

Student Undertaking of Internship/ OJT Academic Details (only applicable when no Internship/ OJT pathway is present in the program scheme)

Name: Hyndavi Gundapuneni Reg. No.: 12018017

Program Code and Name: CSE441–Industry Internship Project Section No.: K20UP

Name of Company: UPGRAD CAMPUS Start Date of Internship/ OJT: 18 Jan

Stipend during Internship/ OJT: NA Package: NA

Academic Requirement during Internship/ OJT (To be filled in consultation with Academic HOD and AOC)

Autumn Term (Term id): _____ **No. of course to be studied:** _____

Details of courses to be studied:

No. of courses to be waived off:

Details of courses to be waived off:

Requirement of CA:

CA is to be prorated as per the provisions of proration policy: _____

Term paper will be assigned in lieu of CA: _____

Any Other: _____

Spring Term (Term id): _____ **No. of course to be studied:** _____

Details of courses to be studied:

No. of courses to be waived off:

Details of courses to be waived off:

Requirement of CA:

CA is to be prorated as per the provisions of proration policy: _____

Term paper will be assigned in lieu of CA: _____

Any Other: _____

Name of Academic HOD:

Name of AOC:

UID of Academic HOD:

UID of AOC:


Signature of Academic HOD:

Signature of AOC:

Undertaking by Student:

1. I have been informed and I am aware about the academic requirements that I need to fulfill along with OJT/Full term Internship/Full year internship.
1. I understand that I have to fulfill my professional responsibilities in organization and academic requirements like ETE/ETP, Field project, CA etc. simultaneously without seeking any favour from the university.
2. I will manage my leaves in my organization and will appear for ETE/ETPs as per the Examination schedule of University.
3. I understand that if I will not able to appear for exam (due to any reason) then I will appear for reappear/Backlog as per the provisions and schedule of University.

Date: 09/05/2024

Signature of Student: 

LANGUAGE TRANSLATOR

Project report submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Hyndavi Gundapuneni (12018017)

Supervisor

Shamneesh Sharma



School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

May 2024

@ Copyright LOVELY PROFESSIONAL UNIVERSITY, Punjab (INDIA)

May 2024

ALL RIGHTS RESERVED

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B.Tech project report entitled “**Language Translator**”, submitted by **Hyndavi Gundapuneni** at **Lovely Professional University, Phagwara, India** is a bonafide record of his original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

Shamneesh Sharma

Date: 09/05/2024

DECLARATION STATEMENT

I hereby declare that the work reported in the project report entitled "LANGUAGE TRANSLATOR" in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Shamneesh Sharma. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this project report represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my project report.



Hyndavi Gundapuneni

12018017

ACKNOWLEDGEMENT

I have given sincere efforts in this project. However, it would not have been possible without the knowledge and teachings provided by my mentor. I would like to extend my sincere thanks to Mr. Shamneesh Sharma. I would like to express my gratitude towards my parents & mentors of LPU for their kind encouragement which helped me in completion of this project.



Hyndavi Gundapuneni (12018017)

Dated: 09/05/2024

TABLE OF CONTENTS

CONTENTS	PAGE NO.
Inner first page – Same as cover	i
Supervisor’s Certificate	ii
Declaration by the Scholar	iii
Acknowledgement	iv
Table of Contents	v
List of Figures	vi
CHAPTER 1: INTRODUCTION	11
1.1 Executive Summary	11
1.2 Objectives	12
1.3 Introduction	13
CHAPTER 2: THEORETICAL BACKGROUND	14
2.1 Transformers	14
2.2 Neural Machine Translation (NMT)	15
2.3 Statistical Machine Translation (SMT)	18
2.4 Google Cloud Translation	21
2.5 Live Translate	22
2.6 Automatic Speech Recognition (ASR)	24
2.7 Optical Character Recognition (OCR)	25
2.8 Website Translation	26
2.9 Document Translation	27

CHAPTER 3: METHODOLOGY	29
3.1 Data Collection and Preparation	30
3.2 Text Language Translation	35
3.3 Audio-to-Text Translation	40
3.4 Optical Character Recognition (OCR) Translation	48
3.5 Integration and User Interface Development	54
CHAPTER 4: CONCLUSION	56
REFERENCES	57

LIST OF FIGURES

FIGURE NO.	FIGURE DESCRIPTION	PAGE NO.
Figure1	Translator Visualisation	11
Figure2	Our models idea visualisation	13
Figure3	Transformers	14
Figure4	Google Translate basic functioning visualisation	15
Figure5	NMT table example	16
Figure6	One hot encoding visualisation	17
Figure7	Encoder to vector to decoder visualisation	17
Figure8	SMT matching visualisation	19
Figure9	Union of words from input and target languages	20
Figure10	Google Cloud Translate working visualisation	21
Figure11	Text-to-text visualisation	23
Figure12	Automatic Speech Recognition (ASR) working	24
Figure13	Optical Character Recognition (OCR) working	25
Figure14	Document translation idea visualisation	27
Figure15	Flowchart of methodology	29
Figure16	Dataset overview	31
Figure17	Languages supported	31
Figure18	Token lengths histogram of input texts	32
Figure19	Token lengths histogram of output texts	33
Figure20	Visualization of training and validation losses	34
Figure21	Examples of wrong predictions	35
Figure22	Pseudo code for Translation using Transformer model (1)	37
Figure23	Pseudo code for Translation using Transformer model (2)	37
Figure24	Pseudo code for Language Translation using APIs	39
Figure25	Text-to-text translation output using Google Translate API	39
Figure26	Text Language Translation application deployed	40
Figure27	Flowchart of ASR working	42
Figure28	Pseudo code for Live Translate model	44
Figure29	Live Translate application deployed	45
Figure30	Pseudo code for Audio Translation model (1)	46

Figure31	Pseudo code for Audio Translation model (2)	47
Figure32	Working of Audio Translation model	48
Figure33	Pseudo code for Live Capture Translation model (1)	49
Figure34	Pseudo code for Live Capture Translation model (2)	50
Figure35	Live Capture Language Translation model working	50
Figure36	Working of Image Document to Text Translation model	51
Figure37	Pseudo code for Image file to Text Translation model (1)	52
Figure38	Pseudo code for Image file to Text Translation model (2)	53
Figure39	Working of document language translation	53
Figure40	Pages division in document translation	54
Figure41	Model after deployment (1)	54
Figure42	Model after deployment (2)	55
Figure43	Model after deployment (3)	55

CHAPTER 1

INTRODUCTION

1.1 Executive Summary

Recent years have seen significant developments in language translation technology, which has made it possible to communicate with people all over the world in a seamless manner. In this report, a comprehensive investigation into the impact that a variety of cutting-edge technologies, such as Transformers, the Google Translate API, Automatic Speech Recognition (ASR), Optical Character Recognition (OCR), Website Translation, Document Translation, Audio Translation, and Voice Assistance, have had on language translation systems is carried out. Based on a comprehensive assessment of the relevant literature, this report provides an explanation of various technologies that have the potential to improve the capacities of translators. In addition, the approaches that can be utilized to incorporate these technologies into the translation systems that are already in place are outlined. By investigating the ways in which these technologies complement one another, the purpose of this study is to offer insights into the changing landscape of language translation and the implications that this landscape has for communication across cultural boundaries.

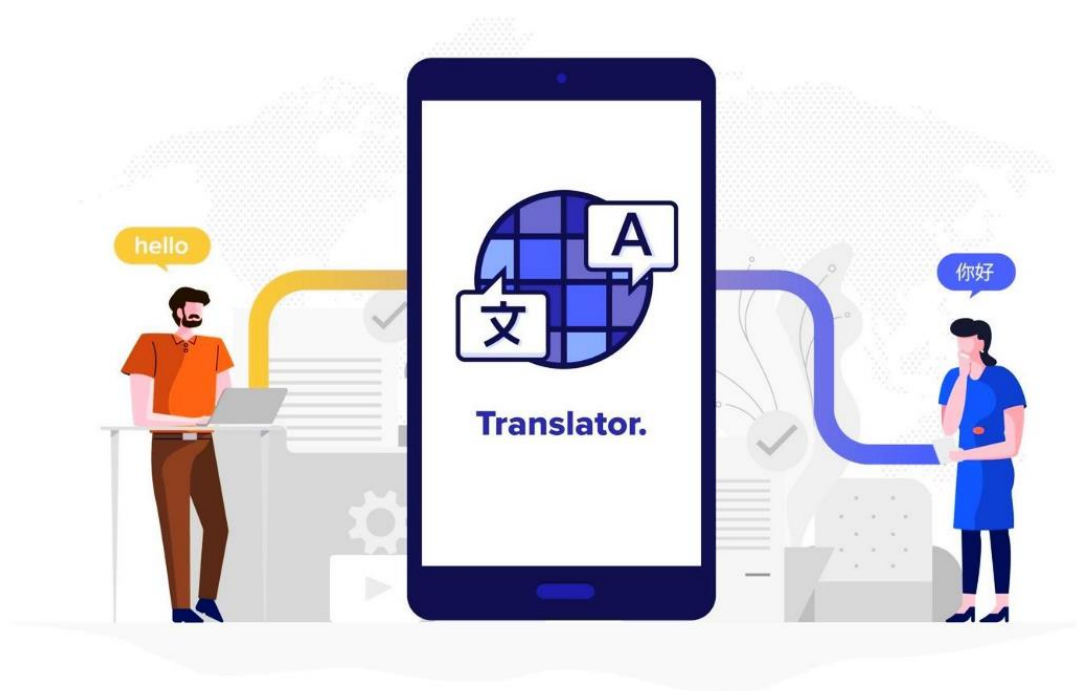


Fig1. Translator Visualisation

1.2 Objective

The primary objective of this report is to explain the making of language translator and its working. To find out the mistakes in the model and the reason for those mistakes. To find out the accuracy of the model and the deployment of the model. To integrate different technologies like Transformers, the Google Translate API, Automatic Speech Recognition (ASR), Optical Character Recognition (OCR), Website Translation, Document Translation, Audio Translation, and Voice Assistance to create a language translator that meets most of the world requirements and has higher accuracy and efficiency compared to present translators. By achieving these objectives, the report aims to provide a valuable resource for Language Translators seeking to optimize their processes, and improve business.

