on Werkzeug and Jinja2 that is BSD licensed and sourced. Flask is a relatively basic yet very extendable framework, despite the fact that it is a microframework. This helps programmers the freedom to pick the configuration they desire, making it easier to write apps or plugins. Werkzeug and Jinja2 are the two main components of Flask. Flask uses Jinja2 as a template engine, while Werkzeug is in charge of routing, debugging, and the Web Server Gateway Interface (WSGI). Flask doesn't enable database access, user authentication, or any other high-level functionality out of the box, but it does support extension integration to add all of these features, making Flask a micro-yet-production-ready framework for building online apps and services.

A basic Flask application may be written in a single Python file or modularized into a production-ready solution. Flask's goal is to provide a solid base for all applications while relying on extensions for the rest. Flask, like all other Python libraries, can be downloaded from the Python Package Index (PPI) and is very simple to set up and use. Getting started with Flask takes only a few minutes. You should be familiar with Python, command line (or at least PIP), and MySQL in order to follow the workings of this project and how these resources are used, as mentioned and summarized in the technical background of this report.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, From Flask!'

if __name__ == '__main__':
    app.run()
```

Figure 23 – Starting a basic Flask Application [12]

The above code imports the Flask library, creates an instance of the Flask class to start the application, specifies the route, and then defines the function that will be run when the route is called. This code is sufficient to get your first Flask application up and running. The following code starts a very basic built-in server, which is good enough for testing but probably not for production, but we'll get to it in subsequent chapters. The index route should return "Hello, From Flask!" when this application begins, as seen in Figure below.

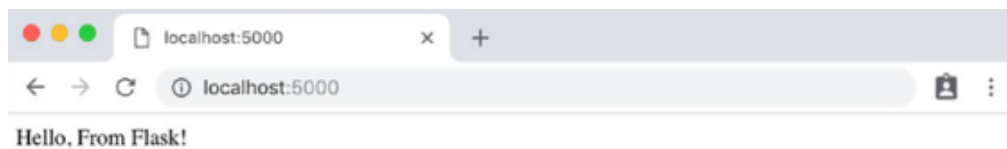


Figure 24 – A minimalistic flask application response [12]

## How is the Flask API developed and Used in this project?

```
from flask import Flask, request, Response
```

The above code imports the Flask library from python3 environment in the local computer which was setup. Along with request and response commands for the sharing and communication link between server and client.

```
app = Flask(__name__)

@app.route('/test', methods=['POST', 'GET'])