1.3 Introduction

In the current world, the barrier between different countries has been lowered and people have started to take interest in other countries' culture and literature. But the language barrier is a major problem as very few people are proficient in many languages most of the people do not know more than one or two languages. Imagine yourself only, if you want to watch a drama, web series or anime, how would you feel if it's not available in your language or any language that you understand. If you want to read a novel but it's not available in any language you can understand than you will definitely search for translated versions or even translate it by yourself. To solve this problem language translators were invented but most of the language translators either have very few features or have low accuracy and efficiency.

In recent times, there's been remarkable advancements in various technologies, particularly in language translation. These advancements have profoundly reshaped how we communicate globally, breaking down language barriers and fostering better understanding between cultures. This report dives deep into the transformative impact of cutting-edge technologies like Transformers, Google Translate API, ASR, OCR, Website Translation, Document Translation, Audio Translation, and Voice Assistance on language translation systems. Through an in-depth exploration, we uncover how these innovations have evolved, how they function, and what they mean for cross-cultural communication.

Transformers, for instance, utilize sophisticated attention mechanisms to capture intricate linguistic nuances and context, resulting in more accurate and contextually relevant translations. Similarly, the Google Translate API, armed with extensive datasets and neural machine translation algorithms, continuously refines its translation quality. ASR systems transcribe spoken language into text, enabling seamless translation, while OCR technology extracts text from images for translation. Website Translation tools adapt web content for global audiences, while Document Translation systems streamline the translation of documents. Audio Translation facilitates real-time spoken language translation, and Voice Assistance technology enables on-demand translation through virtual assistants.

By integrating these technologies, language translation systems can achieve higher accuracy, efficiency, and scalability. However, successful integration requires careful consideration of

compatibility, interoperability, and resource allocation. Identifying and rectifying model inaccuracies is crucial, as is assessing model accuracy across different languages and domains. Deployment poses challenges like resource constraints and scalability issues, which we explore in-depth, along with various deployment strategies.

The future of language translation holds promise, fuelled by advancements in AI, ML, and NLP. Emerging technologies like neural machine translation and contextual embeddings are set to enhance translation quality further. With an abundance of multilingual datasets and parallel text resources, translation models will become more robust and domain-adapted.

In summary, this report offers a holistic view of how cutting-edge technologies are reshaping language translation. By understanding the interplay between these technologies, we aim to empower language translators and individuals to communicate effectively across linguistic and cultural barriers, fostering a more connected global community.



Fig2. Our models idea visualisation

CHAPTER 2

THEORETICAL BACKGROUND

2.1 Transformers:

Transformers, in the context of machine learning and natural language processing (NLP), are a type of deep learning model that has revolutionized various tasks such as language translation, text generation, and sentiment analysis. They were introduced in a seminal paper titled "Attention is All You Need" in 2017. [1]

At the core of transformer models lies the "attention mechanism," which allows the model to focus on different parts of the input sequence when generating an output sequence. This mechanism enables transformers to capture long-range dependencies in data more effectively compared to previous sequence-to-sequence models like recurrent neural networks (RNNs) and long short-term memory networks (LSTMs).

The architecture of a transformer typically consists of an encoder and a decoder. The encoder processes the input sequence, such as a sentence in one language, and converts it into a series of contextualized representations. The decoder then takes these representations and generates an output sequence, such as a translation in another language. Each encoder and decoder layer in a transformer model contains multiple self-attention mechanisms and feedforward neural networks.

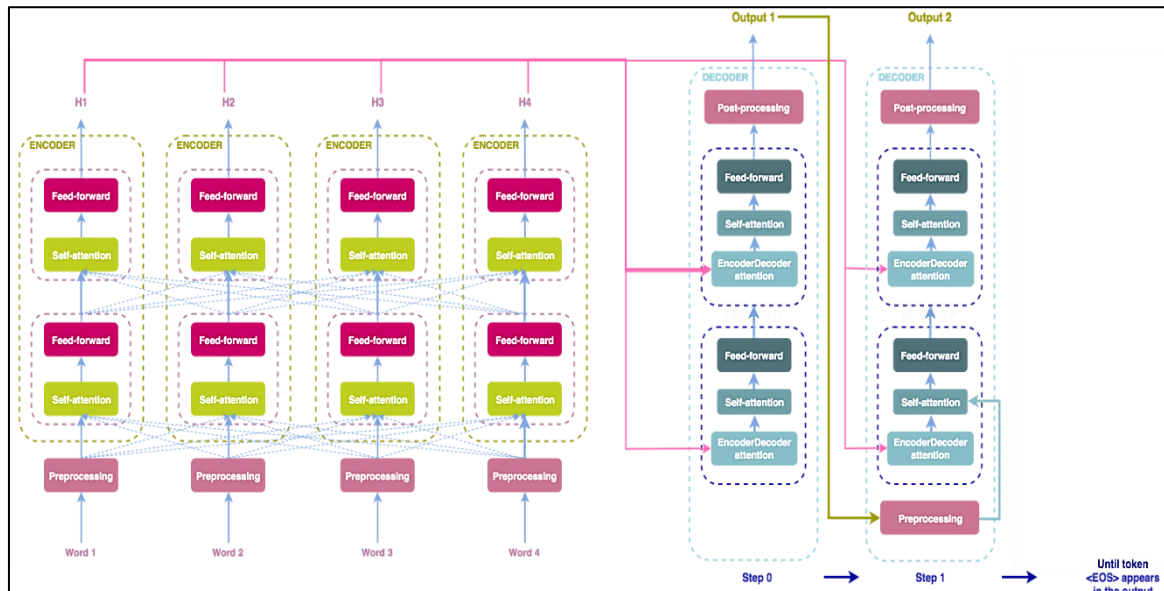


Fig3. Transformers

Advantages of Transformers:

- **Better Performance:** Transformers have shown superior performance in various natural language processing tasks compared to previous models like recurrent neural networks (RNNs) and convolutional neural networks (CNNs). They achieve this through attention mechanisms that enable capturing long-range dependencies effectively.
- **Parallelization:** Transformers allow for more efficient parallelization during training, making them faster to train compared to sequential models like RNNs. This results in reduced training time and improved scalability.
- **Flexibility:** Transformers can handle tasks beyond translation, including text generation, summarization, question-answering, and sentiment analysis, making them versatile for a wide range of natural language processing applications.
- **Contextual Understanding:** Transformers capture contextual information more effectively, enabling them to produce translations or predictions that are more contextually relevant and accurate.

Disadvantages of Transformers:

- **Computational Resources:** Transformers typically require significant computational resources, especially for large models with millions or billions of parameters. This can make them impractical for deployment on resource-constrained devices or in real-time applications.
- **Data Requirements:** Transformers, like other deep learning models, require large amounts of annotated data for training. Obtaining high-quality, diverse training data for certain languages or domains can be challenging and costly.
- **Fine-Tuning Complexity:** Fine-tuning transformers for specific tasks or domains may require expertise in machine learning and natural language processing, as well as substantial computational resources for experimentation and optimization.

2.2 Neural Machine Translation (NMT):

Machine Translation (MT) is a subfield of computational linguistics that is focused on translating text from one language to another.[2] With the power of deep learning, Neural Machine Translation (NMT) has arisen as the most powerful algorithm to perform this task. This state-of-the-art algorithm is an application of deep learning in which massive datasets of translated sentences are used to train a model capable of translating between any two languages.

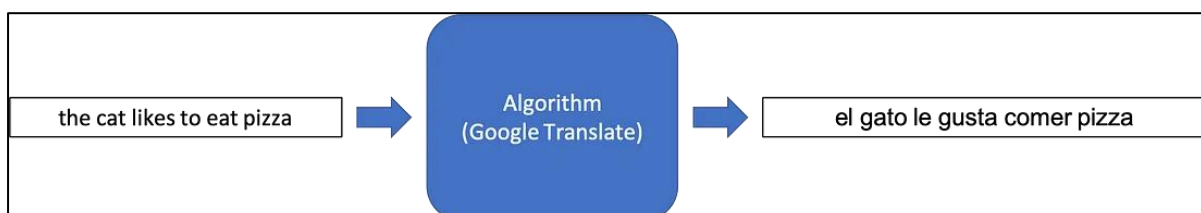


Fig4. Google trans basic functioning visualisation

One of the older and more established versions of NMT is the Encoder Decoder structure. This architecture is composed of two recurrent neural networks (RNNs) used together in tandem to create a translation model.[3] And when coupled with the power of attention mechanisms, this architecture can achieve impressive results.

The first step towards creating these vectors is to assign an index to each unique word in the input language, and then repeat this process for the output language. In assigning a unique index to each unique word, we will be creating what is referred to as a Vocabulary for each language. Ideally, the Vocabulary for each language would simply contain every unique word in that language. However, given that any single language can have hundreds of thousands of words, the vocabulary is often trimmed to the N most common words in the dataset we are working with (where N is chosen arbitrarily, but often ranges from 1,000–100,000 depending on the dataset size). To understand how we can then use a Vocabulary to create One Hot Encoding vectors for every word in our dataset, consider a mini-Vocabulary containing just the words in Table 1 below.

a	0
the	1
red	2
orange	3
blue	4
black	5
fish	6
whale	7
beaver	8
ate	9
drank	10
<SOS>	11
<EOS>	12

Fig5. NMT table for example

Given this table, we have assigned a unique index 0–12 to every word in our mini-Vocabulary. The <SOS> and <EOS> tokens in the table are added to every Vocabulary and stand for START OF SENTENCE and END OF SENTENCE respectively. Now to convert the words in the sentence “the blue whale ate the red fish” to their one hot encoding vectors:

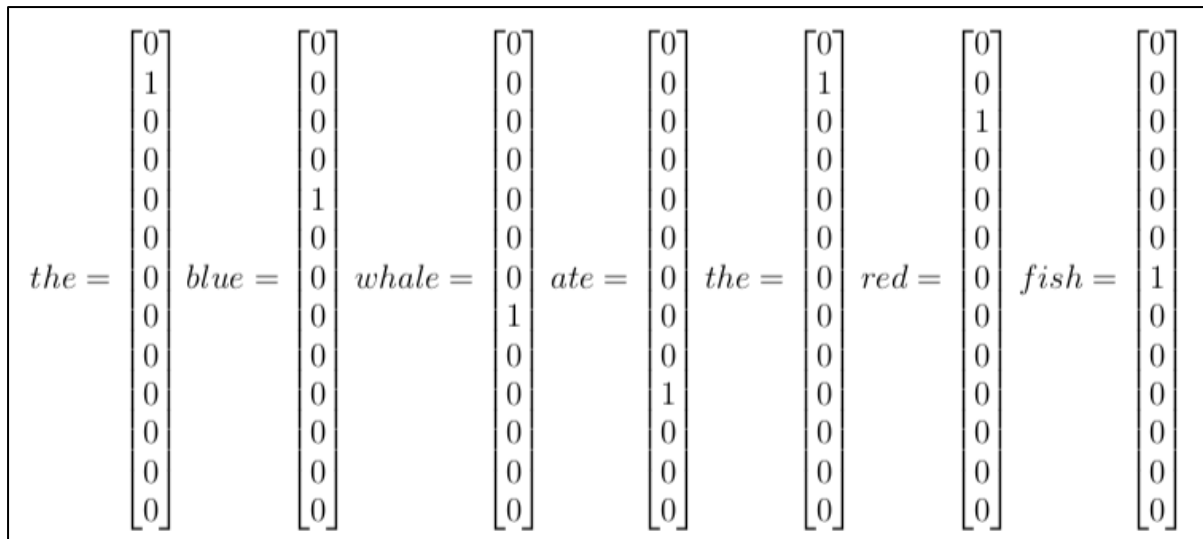


Fig6. One hot encoding visualisation

As you can see above, each word becomes a vector of length 13 (which is the size of our vocabulary) and consists entirely of 0s except for a 1 at the index that was assigned to that word.

At the most basic level, the Encoder portion of the model takes a sentence in the input language and creates a thought vector from this sentence. This thought vector stores the meaning of the sentence and is subsequently passed to a Decoder which outputs the translation of the sentence in the output language. This process is shown in the figure below.

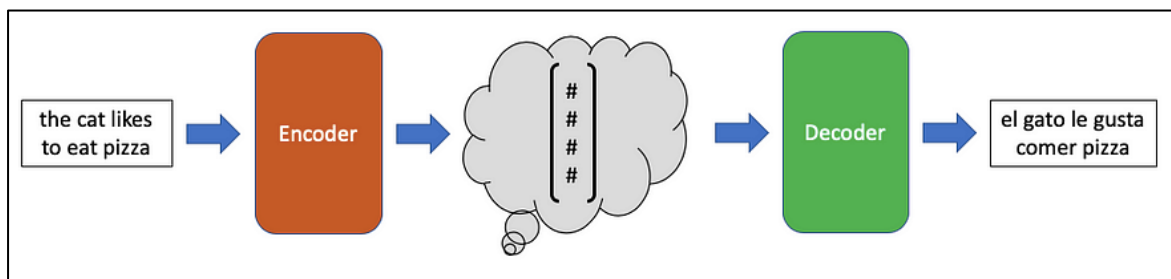


Fig7. Encoder to vector to decoder visualisation

Advantages of Neural Machine Translation (NMT):

- **Context-Aware Translation:** NMT models, including transformers, can produce translations that are more contextually accurate by considering the entire input sentence and generating translations word by word.
- **End-to-End Learning:** NMT models learn to translate directly from source to target language without relying on intermediate representations or handcrafted features, leading to simpler architectures and potentially better performance.
- **Better Handling of Morphology:** NMT models are better at handling morphologically rich languages compared to rule-based or statistical approaches, as they can capture complex patterns and dependencies in the data.

Disadvantages of Neural Machine Translation (NMT):

- **Data Dependency:** NMT models heavily rely on large amounts of parallel text data for training, which may not be available for all language pairs or domains. This data dependency can limit the applicability of NMT to languages with scarce resources.
- **Domain Adaptation:** NMT models trained on general domain data may not perform well when applied to specialized domains or specific types of text, requiring additional data or fine-tuning to adapt to different contexts effectively.

2.3 Statistical Machine Translation (SMT):

Statistical Machine Translation was first introduced in the 1940s by Warren Weaver. He believed that language worked kinda like solving math problems. So, he came up with the idea that we could use math to translate languages. It was a big deal because before that, people mostly translated languages using fixed rules.

SMT, along with another method called Neural Machine Translation (NMT), relies heavily on loads of data. Imagine having stacks of books full of sentences, words, and phrases in different languages. These piles of data are what SMT and NMT need to get better at translating languages.

SMT works by guessing the best translations based on probability. Imagine you're playing a guessing game, but instead of guessing which card your friend is holding, you're guessing which words or phrases in one language match the ones in another language.[4]

There are two main ways SMT does this: Word-based and Phrase-based.

Word-Based Statistical Machine Translation (SMT): This approach in SMT tries to figure out how likely it is for a word from one language to be turned into a word in another language. Basically, it looks at each word in a sentence and tries to guess the best word to use in the other language. This method is a type of learning where the computer learns from examples of sentences translated between languages, then it guesses the best translation for a word in one language based on how it's used in the other language.

Words can mean different things depending on where they're used. To keep track of this, SMT uses something called a "lexicon." Think of it as a big dictionary that tells the computer which words go together in different situations. So, when the computer sees a word, it checks the lexicon to see which translation makes the most sense based on where it is in the sentence.

Another important part of SMT is making sure each word in one language matches up with the right word in another language. This is called the Alignment Model. It's like connecting the dots between words in different languages to make sure the translation makes sense.

So, SMT is like a big guessing game, where the computer tries to figure out the best way to match up words and phrases in different languages to make accurate translations.

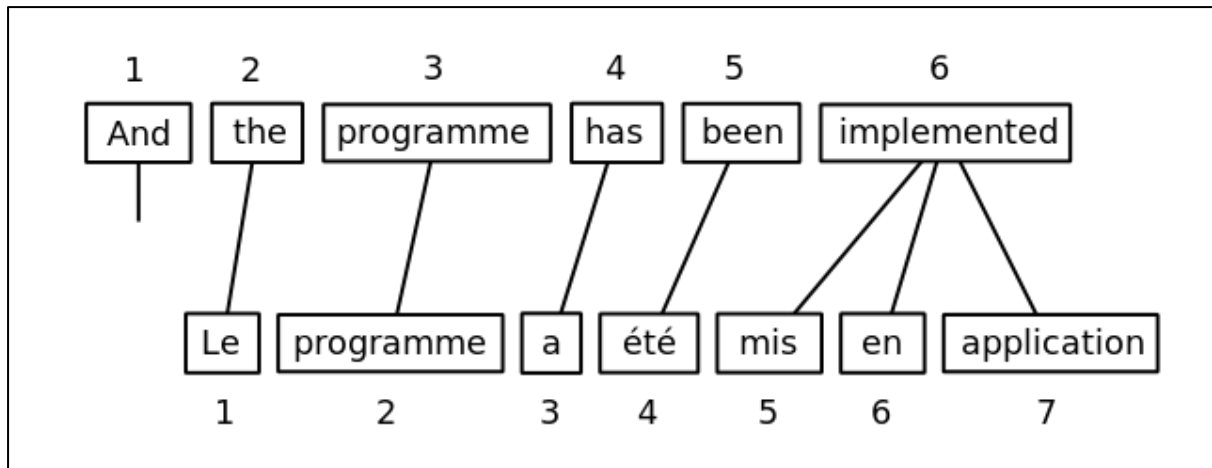


Fig8. SMT matching visualisation

Phrase-Based SMT: In this methodology, instead of dissecting the translation task into singular "words," the emphasis lies on consecutive groupings of words known as "Phrases." Consequently, translation endeavours to convert sets of two or three words jointly from a source sentence to a target sentence. Similar to the word-based approach, a crucial element is the "Alignment model." However, unlike the word-based method, which focuses solely on alignment from source to target, this approach takes into account both directions: source to target and target to source.

The second important component is the "Phrase Extraction process". The process occurs in 3 steps:

- Extract all phrase pairs consistent with the alignment. If source word in phrase pair
- Extract all phrase pairs consistent with the alignment. If target word in phrase pair
- Atleast one source word has to be aligned to one target word