def test():
    model_dict = {'marianMT_En_to_De': "Helsinki-NLP/opus-mt-en-de",
                 'mBART50': 'facebook/mbart-large-50-one-to-many-mmt',
                 't5': "t5-small",
                 'marianMT_De_to_En': "Helsinki-NLP/opus-mt-de-en"}
    r = request
    text = r.json.get('text')
    i_lang = r.json.get('i_lang')
    o_lang = r.json.get('o_lang')
    model = r.json.get('model')
```

The flask is initialized and a route is established and the function is defined and coded which will be required while performing inference on the server program. The *request* protocol is being used here to accept any incoming requests from client and then processing the relevant function that is needed. The *json.get* command is actually receiving the input parameters which will be defined by the user in the remote client device and program. These parameters will be then used in the processing and inference code in the server which will provide the desired output and will the required response of the client.

### 5.1.3 Inferencing the Software Model for Processing Output

Training and inference are the two fundamental steps of machine learning and deep learning. A developer gives their model selected datasets during the training phase so that it may learn what it needs to know about the type of data it will analyze. The model may then generate predictions based on live data in the inference phase, resulting in actionable outcomes.

#### What is Machine Learning Inference?

Inference in machine learning (ML) or Deep Learning (DL) is the act of feeding live data into a machine learning algorithm or model to get a single numerical score. "Operating an ML model" or "bringing an ML model into production" are other terms for this procedure. When a machine learning (ML) model is used in production, it is sometimes referred to as artificial intelligence (AI) since it performs functions akin to human reasoning and analysis. Because the ML model is often merely software code that performs a mathematical technique, machine learning inference essentially includes putting a software application into a production environment. That algorithm performs computations based on the data qualities, which are referred to as "features" in machine learning jargon. [13]

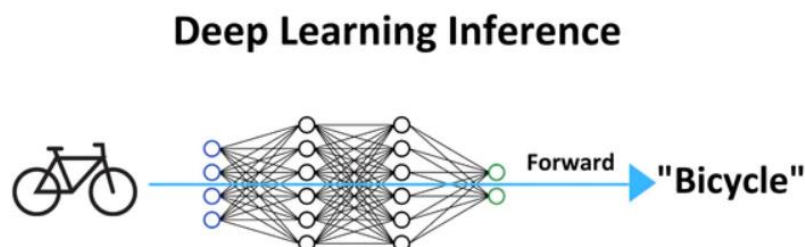


Figure 25 – Deep Learning Inference Example

A machine learning lifecycle may be divided into two separate sections. The first phase is the training phase, during which an ML model is formed or trained by feeding it a subset of data. The second phase is machine learning inference, which involves putting the model to work on real-time data to generate useful results. The ML model's data processing is sometimes referred to as scoring, therefore one might say that the ML model scores the data, with a score as the result.

DevOps engineers or data engineers are the most common users of machine learning inference. The data scientists who are in charge of training the models are sometimes expected to take

charge of the ML inference process. Because data scientists aren't always experienced at implementing systems, this latter condition frequently creates substantial roadblocks in getting to the ML inference stage. Successful machine learning deployments are frequently the product of close collaboration across several teams, and emerging software solutions are frequently used to try to make the process easier. MLOps, an emerging profession, is putting more structure and resources into bringing machine learning models into production and sustaining them when modifications are required.

### **How is Machine Learning Inference used and work?**

A data source, a machine learning system to analyze the data, and a data destination are all required for machine learning inference. It's possible that a data source is actually a collection of data from other sources. When data is collected from a variety of IoT sources, this is the situation. A machine learning model sorts and analyses live data as it comes in using a series of mathematical procedures. These algorithms may draw conclusions based on data attributes, or characteristics that distinguish one thing from another. The model then assigns a score to these data points and generates a result depending on how it was taught to interpret the data. Those output ratings can aid in determining the next step. For example, let's say you're collecting data from a customer service chatbot on an e-commerce site. Your data source would be the customer input from the online chat platform. Then a machine learning model could determine that, if a customer writes that they are not satisfied with a product, the customer may want a refund. A separate system could then help issue a refund for that customer. Let's imagine you're gathering information from an e-commerce site's customer support chatbot. The consumer feedback from the online chat platform would be your data source. Then, if a consumer writes that they are unhappy with a product, a machine learning model may predict that the customer may demand a refund. A different mechanism might then assist in issuing a refund to the consumer.

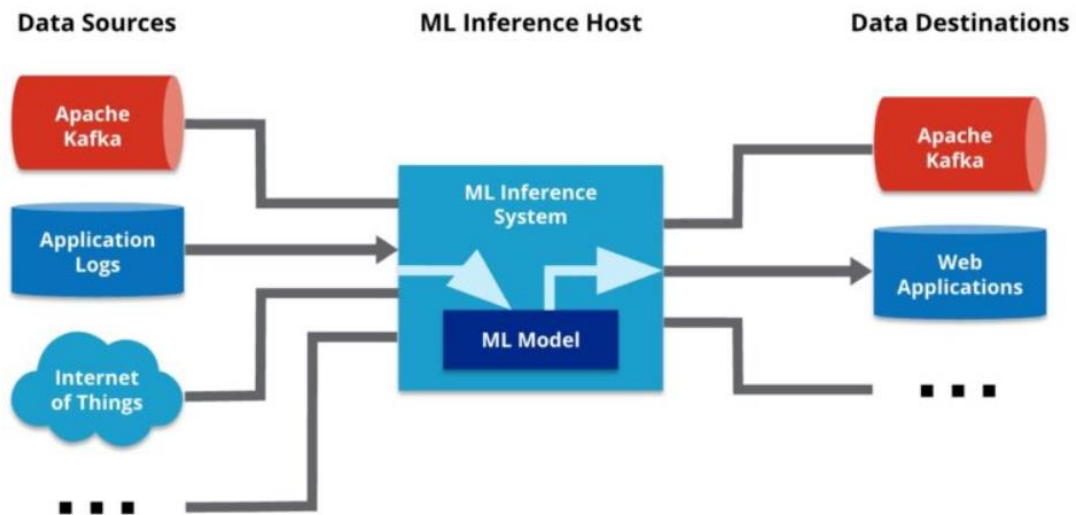


Figure 26 – A machine learning inference system diagram

The data sources in machine learning inference are usually a system that gathers live data from the process that creates the data. The machine learning model's host system takes data from the data sources and feeds it into the machine learning model. The data destinations are the locations to which the host system should send the machine learning model's output score. The data sources are usually a system that gathers real-time data from the process that creates it. An Apache Kafka cluster, for example, may store data produced by an Internet of Things (IoT) device, a web application log file, or a Point of Sale (POS) machine. A data source might be as simple as a web application that captures user clicks and transmits the information to the ML model's hosting system.

The ML model's host system receives data from the data sources and feeds it into the model. The infrastructure required to transform the code in the ML model into a fully functional application is provided by the host system. The host system delivers the output of the ML model to the data destinations when it has been created. A web application that receives data input via a REST interface, or a stream processing application that takes an incoming feed of data from Apache Kafka and processes numerous data points per second, are examples of host systems. The data destinations are the locations to which the host system should send the ML model's output score. Any sort of data repository, such as Apache Kafka or a database, can be used as a destination, and downstream applications can then act on the scores. If the ML model develops a fraud score on purchase data, for example, the data destinations apps may send an accept or reject message back to the buy site.

## What Is Deep Learning Training and How Does It Work?

A deep neural network (DNN) learns how to assess a specified collection of data and make predictions about what it means through deep learning training. It takes a lot of trial and error before the network can make correct inferences based on the intended outcome. DNNs are frequently referred to as a sort of artificial intelligence since they employ artificial neurons to approximate human intellect. A DNN is capable of sorting through abstract data kinds like photos and audio recordings.

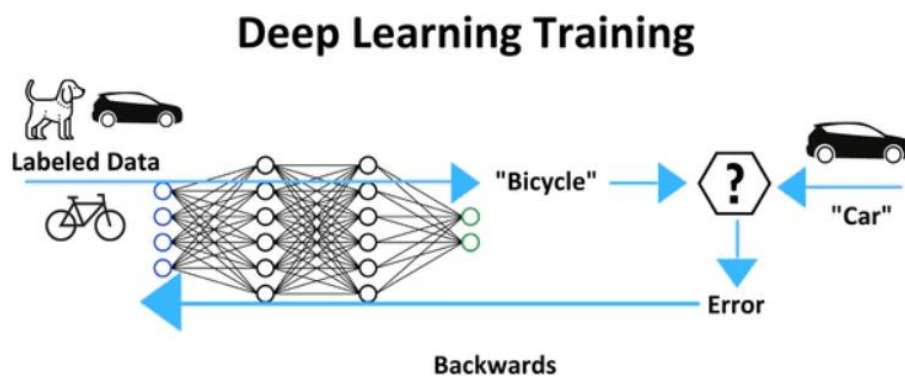


Figure 27 – Deep Learning Training Example

A DNN sifts through prior data during deep learning training and derives inferences based on what it thinks the data represents. When the system reaches an inaccurate conclusion, the result is sent back to the system so that it can learn from its error. This process strengthens the connections between the artificial neurons over time, improving the system's ability to make correct predictions in the future. The DNN should be able to categorize and evaluate new and maybe more complicated data as it is provided with it. In the end, it will learn from its experiences and grow more perceptive over time.

## What's the Difference Between Deep Learning Inference and Training?

Without training, deep learning inference is impossible. The distinction between deep learning training and machine learning inference is that the former serves as a foundation for the later. Consider how humans learn to accomplish a task by studying how it's done, whether by reading a handbook, watching how-to videos, analyzing labeled datasets, or seeing someone else do it. In the same way that DNNs pick up on patterns and draw inferences from existing data, we learn to imitate others.



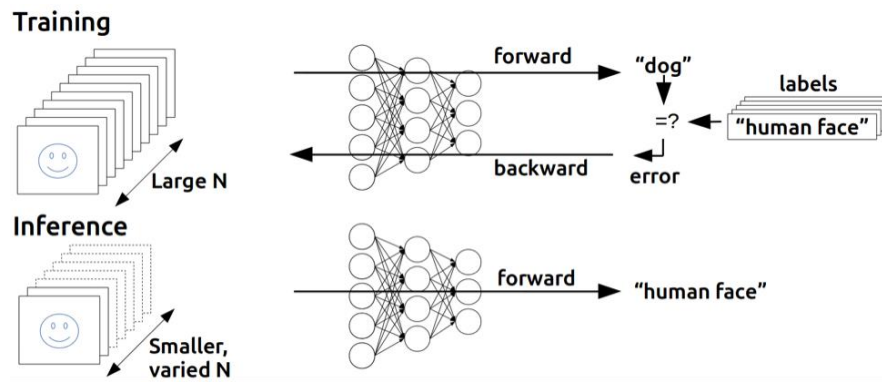


Figure 28 – Deep Learning Training vs Inference Example

In their own ways, both machine learning training and inference are computationally expensive. In terms of training, feeding a DNN huge volumes of data is time-consuming for GPU computing, and it may need the use of more or better efficiency units. In addition, mitigating latency difficulties throughout the inference process might make it difficult for the system to make real-time choices. Machine learning initiatives may often be difficult to coordinate across several departments within a corporation.

For example, if a machine learning project is written in TensorFlow but a member of the team is better experienced with PyTorch, there may be difficulties testing and deploying the system. And, especially if the models need to be trained on fresh data, upgrading them may be costly and laborious.

Furthermore, AI hardware suppliers provide varying levels of software support for frameworks and layers, making it difficult to compare multiple hardware solutions in order to reach the project's Total Cost of Ownership (TCO).

The cost for developing an external hardware in place of a computer or an operating system machine is significant for having to setup inference of such machine learning and deep learning models. Therefore, development of a Graphical user Interface (GUI) on our raspberry pi hardware because it was not compatible for inference to be done on it. Therefore, creating a client program and operating an interface on the hardware such that the hardware operates as a remote device which will be providing translation results with the help of server parameters obtained after processing the inference model on the server.

## How is this project Inferencing the Translation Model on the server?

```
import transformers
from transformers import MBart50TokenizerFast, MBartForConditionalGeneration, MarianTokenizer, MarianMTModel
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
```

Here we are importing the relevant libraries which were used to operate the training models and the corresponding weights in the software translation model as used and described the software implementation chapter of the report. Importing transformers libraries and tokenizers packages to analyze the translation models and provide the results.

```
if (i_lang=='English') and (o_lang=='German'):

    if model=='t5':
        model_name = 't5-small-finetuned-en-to-de-large-ds\\content\\t5-small-finetuned-en-to-de-large-ds'
        tokenizer = AutoTokenizer.from_pretrained('t5-small')
        model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
        translated = model.generate(**tokenizer(text, return_tensors="pt", padding=True),max_length=1024)
        trans_text=[tokenizer.decode(t, skip_special_tokens=True) for t in translated]

    elif model=='marianMT_En_to_De':
        model_name = 'opus-mt-en-de-finetuned-en-to-de-large-ds\\content\\opus-mt-en-de-finetuned-en-to-de-large-ds'
        tokenizer = MarianTokenizer.from_pretrained(model_name)
        model = MarianMTModel.from_pretrained(model_name)
        translated = model.generate(**tokenizer(text, return_tensors="pt", padding=True))
        trans_text=[tokenizer.decode(t, skip_special_tokens=True) for t in translated]

    elif model=='mBART50':
        model_name=model_dict['mBART50']
        tokenizer = MBart50TokenizerFast.from_pretrained(model_name,src_lang="en_XX")
        model = MBartForConditionalGeneration.from_pretrained(model_dict['mBART50'])
        model_inputs = tokenizer(text, return_tensors="pt")
        generated_tokens = model.generate(**model_inputs,forced_bos_token_id=tokenizer.lang_code_to_id["de_DE"])
        trans_text = tokenizer.batch_decode(generated_tokens, skip_special_tokens=True)

elif (i_lang=='German' or 'german') and (o_lang=='English' or 'english'):
    if model=='t5':
        model_name = 't5-small-finetuned-de-to-en-large-ds\\content\\t5-small-finetuned-de-to-en-large-ds'
        tokenizer = AutoTokenizer.from_pretrained('t5-small')
        model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
        translated = model.generate(**tokenizer(text, return_tensors="pt", padding=True),max_length=1024)
        trans_text=[tokenizer.decode(t, skip_special_tokens=True) for t in translated]

    elif model=='marianMT_De_to_En':
        model_name = 'opus-mt-de-en-finetuned-de-to-en-large-ds\\content\\opus-mt-de-en-finetuned-de-to-en-large-ds'
        tokenizer = MarianTokenizer.from_pretrained(model_name)
        model = MarianMTModel.from_pretrained(model_name)
        translated = model.generate(**tokenizer(text, return_tensors="pt", padding=True))
        trans_text=[tokenizer.decode(t, skip_special_tokens=True) for t in translated]

else:
    trans_text = 'Unable to translate'

output = {'output': trans_text}
return output
```

In this python code, the program is analyzing the input parameters received from the user client program with the help of the API connection developed between this server code with the client code on a remote device and hardware. Input parameters are given by the user from the client program which are taken in this server program and used for processing the results.

### 5.1.4 Local Server Running and Response

HTTP and all HTTP-based extension protocols have a fairly basic communications mechanism. The way it works is that a client, usually a web browser, submits a request for a resource to a server, and the server responds with either the resource or an error message if it can't handle the request for any reason. A resource can be anything from a basic HTML file sent to the browser verbatim to a software that dynamically constructs the response.

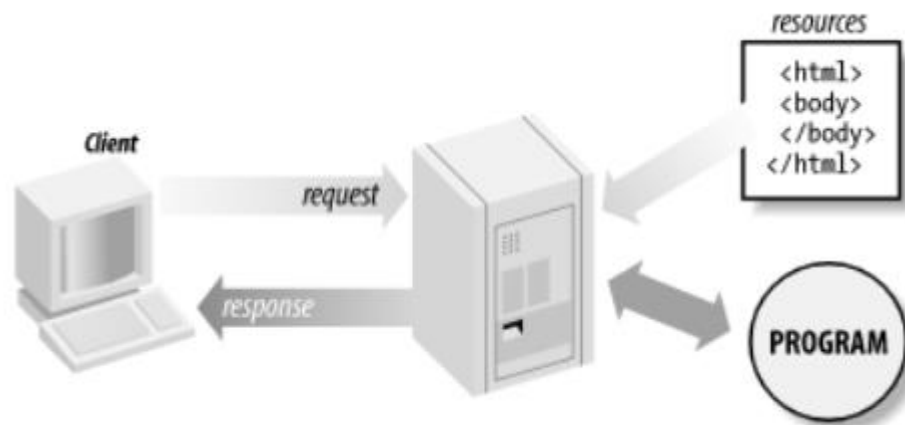


Figure 29 – HTTP request/response

This straightforward model indicates key points to consider:

1. HTTP is a protocol that has no state. This implies that after sending a response, the server doesn't maintain any information about the client, and hence can't tell if successive requests from the same client are linked.
2. Web apps are unable to deliver the same level of instant response as standalone GUI programs like word processors or classic client/server applications. A request/response exchange is required for every interaction between the client and the server. When a user picks an item from a list box or fills out a form element, a request/response exchange is generally too expensive on the bandwidth available to most Internet users.

HTTP, which stands for Hypertext Transfer Protocol, is a request/response protocol that is used by web clients and servers to interact. The GET and POST methods of HTTP are used to retrieve and manipulate data.

GET: Gets information from the server. A URI is used to specify the request's target also known as a resource (Uniform Resource Identifier). When used in this context, this is generally but not always an absolute reference to a file on the server and is referred to as a URL.

POST: Returns data to the server. A POST request provides a payload in addition to the URL

and query parameters. The payload is generally form data, which is the sum of the contents of the response's multiple fields (also known as controls).

Anyone who has used a web browser is familiar with the general shape of a URL:

```
http:// host [:port] / [absolute_path [ ? query_parameters ] ]
```

Such that:

http://

Indicates that the request is being made via the Hypertext Transfer Protocol. This is referred to as the scheme in a URI. Two schemes are supported by BIS: http and https are interchangeable terms (secure http).

Host:

The name of the machine that will receive the request, as well as its location. Generally, this would be indicating the IP Address of the computer or device.

Port:

An optional number that indicates the server port on which the request will be received. This defaults to 80 for the http scheme and 443 for the https scheme if not specified. A unique web server is identified by the combination of host and port (along with host headers, which is a method that allows a single host to serve numerous domains).

Absolute Path:

The absolute path to the resource on the host that is being requested. This is the name of a file most of the time (but not always). The base directory is the root directory of the web tree that is being served by the host on the given port, not the root directory of the file system.

Query Parameters:

Optional parameters that the web server and service software have access to.

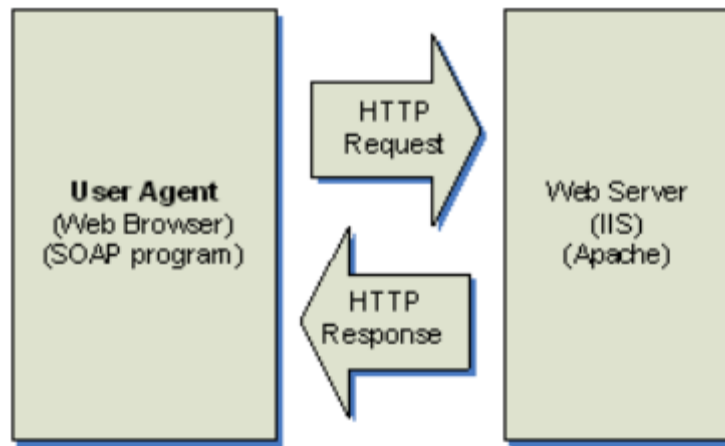


Figure 30 – Request/Response between User and Web Server