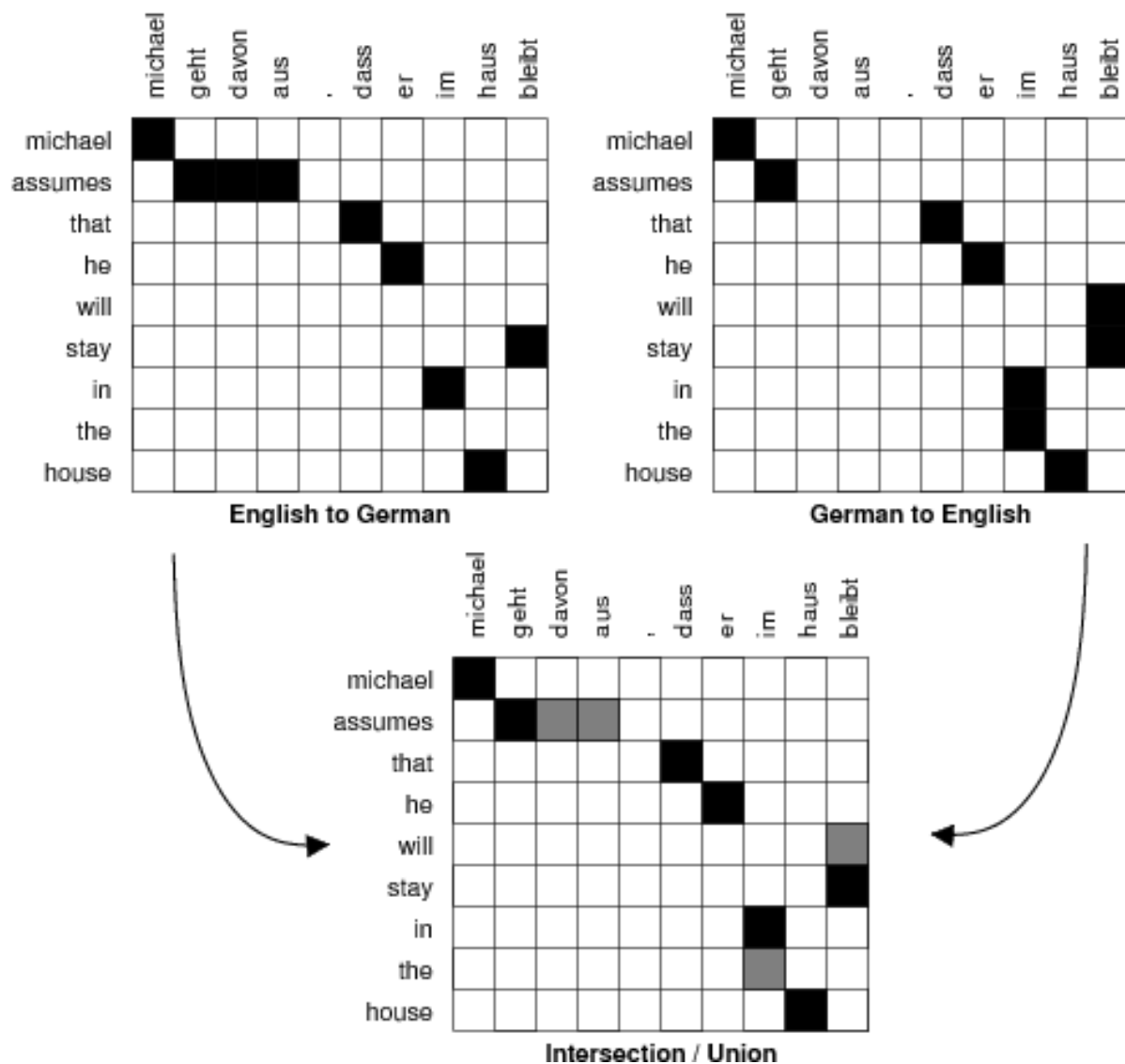


Fig9. Taking union of words from input and target languages

Advantages of Statistical Machine Translation (SMT):

- **Robustness:** SMT models can handle new or uncommon words or strange language phenomena better because they learn from big collections of data.
- **Interpretability:** SMT models are easier to understand compared to newer models like transformers because they use clear methods like phrase-based or syntax-based approaches to translate.

Disadvantages of Statistical Machine Translation (SMT):

- **Limited Contextual Understanding:** SMT models may struggle with capturing long-range dependencies or contextually relevant information, leading to less accurate translations compared to NMT models, especially for complex or ambiguous sentences.

- **Feature Engineering:** SMT systems usually need people to make special features like alignment models or language rules, which takes a lot of time and effort, especially for languages with complex syntax or morphology.

2.4 Google Cloud Translation:

Google Cloud Translation is a powerful tool in the realm of artificial intelligence and natural language processing. Developed by Google, it leverages advanced machine learning algorithms to provide accurate and efficient translation services across multiple languages.[5]

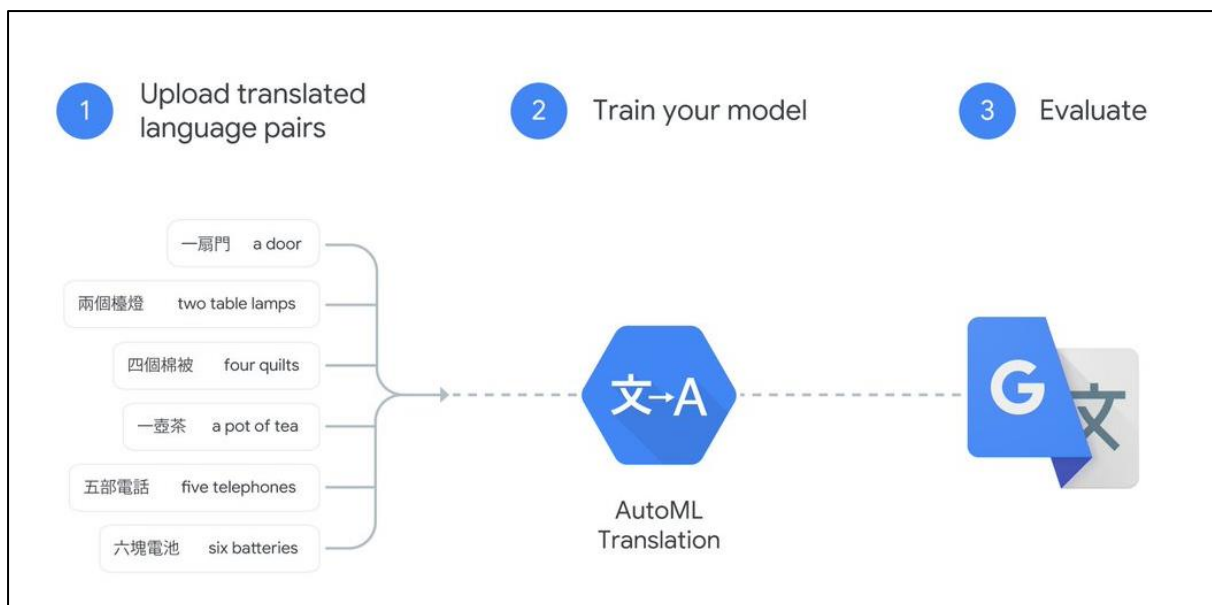


Fig10. Google cloud translate working visualisation

Advantages of Google Cloud Translation:

- **Accuracy:** Google Cloud Translation leverages advanced neural machine translation (NMT) models, which have been trained on vast amounts of data to produce highly accurate translations across a wide range of language pairs. This ensures that translated content maintains its original meaning and context, even for complex sentences or specialized domains.
- **Scalability:** As a cloud-based service, Google Cloud Translation offers scalability to handle translation requests of any scale, from individual phrases to large documents or entire websites. This scalability makes it suitable for businesses and organizations with varying translation needs, ensuring consistent performance and responsiveness.
- **Multilingual Support:** Google Cloud Translation supports a comprehensive list of languages, covering a majority of the world's spoken and written languages. This expansive language coverage enables users to translate content between virtually any language pair, facilitating communication and collaboration on a global scale.
- **Integration:** Google Cloud Translation seamlessly integrates with other Google Cloud services and APIs, as well as third-party platforms and applications. This

integration allows developers to incorporate translation capabilities into their existing workflows and applications easily, enhancing user experience and accessibility across various platforms and devices.

- **Customization:** Google Cloud Translation offers customization options, allowing users to fine-tune translation models for specific domains, industries, or language pairs. This customization ensures that translations are tailored to meet the unique requirements and preferences of individual users or organizations, improving translation quality and consistency.

Disadvantages of Google Cloud Translation:

- **Privacy Concerns:** As a cloud-based service, Google Cloud Translation processes and stores data on Google's servers, raising potential privacy concerns, particularly for sensitive or confidential information. Users should review Google's privacy policies and consider data protection measures when using Google Cloud Translation for translating sensitive content.
- **Language Limitations:** While Google Cloud Translation supports a wide range of languages, certain language pairs or dialects may have limited support or availability. Users should verify language support and coverage for their specific translation requirements to ensure compatibility with Google Cloud Translation.
- **Customization Complexity:** Customizing translation models in Google Cloud Translation may require expertise in machine learning and natural language processing, as well as significant time and effort for experimentation and optimization. Users should be prepared to invest resources into customizing translation models to achieve desired performance and accuracy levels.

Overall, Google Cloud Translation represents a cutting-edge solution for overcoming language barriers in an increasingly interconnected world. By harnessing the power of AI and machine learning, it empowers businesses and individuals to communicate effectively across linguistic boundaries, fostering collaboration, innovation, and cultural exchange on a global scale.

2.5 Live Translate:

Live translation, also known as real-time translation, refers to the process of translating spoken or written language into another language instantly, as the communication occurs. At its core, live translation relies on sophisticated algorithms and models to accurately interpret and translate language in real time. Speech recognition technology converts spoken words into text, which is then analyzed and translated using machine translation algorithms. Natural language processing techniques are applied to understand the context and nuances of the input text, ensuring that translations are contextually relevant and accurate. Finally, the translated text is synthesized into speech or displayed as text to the intended recipient.[7]

Advances in artificial intelligence and machine learning have significantly improved the capabilities of live translation systems, enabling them to handle complex language structures, idiomatic expressions, and dialectal variations with increasing accuracy and

fluency. These technologies have made live translation accessible across various platforms and devices, including mobile apps, web services, and live interpretation services.

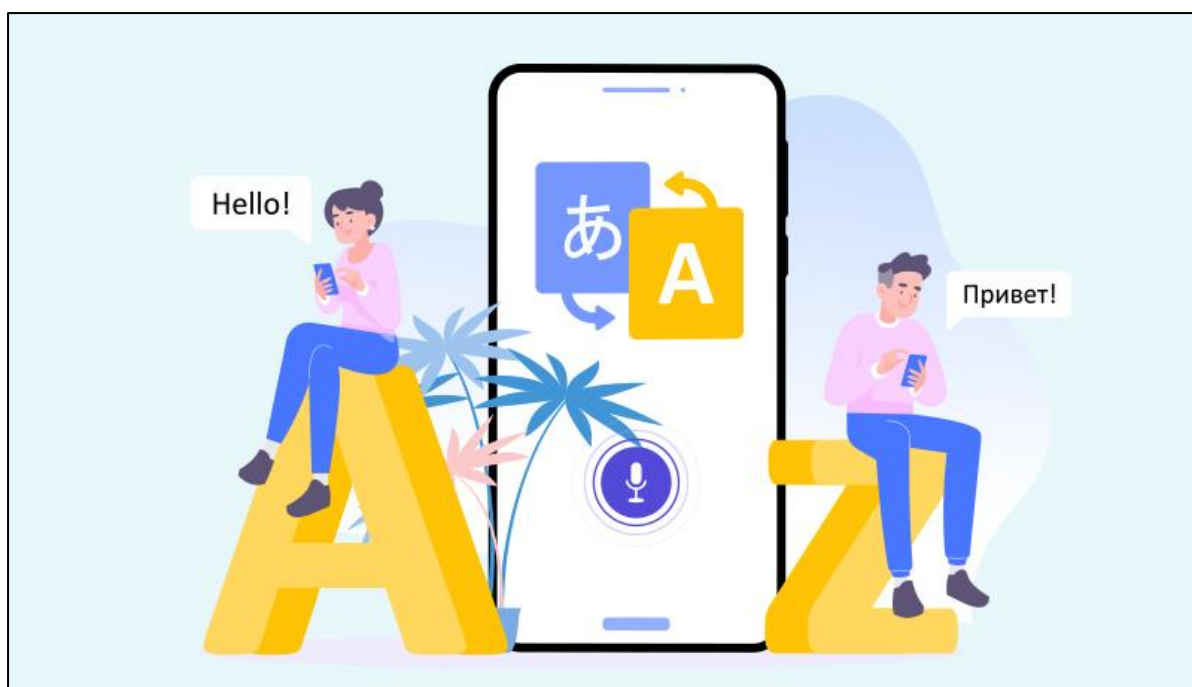


Fig11. Text to text visualisation

Advantages of Live Translation:

- **Instant Communication:** Live translation enables instant communication between individuals who speak different languages, overcoming language barriers and facilitating seamless interaction in real time.
- **Enhanced Accessibility:** Live translation makes information and resources more accessible to a global audience, allowing individuals to participate in discussions, access content, and engage with others regardless of their language proficiency.
- **Improved Collaboration:** Live translation fosters collaboration and cooperation in diverse settings, such as international meetings, conferences, and business negotiations, by enabling participants to communicate effectively across language boundaries.
- **Cultural Exchange:** Live translation promotes cultural exchange and understanding by facilitating communication between speakers of different languages, fostering mutual respect, appreciation, and empathy across linguistic and cultural divides.

Disadvantages of Live Translation:

- **Accuracy Challenges:** Despite advances in machine translation technology, live translation systems may still encounter accuracy challenges, especially in handling complex or ambiguous language structures, cultural nuances, and domain-specific terminology.
- **Technical Limitations:** Live translation systems rely on complex technologies such as speech recognition and machine translation, which may be prone to technical issues

such as errors in speech recognition, misinterpretation of context, and inaccuracies in translation output.

2.6 Automatic Speech Recognition (ASR):

Automatic Speech Recognition (ASR) is a technology that enables the conversion of spoken language into text, allowing computers to interpret and process spoken commands, dictation, or natural language input. ASR systems utilize a combination of acoustic modeling, language modeling, and speech decoding techniques to accurately transcribe spoken words into text. [8]

Acoustic modeling involves analysing the audio signal to extract relevant features such as spectral characteristics, pitch, and duration, which are then used to represent speech sounds.[9] Language modeling helps ASR systems understand the structure and context of spoken language, enabling them to predict the likelihood of word sequences and improve transcription accuracy. Speech decoding algorithms combine acoustic and language models to search for the most probable sequence of words that best matches the input speech signal.[10]

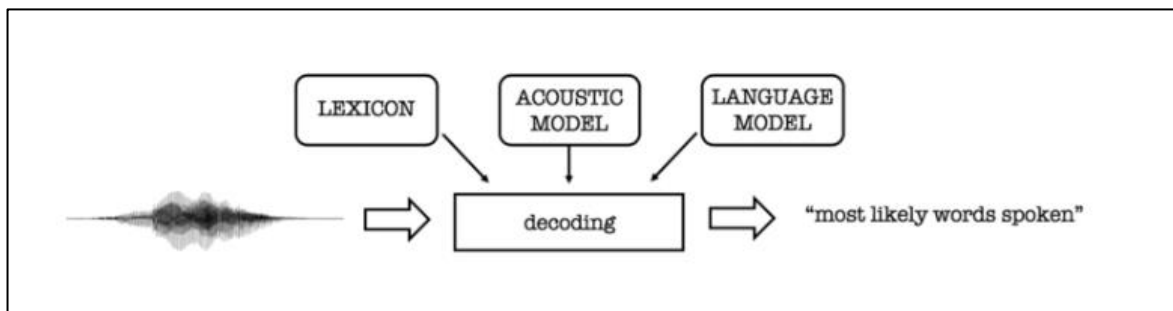


Fig12. Automatic speech recognition

Advantages of Automatic Speech Recognition (ASR):

- **Hands-Free Interaction:** ASR enables hands-free interaction with computers and devices, allowing users to control applications, compose text, or perform tasks using voice commands without the need for manual input.
- **Accessibility:** ASR makes information and services more accessible to individuals with disabilities or impairments, such as visual or motor impairments, by providing alternative means of input and interaction through speech.
- **Multimodal Integration:** ASR can be integrated with other modalities such as text-based input, gesture recognition, and natural language understanding to create more versatile and interactive user interfaces that support multimodal interaction.
- **Scalability:** ASR technology is highly scalable and adaptable to different languages, dialects, and domains, making it suitable for a wide range of applications and use cases, from personal assistants to enterprise communication systems.

Disadvantages of Automatic Speech Recognition (ASR):

- **Accuracy Challenges:** Despite advancements in ASR technology, accuracy can still be affected by factors such as background noise, speaker accents, speech variability, and domain-specific vocabulary, leading to errors in transcription.
- **Limited Vocabulary and Context Understanding:** ASR systems may struggle to accurately transcribe speech containing uncommon or specialized vocabulary, slang, or ambiguous language constructs, leading to errors or misinterpretations.
- **Dependency on Audio Quality:** ASR performance is highly dependent on the quality of the audio input, including factors such as microphone quality, background noise levels, and recording conditions. Poor audio quality can degrade transcription accuracy and usability.

2.7 Optical Character Recognition (OCR):

Optical Character Recognition (OCR) is a technology that enables the conversion of printed or handwritten text into digital text, allowing computers to interpret and process text from scanned documents, images, or physical surfaces.[11] OCR systems utilize image processing, pattern recognition, and machine learning algorithms to identify and recognize characters and symbols within an image and convert them into machine-readable text.

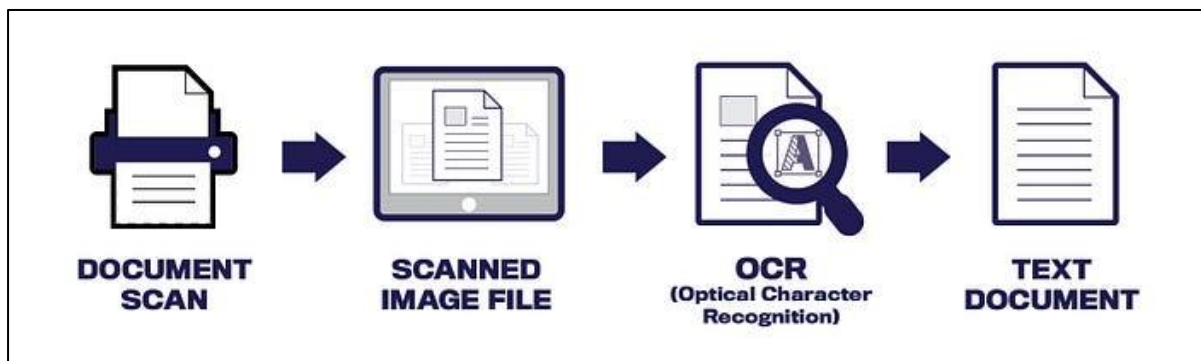


Fig13. Optical Character Recognition working

Image pre-processing techniques are used to enhance the quality of the input image, including operations such as noise reduction, binarization, and image normalization. Pattern recognition algorithms analyse the visual patterns and features of individual characters, such as shape, size, and stroke structure, to classify and identify them accurately. Machine learning models, including deep learning architectures like convolutional neural networks (CNNs), are often employed to improve OCR performance by learning from large datasets of annotated text and images.[12]

Advancements in OCR technology have led to significant improvements in accuracy, speed, and versatility, making OCR systems widely used in various applications, including document digitization, text extraction, and automated data entry.

Advantages of Optical Character Recognition (OCR):

- **Digitization of Documents:** OCR enables the digitization of printed documents, handwritten notes, and other text-based materials, making them searchable, editable, and accessible in digital formats.
- **Text Extraction:** OCR systems can extract text from images, PDFs, and other scanned documents, allowing users to copy, edit, or repurpose text content for various purposes, such as data analysis, information retrieval, and content management.
- **Automation of Data Entry:** OCR automates the process of data entry by converting scanned documents into machine-readable text, reducing manual data entry errors and accelerating document processing workflows.
- **Improved Accessibility:** OCR enhances accessibility for individuals with visual impairments by converting printed or handwritten text into digital formats that can be read aloud by text-to-speech (TTS) systems or displayed in enlarged fonts for easier readability.

Disadvantages of Optical Character Recognition (OCR):

- **Accuracy Limitations:** Despite advancements in OCR technology, accuracy can be affected by factors such as image quality, font style, text layout, and language complexity, leading to errors or inaccuracies in text recognition.
- **Handwriting Recognition Challenges:** OCR systems may struggle to accurately recognize handwritten text, especially cursive handwriting or stylized fonts, due to variations in writing styles, letter shapes, and penmanship.
- **Processing Time:** OCR processing can be time-consuming, especially for large documents or images with high-resolution graphics, as it requires intensive computational resources for image processing, character recognition, and text analysis.

2.8 Website Translation

Website translation involves the process of translating the content of a website from one language into another, enabling multilingual users to access and understand the website's information, products, and services in their preferred language. Website translation typically encompasses translating web pages, navigation menus, product descriptions, user interface elements, and other textual content to cater to the linguistic and cultural preferences of diverse audiences.

Advantages of Website Translation:

- **Enhanced User Experience:** Website translation enhances user experience by providing content in users' native languages, improving comprehension, engagement, and satisfaction, and reducing language-related barriers to access and navigation.
- **Improved SEO and Visibility:** Website translation can improve search engine optimization (SEO) and visibility in international markets by enabling localized

content targeting relevant keywords and search queries in different languages, driving organic traffic and visibility.