```
app.run(host='0.0.0.0', port=5000, ssl_context="adhoc")
```

```
* Running on all addresses (0.0.0.0)  
WARNING: This is a development server. Do not use it in a production deployment.  
* Running on https://127.0.0.1:5000  
* Running on https://192.168.52.40:5000 (Press CTRL+C to quit)
```

## 5.2 Remote Client

As discussed above that a client program interacts with the user and establishes connection with the server via the HTTPS request/response protocol. Client and server programs must create a communication session over the network or networks that connect them in order to communicate. The client can then call remote processes in the server programs as if they were local to the client programs once, they've established the connection.

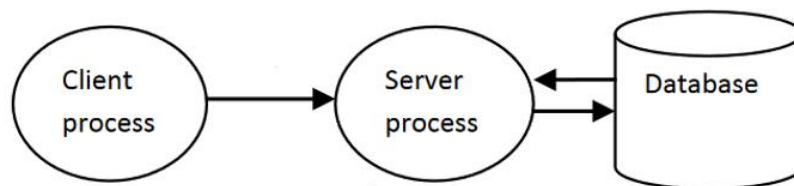


Figure 31 – A Client-Server System [14]

The development of the client program in this project is divided into two steps:

1. Establishing the Connection with Server
2. Graphical User Interface for User Interaction

### 5.2.1 Establishing Connection with Server

The client establishes the connection in the client/server model. The server simply sits there, waiting for any number of clients to send requests. Similarly, to how a waitress at a restaurant is available to service any clients who enter at any time, regardless of reservations. Customers arrive at their leisure, asynchronously. The server assigns a port to listen for client connections in the client server architecture, and then waits for the clients to connect. They can do so asynchronously and at their leisure. The server should be able to manage numerous connections at the same time, exactly like a restaurant server. The programmer determines how the server process achieves. However, it frequently includes several threads, or child processes, operating in parallel.

The communication begins when a client establishes a connection with a server. The client makes requests of the server, which the server responds to. The client then disconnects. Clients may connect to several servers at the same time, and a server may manage numerous connections at the same time. A lot of internet apps function in this way. Web browsers are web server clients, email programmes like Outlook are mail server clients, and so on. Client and server processes are frequently run-on separate computers on the Internet, connected simply by the network, but this does not have to be the case; the server might operate on the same system as the client.

Client programmes using explicit handles must generate a binding handle before starting a client/server communication session with a server programme. The run-time library then locates the server program's host machine. It then locates the endpoint being monitored by the server application and routes the call there. This procedure is shown in the diagram below:

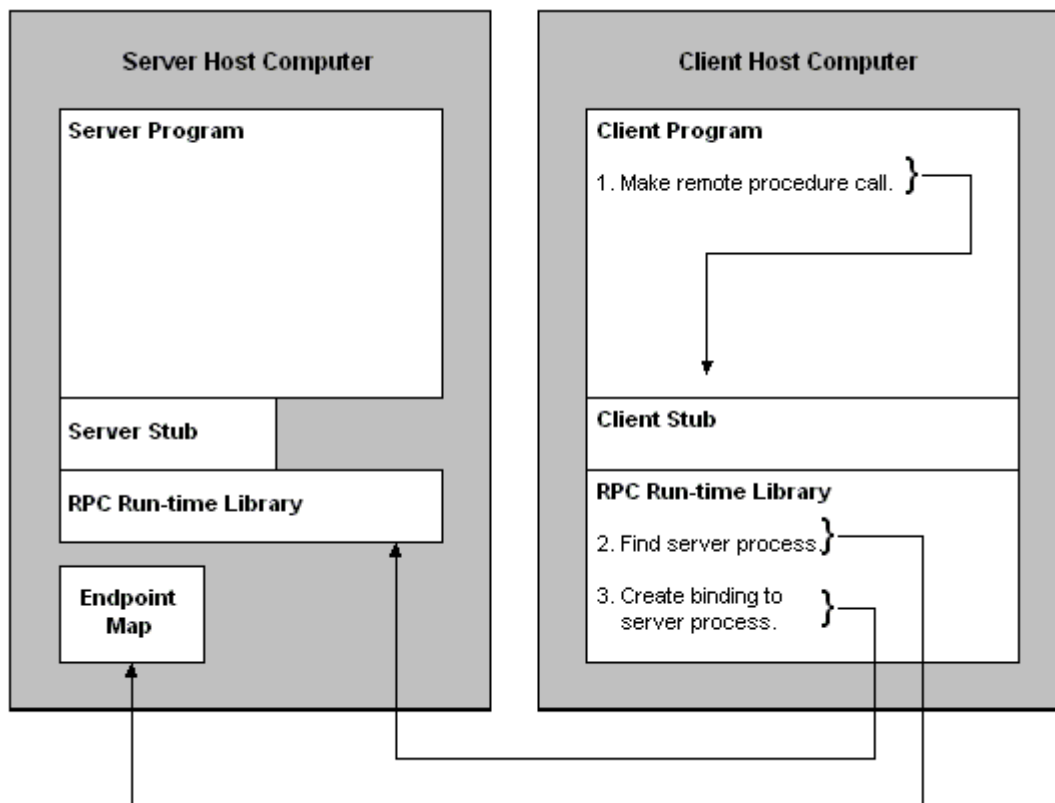


Figure 32 – Run-time process of a Server-Client Connection

This section explains how the client communicates with the server application and uses the remote procedures it provides. There are several ways to complete these processes; depending on the design selected, an application may select a different set of steps. This is only one example of how to go about it.

The client-server paradigm, also known as client-server architecture, is a distributed application framework that divides responsibilities between servers and clients, who can either be in the same system or interact across a network or the Internet. In order to access a service provided by a server, the client must make a request to another software. One or more programmes run on the server that share resources and distribute tasks across clients.

The client-server interaction uses a request–response message pattern and must follow a common communications protocol that officially defines the rules, terminology, and conversation patterns that must be followed. The TCP/IP protocol suite is commonly used for client-server communication.



The Client-server computing is divided into four categories:

1. **One-Tier Architecture:** It comprises of a basic application that runs on a single machine and does not require network connection. Because user requests do not control any network protocols, the code is simple and the network is free of the unnecessary traffic.
2. **Two-Tier Architecture:** The client, the server, and the protocol that connects the two tiers make up a two-tier architecture. The domain logic is on the server host, while the graphical user interface code is on the client host. High-level languages like Python, C++ and Java are used to create the client-server GUI.
3. **Three-Tier Architecture:** The presentation tier, which is the User Interface layer, the application tier, which is the service layer that does detailed processing, and the data tier, which is the database server that stores information, make up a three-tier architecture.
4. **N-Tier Architecture:** This tier design separates an application into logical levels that segregate duties and handle dependencies, as well as physical tiers that operate on distinct computers, enhance scalability, and add delay due to increased network traffic. Closed-layer architecture, in which a layer can only connect with the layer below it, or open-layer architecture, in which a layer can communicate with any layer below it. [15]

The client-server architectural concept has a number of advantages:

1. A single server that hosts all of the essential data in one location simplifies data protection and user authorization and authentication administration.
2. A client-server network may be expanded by adding network segments, servers, and computers without causing substantial downtime.
3. Data may be accessed effectively without requiring clients and the server to be in close proximity.
4. The client-server system's nodes are all self-contained, requesting data solely from the server, making upgrades, replacements, and relocation of nodes simple.

Data exchanged over client-server protocols is platform-independent.

```
import requests  
addr = 'https://host:5000'  
test_url = addr + '/test'
```

```
response = requests.post(test_url, json = {'text': text, 'i_lang': i_lang, 'o_lang': o_lang, 'model': model})
```

In the code the requests were made and the address is included. The request provides the input parameters and obtains the output response after server processing.

## 5.2.2 Graphical User Interface Data for User Interaction

A graphical user interface (GUI) is a user interface that uses icons, menus, and other visual indications or representations of images to allow users to interact with electronic devices such as computers and smartphones. Unlike text-based interfaces, where data and commands are purely in text, GUIs graphically show information and related user controls. A pointing device, such as a mouse, trackball, pen, or a finger on a touch screen, is used to control GUI representations. A GUI (graphical user interface) is a set of visual components that interact with computer software. A graphical user interface (GUI) shows items that communicate information and indicate actions that the user may do. When the user interacts with the items, they change color, size, and visibility. [16]

### **Using Tkinter for developing the project's User Interface:**

In Python, there are several ways to create graphical user interfaces (GUIs). Here are the most critical.

1. Tkinter is the Python interface to the Tk GUI toolkit that comes with Python. In this chapter, we'll examine this possibility.
2. It's an open-source Python interface to the Windows framework.
3. Java class libraries are readily available to Python scripts via the Python programming language, which is a Java translation of the Python programming language.

Installing dependencies on the laptop is complete. For Raspberry Pi apps, Tkinter is used to construct user interfaces (UIs). It is ideal because of its small size and portability on the Raspberry Pi. Python's standard GUI package, Tkinter, is a must-have for anybody using the language. Python with Tkinter may be used to construct GUI apps quickly and easily. It's a strong object-oriented interface to the Tk GUI toolkit that is provided by Tkinter. Tkinter is an attractive choice for a Python GUI framework since it's included in the Python standard library and making applications using it is pretty straightforward.

```

import tkinter
from tkinter import *
from tkinter import ttk

root = Tk()
root.geometry('1080x400')
root.resizable(0,0)
root.config(bg = 'black')
root.title("English-German Language Translator")
Label(root, text = "LANGUAGE TRANSLATOR", font = "arial 20 bold", bg='green').pack()
Label(root, text = "Neural Machine Translation", font = 'arial 15 bold', bg = 'green', width = '30').pack(side = 'bottom')
Label(root, text = "Enter Text to be translated", font = 'arial 13 bold', bg = 'green').place(x=200,y=60)

Input_text = Text(root, font = 'arial 10', height = 11, wrap = WORD, padx=5, pady=5, width = 60)
Input_text.place(x=30,y = 100,)

Label(root, text = "Output", font = 'arial 13 bold', bg = 'green').place(x=780,y=60)

Output_text = Text(root, font = 'arial 10', height = 11, wrap = WORD, padx=5, pady= 5, width =60)
Output_text.place(x = 600 , y = 100)

src_lang = ttk.Combobox(root, values=['English','German'], width =12)
src_lang.place(x=20,y=60)
src_lang.set('choose input language')
dest_lang = ttk.Combobox(root, values= ['English','German'], width =12)
dest_lang.place(x=890,y=60)
dest_lang.set('choose output language')
model_list = ttk.Combobox(root, values=[*model_dict], width =20)
model_list.place(x=500,y=60)
model_list.set('choose model')

trans_btn = Button(root, text = 'Translate', font = 'arial 12 bold', pady = 5, command = translate_func , bg = 'green')

trans_btn.place(x = 490 , y= 180 )
def Close():
    root.destroy()

root.mainloop()

```

The user interface of this project has been created with Tkinter; the client program includes the interface which has been developed in a python environment with Tkinter. The figure below is the interface that has been developed in this project.

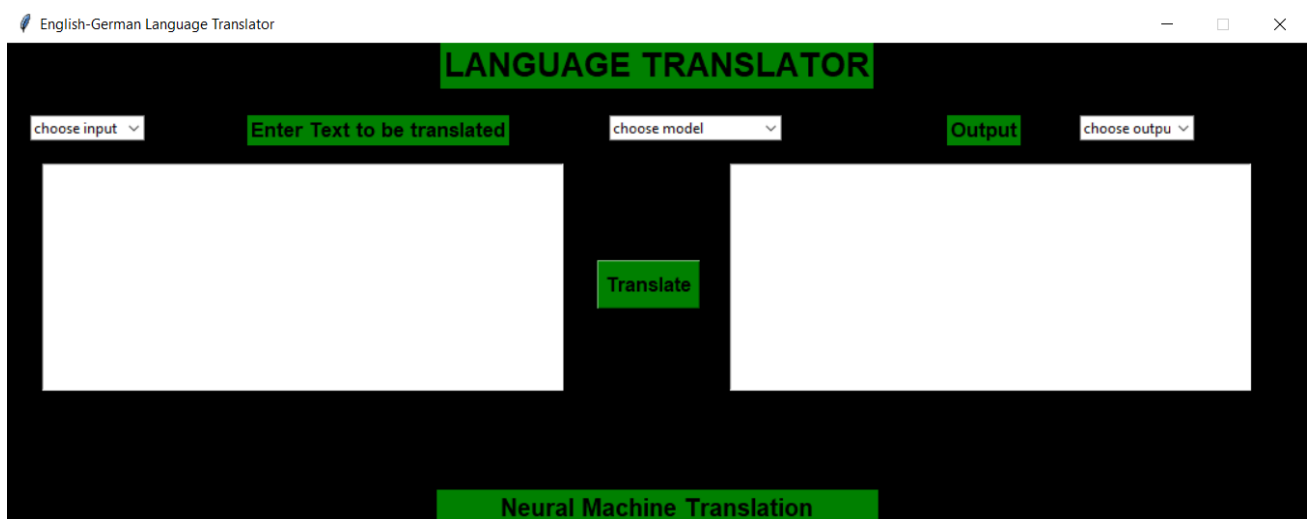


Figure 33 – Tkinter based User Interface of the project.

### 5.3 Raspberry Pi Integration

Raspberry Pi is a single-board computer with a compact footprint. It is possible to attach peripherals such as a keyboard, mouse, and monitor to the Raspberry Pi. The Raspberry Pi is a low-cost computer that runs Linux and has a set of General-Purpose Input/Output (GPIO) pins for controlling electronic components and exploring the Internet of Things (IoT). It will operate as a little personal computer. Raspberry Pi is more than a computer since it allows developers to access on-chip hardware, such as GPIOs, to create applications. By using GPIO, we may connect and control devices such as LEDs, motors, and sensors.



Figure 34 – Raspberry Pi Board

The Raspberry Pi is available in a variety of variants, the more prominent versions used are mentioned below:

1. Raspberry Pi 1 Model B+ (2014)
2. Raspberry Pi 1 Model A+ (2014)
3. Raspberry Pi 2 Model B (2015)
4. Raspberry Pi Zero (2015)
5. Raspberry Pi 3 Model B (2016)
6. Raspberry Pi Zero W (2017)
7. Raspberry Pi 3 Model B+ (2018)
8. Raspberry Pi 4 (2021)

### 5.3.1 Use and Specifications of the Raspberry Pi

The Raspberry Pi 3 B+'s on-chip hardware is used in this project and depicted in the diagram below:

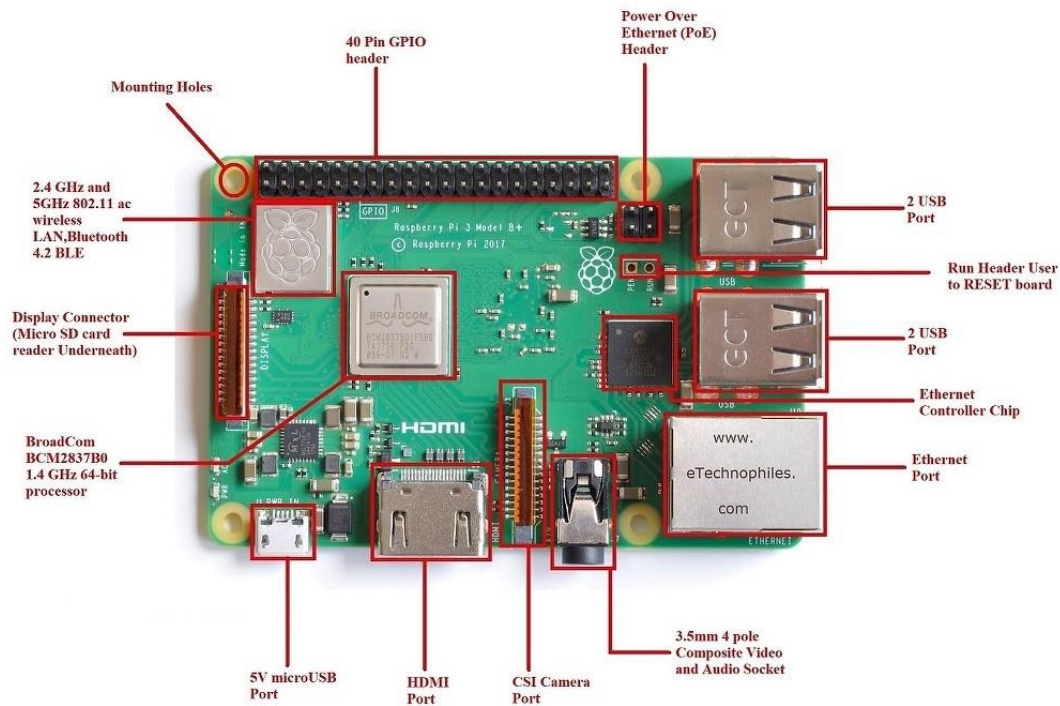


Figure 35 – Raspberry Pi Model 3 B+ Hardware

Some of the above-mentioned hardware components are listed below:

1. **HDMI (High-Definition Multimedia Interface):** This interface is used to send uncompressed video or digital audio data to a computer monitor, digital television, or other device. This HDMI connector is used to connect the Raspberry Pi to a digital television in general.
2. **CSI Camera Interface:** The CSI (Camera Serial Interface) interface connects the Broadcom Processor to the Raspberry Pi camera. This interface is used to link two devices electrically.
3. **DSI Display Interface:** The DSI (Display Serial Interface) Display Interface is used to connect a 15-pin ribbon cable to the Raspberry Pi. DSI is a high-resolution display interface that allows video data to be sent directly from the GPU to the LCD display.
4. **Composite Video and Audio Output:** This port sends video and audio signals to audio/video systems.

5. Power LED: A RED-colored LED that indicates power. When you add power to the Raspberry Pi, this LED will illuminate. It is directly linked to 5V and will begin to flash if the supply voltage falls below 4.63V.
6. ACT PWR: ACT PWR is a green LED that displays the activity of the SD card.


<b>Raspberry Pi Features</b>	<b>Value</b>
<b>Model</b>	<b>Raspberry Pi 3 Model B+</b>
<b>SOC</b>	<b>Broadcom BCM2837B0</b>
<b>CPU</b>	<b>Quad-Core A53 (ARMv8)</b>
<b>Operating Frequency</b>	<b>1.4 GHz</b>
<b>RAM</b>	<b>1GB LPDDR2 SDRAM</b>
<b>GPU</b>	<b>Broadcom Videocore-IV</b>
<b>ETHERNET</b>	<b>Gigabit Ethernet via USB channel, 2.4GHz and 5GHz Wi-Fi</b>
<b>GPIO</b>	<b>40-pin GPIO header</b>
<b>HDMI</b>	<b>1x HDMI Port</b>
<b>USB Port</b>	<b>4x USB 2.0</b>
<b>Display</b>	

Figure 36 – Raspberry Pi 3 Model B+ Specifications

### 5.3.2 Setting Up an Environment for User Interface on Hardware

#### Steps Involved:

Now the below steps need to be performed.

**Step 1:** To begin, we'll use some tools to transfer the Raspberry operating system to the microSD card. Afterward, we'll go on to the next stage of setup. We're going to go with the desktop version of the Raspberry Pi as it has complete support. Alternatively, you might use NOOBS and raspberry Lite.

We'll insert it into the microSD card and run the ether program when both downloads are complete. Once the ISO file and microSD card were selected, we proceeded to the next step: The Ether UI is extremely user-friendly and straightforward. Once you've clicked Flash, the Raspberry operating system will be installed automatically.

Insert the microSD card into the Raspberry Pi when the flash has finished. Using a USB connection, power up the raspberry pi and attach it to a monitor via HDMI or VGA.

**Step 2:** As a result, the default login credentials for Raspbian OS will be pi (username) and raspberry (password). Use these credentials to access the desktop.

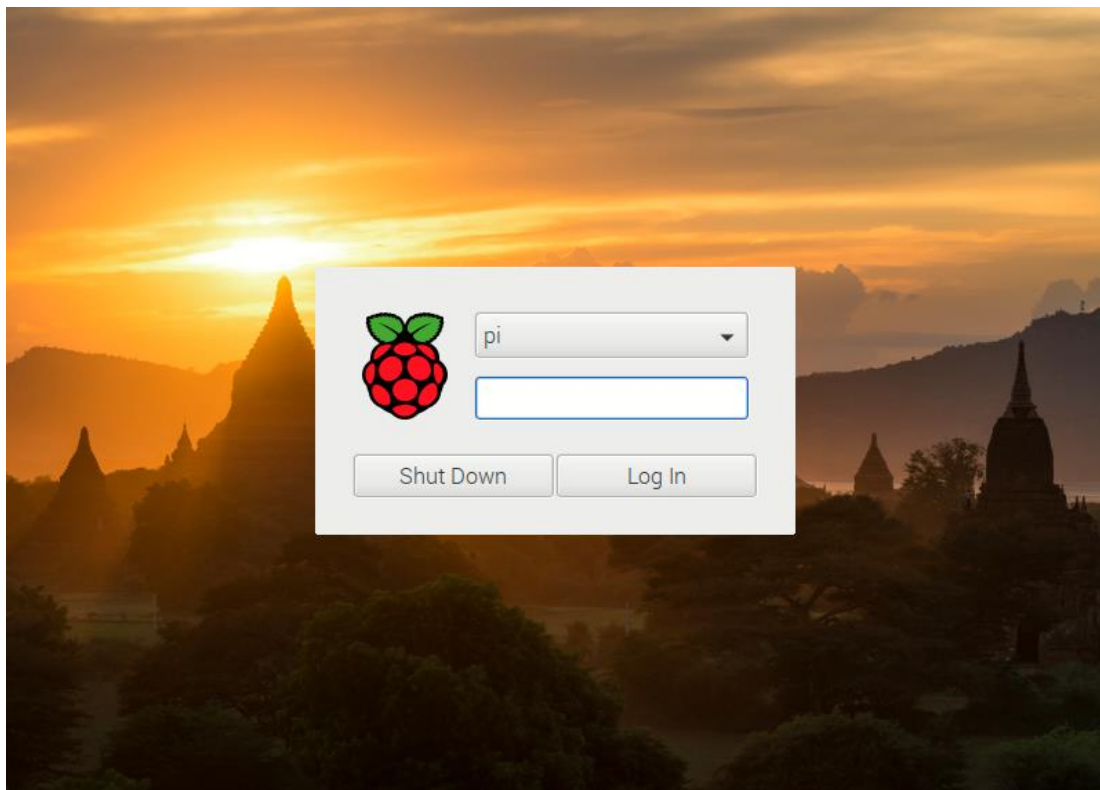


Figure 37 – Raspberry Desktop

Linux systems come with Python installed by default, but they are usually not the latest. Python also cannot be updated by a typical apt upgrade command as well. To check the version of Python installed on your system open a terminal and check to see if a newer version of Python has been installed on the Raspberry Pi. Run the command:

```
python3 --version
```



If python needs an upgrade, run the following commands:

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

```
sudo apt update
```

**Step 3:** Run the python client script on the raspberry pi. The Interface will start up and the user can access and use the translation device in real time. Refer to figure of the user Interface made for this project as described and showed earlier.

### 5.3.3 Starting User Interface on Bootup

By making the required changes in the setting of my raspberry pi, you can follow the guide provided here to do this yourself. The client script will AutoStart on boot up and the user will be able to use the translator even if they don't have any knowledge about how to start a python script on raspberry pi.

## **5.4 Displaying and Evaluation of Translation Results**

The raspberry pi is connected with few other components so that the user can operate with the help and the translation results can be achieved.

### **5.4.1 Equipment/Hardware Required**

In order for the user to use the translation device i.e., the Raspberry Pi, there are a few hardware items or equipment that are needed for this purpose.

1. Power Charger/Adapter
2. Raspberry Pi Casing
3. HDMI Cable
4. HDMI to VGA Adapter
5. LED Display
6. Keyboard
7. Mouse
8. Raspberry Pi Display

### **5.4.2 User Input Parameters**

This project is somewhat unique since it allows the user to choose the input parameters required for the processing of the translation in Realtime. The parameters are the input text in English language or German Language and the models based with whom the translation will be done.

For example, if the user enters the input parameter, what would it mean in terms of the interaction with the interface and the back-end code entries.

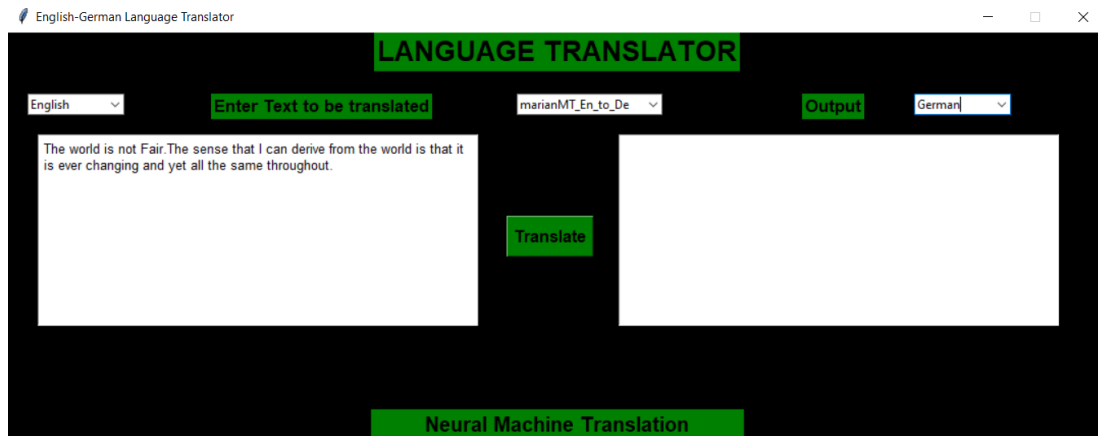


Figure 38 – User entering Input Parameters in the Interface.

What this means in the back-end code of the client program. The syntax will be something like this.

```
input_text = 'The world is not Fair. The sense that I can derive from the world is that it is ever changing
              and yet all the same throughout.'
input_lang = 'English'
output_lang = 'German'
model_name = 'marianMT_En_to_De'

response = requests.post(test_url, json = {'text': input_text, 'i_lang': input_lang, 'o_lang': output_lang, 'model': model_name})
```

This is the function in the code behind the display basically which does the job of collecting input parameters from the interface interaction of the user. Then requesting the server with the parameters taken from the user and come back with the response of the translated results from the server.