- **Cultural Sensitivity:** Website translation facilitates cultural sensitivity and localization by adapting content to the linguistic, cultural, and regulatory requirements of target markets, ensuring relevance, appropriateness, and compliance with local norms and conventions.

Disadvantages of Website Translation:

- **Technical Challenges:** Website translation may pose technical challenges related to website architecture, design, and functionality, such as dynamic content generation, multilingual SEO optimization, and responsive web design, requiring technical expertise and coordination to address effectively.
- **Quality Control:** Website translation requires rigorous quality control and assurance processes to ensure linguistic accuracy, consistency, and coherence across languages, as well as adherence to brand guidelines, style preferences, and localization standards, which may require dedicated resources and expertise.

2.9 Document Translation

Document translation involves the process of translating written documents, texts, or content from one language into another, preserving the meaning, tone, and style of the original text while ensuring linguistic accuracy, coherence, and readability in the target language. Document translation requires linguistic proficiency, cultural competence, subject matter expertise, and specialized translation skills to accurately convey the intended meaning, terminology, and nuances of the source text in the target language. Document translation encompasses a wide range of textual materials, including legal documents, contracts, technical manuals, academic papers, marketing materials, and literary works, among others.



Fig14. Document translation idea visualisation

Advantages of Document Translation:

- **Cross-Cultural Communication:** Document translation enables cross-cultural communication and understanding by facilitating the exchange of information, ideas, and perspectives across linguistic and cultural boundaries, fostering mutual respect, appreciation, and empathy among individuals and communities.
- **Legal and Business Transactions:** Document translation is essential for legal and business transactions, including contracts, agreements, licenses, and certifications, enabling parties to negotiate, execute, and enforce agreements in multiple languages, ensuring compliance with legal and regulatory requirements.
- **Knowledge Sharing and Education:** Document translation promotes knowledge sharing and education by making educational materials, scholarly research, scientific publications, and technical documentation accessible in different languages, supporting learning, research, and innovation on a global scale.
- **Cultural Preservation:** Document translation contributes to cultural preservation and heritage conservation by translating literary works, historical documents, religious texts, and indigenous knowledge into other languages, ensuring the preservation and dissemination of cultural heritage and identity.

Disadvantages of Document Translation:

- **Complexity and Specialization:** Document translation can be complex and specialized, requiring linguistic proficiency, subject matter expertise, and specialized terminology knowledge to accurately translate technical, legal, scientific, or academic content, which may pose challenges for translators without domain-specific expertise.
- **Quality Assurance:** Document translation requires rigorous quality assurance and proofreading processes to ensure translation accuracy, consistency, and readability, as well as adherence to client specifications, style guidelines, and formatting conventions, which may require additional time, effort, and resources.
- **Cultural and Linguistic Challenges:** Document translation may encounter cultural and linguistic challenges, such as idiomatic expressions, cultural references, and linguistic nuances that may not have direct equivalents in the target language, requiring creative adaptation, localization, or explanation to convey the intended meaning accurately and effectively.

CHAPTER 3

METHODOLOGY

The flowchart shown below describes the whole making structure of the AI based Language translation web-application

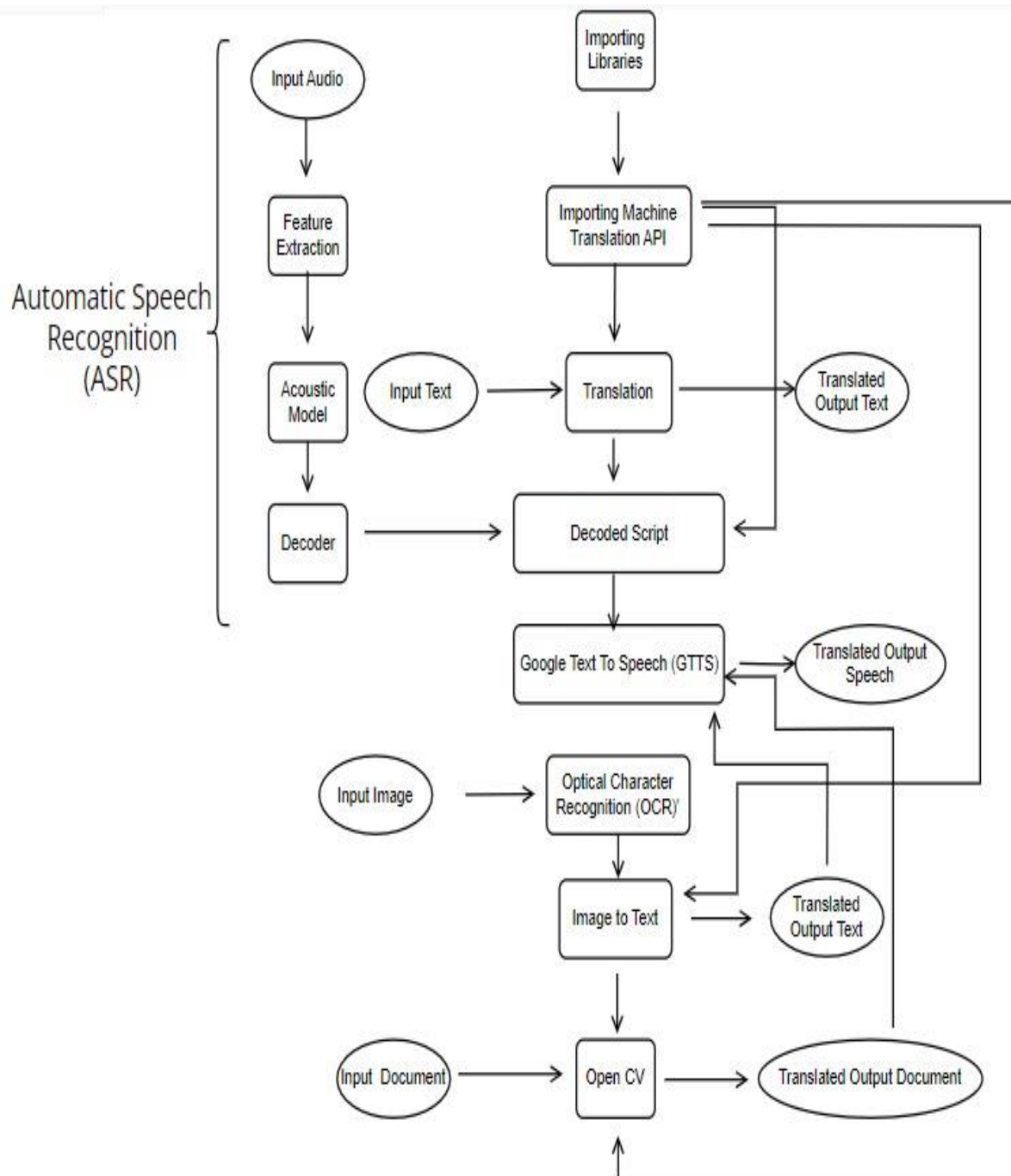


Fig15. Flowchart of methodology

3.1 Data Collection and Preparation:

Data Collection:

The data collection process for our language translation system was critical to ensuring comprehensive language coverage and robust translation capabilities. We employed a multi-faceted approach to gather the necessary data, which involved:

- **Compilation of Language List:** We began by compiling a comprehensive list of languages that our translation system would support. This list was curated from various sources, including official language repositories, linguistic databases, and international standards organizations. Languages were selected based on factors such as global prevalence, linguistic diversity, and user demand.
- **Research and Verification:** Each language included in our list underwent thorough research and verification to ensure correct language codes and names.
- **Multilingual T5:** The Multilingual T5 (MT5) model is a multilingual adaption of the Text-to-Text Transfer Transformer (T5) that was pretrained on the mC4 dataset, which includes 101 languages. This variation, MT5, inherits T5's architecture and capabilities but is specifically built to perform multilingual text processing jobs efficiently. The MT5-small model is fine-tuned using the XNLI 15 dataset, a parallel corpus with 15 columns, each representing a distinct language. The columns in this dataset are tab-separated, with each column acting as a target text throughout the fine-tuning process and having a header indicating the language connected with the particular text. This fine-tuning process takes use of MT5-small's multilingual nature, allowing it to learn and generalise across the several languages represented in the XNLI 15 dataset.

Data Preparation: Once the language list was finalized, we proceeded with preparing the data for integration into our translation system. This involved:

- **Data Preparation:** The XNLI dataset is preprocessed and organised into a two-column data frame with headers labelled 'input_text' and 'target_text'. After preprocessing, a special token with the prefix is injected into the tokenizer.[13]

The inclusion of this token is important to the model's capacity to recognise and interpret multilingual inputs efficiently.[14] It guarantees that the tokenizer can correctly handle and encode language-specific information inherent in the text input. This tokenization method also seeks to align the total number of tokens with the model's configured vocabulary size.[15] When new token additions surpass the specified vocabulary size, the model's input token embeddings matrix is resized to make the necessary modifications. This resizing procedure is critical for maintaining consistency and compatibility between the tokenizer and the underlying model architecture improve the model's ability to handle and analyse multilingual input texts from the XNLI dataset.

Figure shows the data frame headers being the “input_text” and “target_text” in the backend:

	input_text	target_text
0	وقال، ماما، لقد عدت للمنزل	ar
1	...حسنا ، لم أكن أفكر حتى حول ذلك ، لكن كنت محبطا	ar
2	...واعتقدت أن ذلك شرف لي ، ولا يزال ، ولا يزال ، لك	ar
3	... أخبروني ، إيه، أنه سيتم استدعائي من قبل شاب في	ar
4	...هذه الكثير تستطيع التحدث عنه وأنا سوف أناجوا	ar
...
32410	唱着好运来到我所害怕的人身边，	zh
32411	目前脱口秀的客人经常接受过如何避免回答问题的正式培训，每个3岁的孩子都知道如何提供预先包装的声音。	zh
32412	John Horgan对 Richard Dawkins的《攀登不可能的山》的评论很有趣	zh
32413	索德伯是那些在工作中学习的少数电影制作人之一。	zh
32414	NaN	zh

32415 rows × 2 columns

Fig16. Dataset

- **Data Structuring:** We organized the language data into a structured format that could be easily accessed and manipulated by our system. The data structure was designed to accommodate diverse language attributes, such as names, codes, scripts, and regional variations. This facilitated efficient retrieval and utilization of language information during the translation process. Languages and codes were stored in a dictionary for better retrieval and comparison.