```
def translate_func():
    if (src_lang.get()=='English') and (dest_lang.get()=='German'):
        if model_list.get()=='marianMT_En_to_De':
            text = Input_text.get(1.0, END)
            i_lang = src_lang.get()
            o_lang = dest_lang.get()
            model = 'marianMT_En_to_De'
            response = requests.post(test_url, json = {'text': text, 'i_lang': i_lang, 'o_lang': o_lang, 'model': model})
            Output_text.delete(1.0, END)
            Output_text.insert(END, response.json()['output'] )
```

### 5.4.3 Realtime Translation Response

After the request is processed from the client program on the raspberry pi operating as a remote device. This will require an internet connection which can be established via the Gigabit Ethernet port or the Wi-Fi option which is available.

The response will be obtained from the server if the connection is established. The output obtained will be displayed on the user interface in the output section.

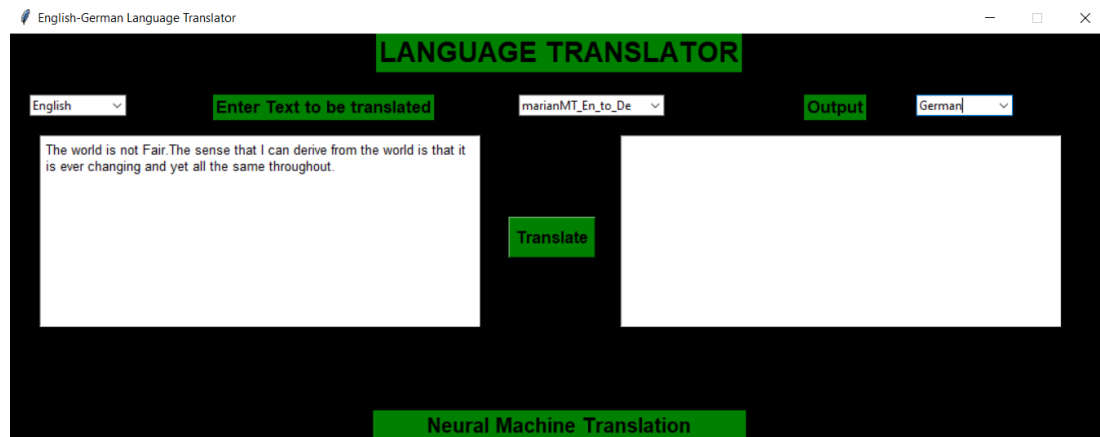


Figure 39 – Translation result on the Interface obtained after the response.

# Chapter 6

## RESULTS

The results of the translation software models are shared with different input text examples obtained from the results of the models developed in the translation software.

### 6.1 Marian MT Model Results

There are two Marian MT Models, one for English to German translation and the other for German to English Translation.

#### 6.1.1 English to German Results

The translation results have been obtained here. The translation is analyzed contextually and retain the sense of the sentence as much as possible.

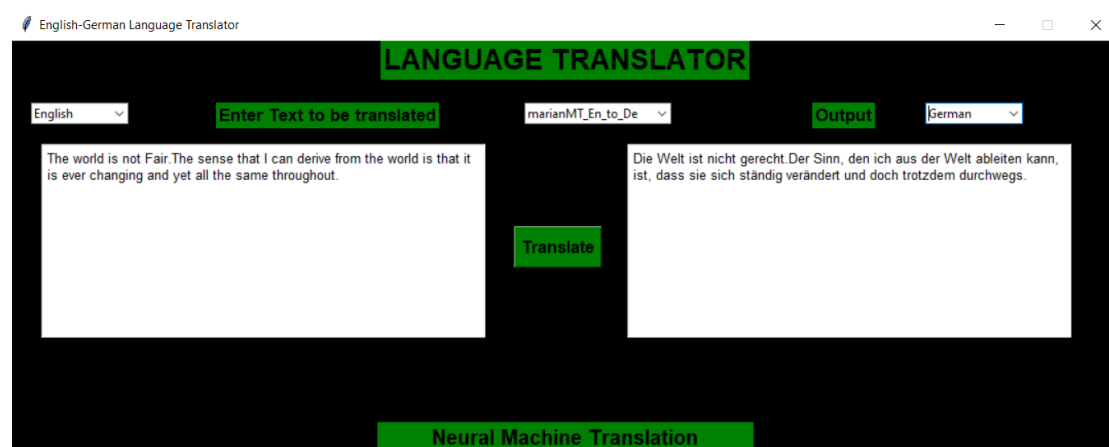


Figure 40 – Marian MT: English to German translation result

#### 6.1.2 German to English Results

The results of the German sentence which was obtained from the output obtained earlier is taken as input and the English translation is obtained. This shows and depicts that translation makes sense contextually and is solving the problem observed in general translation models in use.

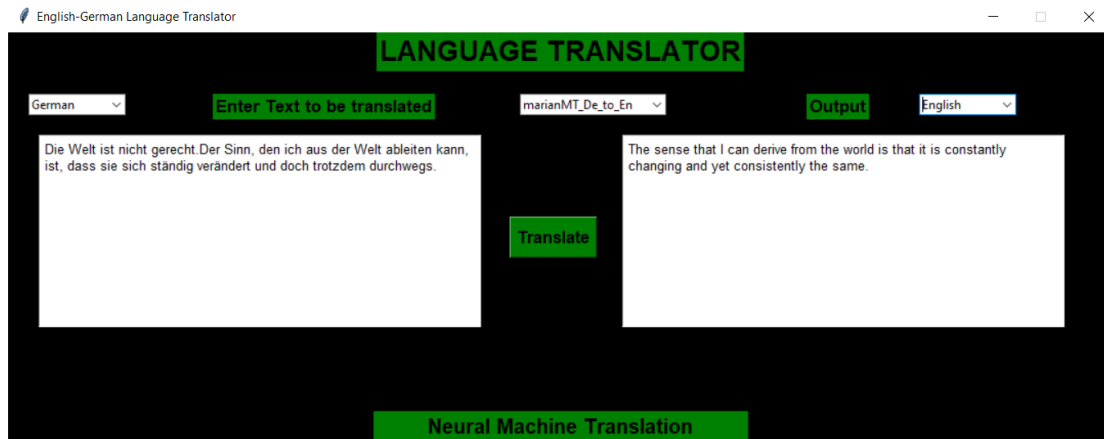


Figure 41 – Marian MT: English to German translation result

## 6.2 T5 Model Results

The T5 Model, can provide both English to German translation and German to English Translation in a single instead of two as compared to the Marian MT models.

### 6.2.1 English to German Results

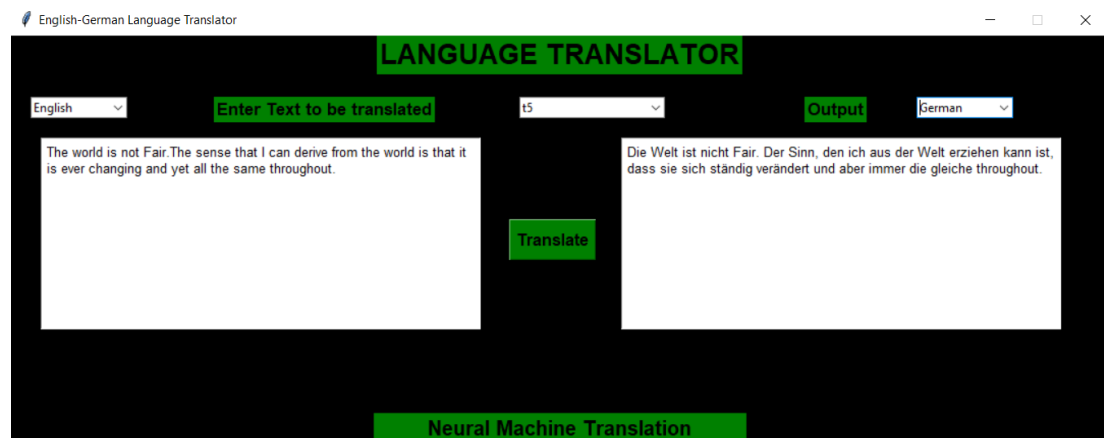


Figure 42 – T-5 Model: English to German translation result

## 6.2.2 German to English Results

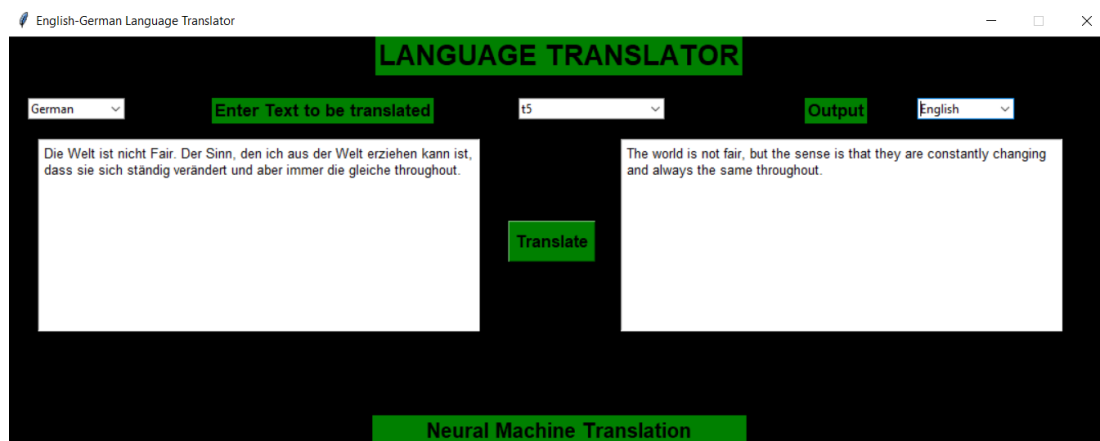


Figure 43 – T-5 Model: English to German translation result

## 6.3 MBART-50 Model Results

The M-Bart 50 Model, computes the English to German translation. The results are shared below:

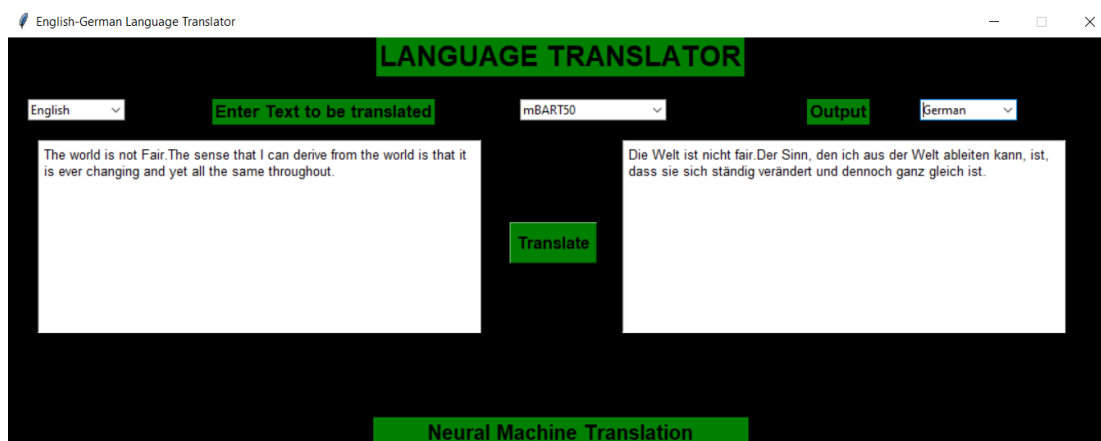


Figure 44 – MBART-50 Model: English to German translation result

## 6.4 Observations

The observation derived from this shows that the translation model is keeping the context of the message between the translation results.

1. Less Syntax Error
2. Generally, more Accurate and Precise results
3. Fast Real-time translation
4. More than 500 words of translation with good precision maintaining the context of the source message.

## Chapter 7

# CONCLUSION AND FUTURE WORK

Here are a few recommendations and conclusions for anyone who wants to build upon our work on this project.

### 7.1 Future Works and Recommendations

#### 1. Local Dataset Development

By working on labelled datasets for local languages, translation algorithm can be developed for regional languages as well.

#### 2. Including more Models

Diverse Model, each with own ingenuity and training environment with respect to the desired Languages, more effective translation can be obtained. Including more models in the software algorithm, this will provide more room for improvement in translation results.

#### 3. Multilingual Support

Multilingual support will allow us to serve users in more than one language. There are around 7,100 languages spoken in the world today.

#### 4. Sentiment Analysis

Sentiment analysis (or opinion mining) is an NLP technique used to determine whether data is positive, negative or neutral. Work is being done in this domain internationally and nationally as well.

#### 5. Comprehension Software

Derive and understand valuable insights from text within documents such as to uncover valuable insights and connections in text. For example, Amazon Comprehend, Google Cloud for NLP etc.