The following figure shows the languages supported by the model:

```
# List of supported languages
dic = {
    'Afrikaans': 'af', 'Albanian': 'sq', 'Amharic': 'am', 'Arabic': 'ar', 'Armenian': 'hy', 'Azerbaijani': 'az',
    'Belarusian': 'be', 'Bengali': 'bn', 'Bosnian': 'bs', 'Bulgarian': 'bg', 'Catalan': 'ca', 'Cebuano': 'ceb',
    'Chinese (Simplified)': 'zh-cn', 'Chinese (Traditional)': 'zh-tw', 'Corsican': 'co', 'Croatian': 'hr',
    'Dutch': 'nl', 'English': 'en', 'Esperanto': 'eo', 'Estonian': 'et', 'Filipino': 'tl', 'Finnish': 'fi',
    'Frisian': 'fy', 'Galician': 'gl', 'Georgian': 'ka', 'German': 'de', 'Greek': 'el', 'Gujarati': 'gu',
    'Hausa': 'ha', 'Hawaiian': 'haw', 'Hebrew': 'he', 'Hindi': 'hi', 'Hmong': 'hmn', 'Hungarian': 'hu',
    'Indonesian': 'id', 'Irish': 'ga', 'Italian': 'it', 'Japanese': 'ja', 'Javanese': 'jw', 'Kannada': 'kn',
    'Korean': 'ko', 'Kurdish (Kurmanji)': 'ku', 'Kyrgyz': 'ky', 'Lao': 'lo', 'Latin': 'la', 'Latvian': 'lv',
    'Luxembourgish': 'lb', 'Macedonian': 'mk', 'Malagasy': 'mg', 'Malay': 'ms', 'Malayalam': 'ml', 'Maltese': 'mt',
    'Marathi': 'mr', 'Mongolian': 'mn', 'Myanmar (Burmese)': 'my', 'Nepali': 'ne', 'Norwegian': 'no', 'Persian': 'fa',
    'Polish': 'pl', 'Portuguese': 'pt', 'Punjabi': 'pa', 'Romanian': 'ro', 'Russian': 'ru',
    'Scots Gaelic': 'gd', 'Serbian': 'sr', 'Sesotho': 'st', 'Shona': 'sn', 'Sindhi': 'sd', 'Sinhala': 'si',
    'Slovenian': 'sl', 'Somali': 'so', 'Spanish': 'es', 'Sundanese': 'su', 'Swahili': 'sw', 'Swedish': 'sv',
    'Telugu': 'te', 'Thai': 'th', 'Turkish': 'tr', 'Ukrainian': 'uk', 'Urdu': 'ur', 'Uyghur': 'ug', 'Uzbek': 'uz',
    'Welsh': 'cy', 'Xhosa': 'xh', 'Yiddish': 'yi', 'Yoruba': 'yo', 'Zulu': 'zu'
}
```

Fig17. Languages supported

- **Normalization and Standardization:** To ensure consistency and compatibility across languages, we normalized and standardized the data elements. This involved aligning language names, codes, and other attributes according to established conventions and standards. Additionally, we implemented normalization techniques to handle case sensitivity, diacritics, and special characters.[16]

- **Encoding Configuration T5:** In the context of encoder-decoder attention in a model such as T5 (Text-to-Text Transfer Transformer), extra parameters are inserted in the decoder to aid the attention process. Encoder-decoder attention is an important component in tasks such as sequence-to-sequence learning and machine translation, in which the decoder attends to the encoded representations (outputs) of the input sequence provided by the encoder.[17] This attention mechanism computes attention ratings for each place in the decoder and the encoder output.

However, one significant disadvantage of this attention mechanism is its computing cost, which can grow quadratic in relation to the sequence lengths of the input and output. Specifically, the length of the input sequence (from the encoder) is indicated as (N) , while the length of the output When the sequence from the decoder is indicated as (M) , the number of attention operations required scales as $(O(N \times M))$. This quadratic scaling can cause increasing processing demands, particularly for longer sequences, affecting both training time and memory needs.

To reduce these computing expenses, many strategies have been investigated, including:

- **Scaled Dot-Product Attention:** This approach divides the attention scores by the square root of the key vectors' dimensionality to assist stabilise the gradients throughout training.
- **Effective Attention Mechanisms:** Implementations such as sparse attention or approximation attention limit the amount of operations by focusing solely on important sections of the input sequence, hence addressing the quadratic scaling problem.
- **Attention Masking:** Methods such as padding masks or future Masks are utilised to avoid attention to padding tokens or future locations, minimising redundant calculations.

The below graph describes token lengths histogram of the input and output texts:

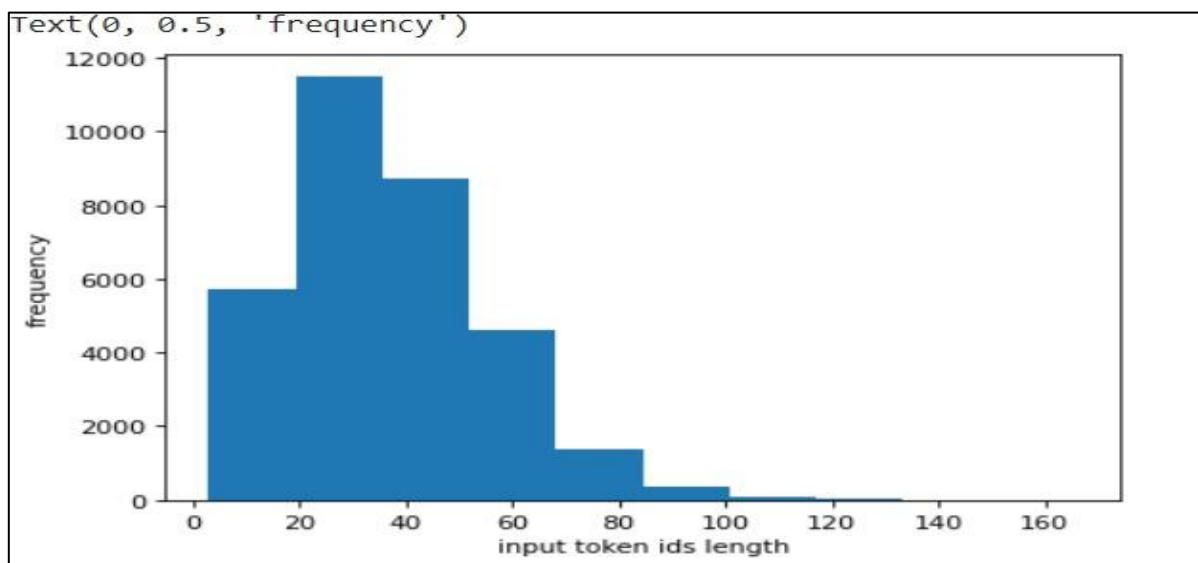


Fig18. Token lengths histogram of the input

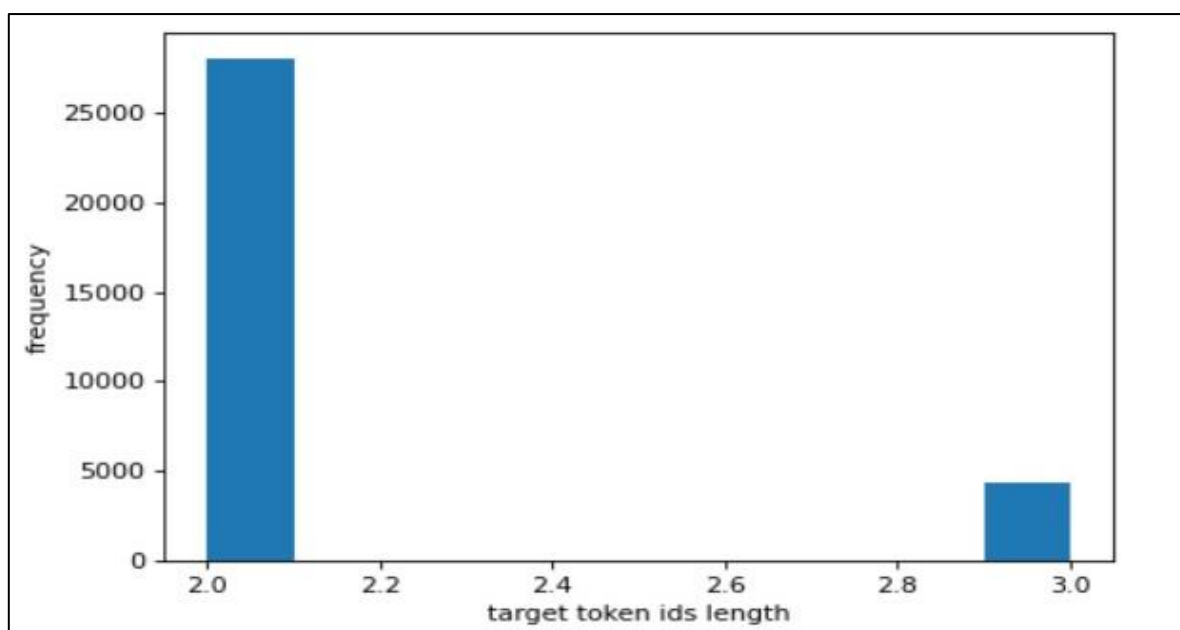


Fig19. Token lengths histogram of the output texts

- The maximum input sequence length is set to 40.
- The maximum target sequence length is set to 3.
- `truncation=True` truncates the sequence to a maximum length specified by the `max_length` argument.
- `padding='max_length'` pads the sequence to a length specified by the `max_length` argument.

The histogram on the top shows the distribution of input token lengths. The x-axis represents the length of the token in IDs, and the y-axis represents the frequency that a token of that length appears in the data. For instance, there are more tokens with a length of around 40 IDs than any other length.

The histogram on the bottom shows the distribution of target token lengths. The x-axis represents the length of the token in IDs, and the y-axis represents the frequency that a token of that length appears in the data. There are many more target tokens with a length between 2.0 and 2.2 IDs than any other length.