## **6. Voice & Text based AI Assistants**

Virtual assistants using natural language processing (NLP) are here to stay as they become more efficient and cost saving.

Language Processing is a broad domain, it isn't just restricted to translation or sentiment analysis. Use these methodologies to work on auto responsive bots and software which will be helpful in automated communication and customer assistance. For example, Amazon has used Language Processing techniques to develop their AI based customer support engine which interacts with users and customers and helps them with their queries and problems, which has been instrumental in Amazon customer support which is provided by AI and NLP based machines.

## 7.2 Conclusion

The conclusion derived from the results of this project shows that the translation model is solving the problem of keeping the context of the message between the translation results.

### 1. Syntax Error Reduction

The translation that our model provides reduced syntax errors comparatively. There is always huge optimization potential, with more time, data, training and computational power, it will provide more efficient and error free translation and other potential opportunities.

### 2. Portable Hardware

A light weight, easy to carry device. Which can be enabled anytime, via internet access. Language translation made easier and reliable.

### 3. Real Time Translation

Real Time translation results are obtained. Efficient and accessible.

### 4. Better Accurate and Precision

The translation between the languages obtained, is quite reliable because of its better accuracy and precision compared to other word-to-word and Statistical Networks based models.

### 5. Web-based Applications Compatible

This software is also ready to be deployed on web-based consumption. With integration with the required tools such as app development, websites etc.

### 6. Language Barrier - not anymore

The work on AI based language processing is reducing language barrier among common people rapidly, if not already.

## REFERENCES

- [1] M. Johnson, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation," *arXiv:1611.04558v2*, 2017.
- [2] Y. Wu, "Google's neural machine translation system: Bridging the gap between human and machine.," *arXiv preprint arXiv:1609.08144*, vol. , 2016.
- [3] A. Vaswani, "Transformer: A Novel Neural Network Architecture for Language Understanding," *Google AI Blog*, 2017.
- [4] I. Sydorenko, "What Is a Dataset in Machine Learning: Sources, Features, Analysis," *Label Your Data*, p. 14, 2021.
- [5] Priya, "Difference between Small Data and Big Data," *EDUCBA*, p. 12, 2021.
- [6] I. Cloud, "Natural Language Processing (NLP)," *IBM Cloud*, p. 30, 2020.
- [7] Q. Chen, "T5: a detailed explanation," *Medium*, p. 20, 2020.
- [8] Z. Keita, "Data Augmentation in NLP Using Back Translation With MarianMT," *Medium*, p. 17, 2020.
- [9] M. D. Gangi, "mBART50: Multilingual Fine-Tuning of Extensible Multilingual Pretraining," *Medium*, p. 5, 2021.
- [10] H. Face, "Tokenizer," *Hugging Face*, p. 20, 2019.
- [11] Broan Posey (TechTarget), ""What is a Server?" (Online), Available: <https://www.techtarget.com/whatis/definition/server>".
- [12] K. Relan, *Building REST APIs with Flask: Create Python Web Services*, 2019.
- [13] Hazelcast, ""What Is Machine Learning Inference?" (Online), Available: <https://hazelcast.com/glossary/machine-learning-inference/>".
- [14] H. S. Oluwatosin, "Client-Server Model," *IOSR Journal of Computer Engineering (IOSR-JCE)*, pp. 1-2, 2014.
- [15] C. Kambalyal, "3-tier architecture," *Retrieved On*, 2, 2010.
- [16] J. Stoltzfus, "Graphical User Interface (GUI)," *Techopedia*, 2021.

- [17] Z. Halim, "Machine learning model deployment as API service using Python Web," *e-Proceedings of Information Technology and Computing Sciences Colloquium Series 'Digital Technology Empowerment Anticipates Expertise'*, 2021.
- [18] Y. Xia, "Dual Inference for Machine Learning," in *Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.
- [19] David Amos, ""Python GUI Programming With Tkinter", (Online) (2022), Available: <https://realpython.com/python-gui-tkinter/>".
- [20] M. O. Laura Sach, *Create Graphical User Interfaces with Python: How to Build Windows, Buttons, and Widgets for Your Python Projects*, Raspberry Pi Press, 2020.
- [21] Z. H, "Architecture of Network and Client-Server model.," *arXiv preprint arXiv:1307.6665.*, 2013.