Overall, the distribution of input token lengths is much wider than the distribution of target token lengths. This suggests that the target tokens tend to be much shorter than the input tokens.

- **Training Results:** The optimizer used in this is AdamW, with a learning rate of $5e-4$, and the learning schedule is a linear schedule with a warmup, in which the learning rate initially decreases linearly from the initial learning rate set in the optimizer to 0, and then increases linearly from 0 to the initial learning rate set in the optimizer, reducing the primacy effect of the early training examples. The graph depicts the

training and validation losses obtained during fine-tuning. The `Model.forward()` function is used to do a forward pass using a batch of input and target token ids. It generates the right `decoder_input_ids` required for training.

The calculated loss is used to update the model weights for each training step. The model is validated and saved at regular intervals.

- **Model Testing:** The stated model test accuracy of 99.5% shows a high degree of performance in generating outputs that closely match the intended goals during assessment.

Below figure shows the visualization of the training and validation losses:

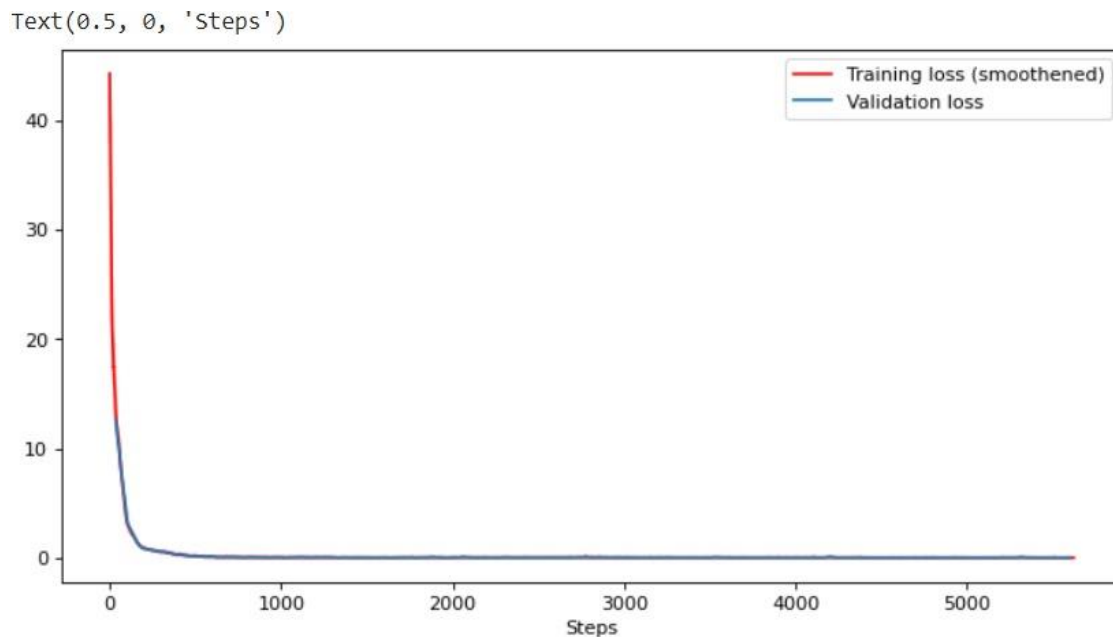


Fig20. Visualization of the training and validation losses

The training loss (red line) appears to be generally decreasing as the number of training steps increases. This suggests that the model is learning and improving its performance on the training data.

The validation loss (blue line) is also decreasing, but it is higher than the training loss. This is typical, as the validation loss is usually higher than the training loss because the model has not been trained on the validation data.

There appears to be some noise in the validation loss (blue line). This is common in real-world data, and it can make it difficult to determine the exact trend of the loss function.

Out of 10,000 examples in which the model is tested only 51 were wrongly predicted.

The below table shows some of the wrong predictions predicted:

	Input_text	True_target	Predicted
00	<idf.lang> andaroni circle ki membership muft nahi hai.	ur	hi
01	<idf.lang> aisa lag raha tha k ye hmaisha k liye hai.	ur	hi
02	<idf.lang> Brock no defiende a Hillary.	es	en
03	<idf.lang> Huevos means balls.	en	th
04	<idf.lang> До свидания!	ru	bg
05	<idf.lang> Той прегърна широко лорд Джулиан.	bg	ru
06	<idf.lang> Mujhe aik glass chocolate ka doodh chahiye	ur	hi
07	<idf.lang> (а) Променете всяко d или t в целта до с.	bg	ru
08	<idf.lang> aur aisa hi tha jaise wo ise mustarad kar rahi thi, kuch tareqon se, jis trha se isne apke sath bartao kiya, dosre nawase.	ur	hi
09	<idf.lang> Кальдас де Мончик окружен лесами.	ru	bg

Fig21. Some of the wrong predictions predicted

Results:

- Nearly 40% wrong predictions are from Urdu language sentences predicted as Hindi and vice versa.
- 20% wrong predictions are from Russian language sentences predicted as Bulgarian and vice versa.
- 10% others are other wrongly predicted outputs.

3.2 Text Language Translation:

After the language prediction, the next step is language translation. In this step we use the text or audio or image's predicted text to translate the text in user desired language. For that purpose, we have used transformers and API's to build the model. Let us know a little about both transformer's architecture and API's used:

a. Language Translations using Transformers:

The transformer architecture, described in the seminal work "Attention Is All You Need" by Vaswani et al. (2017), transformed sequence-to-sequence tasks like language translation. This architecture is built on the notion of multi-head attention, and it works by converting textual inputs into numerical representations called tokens.[18] Each token is turned into a vector representation via a process known as word embedding, in which the model pulls up pre-trained embeddings from a database.

The transformer architecture is made up of two major components: the encoder and the decoder, which work together to assist the translation process:

- **Encoder:** The encoder converts the input sequence into a meaningful representation that includes contextual information and semantic meaning. This is accomplished with a stack of identical layers which are made up of sub-layers that include multi-head self-attention mechanisms and position-wise completely integrated feed-forward networks. Each token in the encoder is contextualised in a context pane with other tokens utilising multi-head attention.[19] This attention mechanism enables the model to assign different levels of relevance to different tokens, boosting the signal for significant tokens while suppressing less relevant ones.
The encoder produces a sequence of contextualised representations (or embeddings) for each character in the input sequence. These representations capture the meaning and context of the input sequence and are subsequently sent to the decoder.
- **Decoder:** The decoder constructs the output sequence based on the encoder's contextualised representations. It also has an attention technique known as encoder-decoder attention, which enables the decoder, like the encoder, it is made up of multiple layers with sub-layers such as masked multi-head self-attention, encoder-decoder attention, and position-wise fully connected feed-forward networks.

During translation, the decoder employs its own self-attention process to create predictions one token at a time, based on the encoder's output and prior tokens.

The transformer architecture's strength is its ability to capture long-range relationships, manage variable-length inputs and outputs, and efficiently use attention methods.

Transformers excel at modelling complicated connections among sequences by including self-attention mechanisms into both the encoder and decoder, making them ideal for tasks such as machine translation, text production, and natural language understanding. The encoder-decoder paradigm allows transformers to produce accurate and contextually relevant translations by utilising learnt representations from input sequences.

Figure shows pseudo code for the “Language Translations using Transformer” model:

```

# Import necessary libraries
Import MBartForConditionalGeneration and MBart50Tokenizer from Transformers
Import Streamlit for web interface
Import 'languages' dictionary with language names and corresponding codes

# Set Streamlit page configuration for wide layout
Set Streamlit page layout to 'wide'

# Sidebar for language selection
Display a sidebar header "Please choose the languages"
Allow user to select a source language from 'languages'
Store the selected source language in 'src_lang'

Allow user to select a target language for translation from 'languages'
Store the selected target language in 'trans_lang'

# Get the language codes for the selected source and target languages
Retrieve 'src_lang_code' by looking up the language code for 'src_lang' in 'languages'
Retrieve 'trans_lang_code' by looking up the language code for 'trans_lang' in 'languages'

# Cached function to download the MBart model and tokenizer
Define 'download_model()' function with @st.cache decorator:
    - Define the model name "facebook/mbart-large-50-many-to-many-mmt"
    - Load the MBart model from the pre-trained model
    - Load the MBart tokenizer from the pre-trained model
    - Return the loaded model and tokenizer

# Main Streamlit interface
Display a Streamlit title "Lingualink"

# Text area for user input
Create a text area for user to enter the text for translation
Store the user input in 'text'

```

Fig22. Pseudo code for the Translations using Transforme model (1)

```

# Download the MBart model and tokenizer
Call 'download_model()' to get the MBart model and tokenizer

# If the "Translate" button is clicked
If 'Translate' button is pressed:
    - If 'text' is empty, display a warning message "Please enter the required text for translation!"
    - Else, with a Streamlit spinner for loading:
        - Set 'tokenizer.src_lang' to 'src_lang_code'
        - Tokenize the input text with 'tokenizer'
        - Generate tokens using 'model.generate()' with the target language code
        - Decode the generated tokens to get the translated text
        - Display a success message "Translation Complete!"
        - Output the translated text in Streamlit

# If the "Translate" button is not pressed, do nothing
Else: Pass

```

Fig23. Pseudo code for the Translations using Transformer model (2)

b. Language Translation using APIs:

Language translation using APIs is a straightforward and fast approach to include translation capabilities into applications without the need for new models.[20] Here's a summary of the two major cloud translation APIs supplied by Google and Microsoft:

- **Google Cloud Translation API:** The Google Cloud Translation API uses a cutting-edge Neural Machine Translation (NMT) model for language translation.

The API provides translation between over 133 languages, with a diverse selection of language pairings for both common and less widely spoken languages. The NMT model used by Google Cloud Translation API is trained on massive amounts of multilingual data to offer accurate and contextually appropriate translations. The Google Cloud Translation API integrates seamlessly with other Google Cloud services and includes capabilities including batch translation, glossary customisation, and language detection.

Now, coming to the model, usually language translation jobs may be tackled in two ways:

- creating a bespoke translation model using fundamental machine learning (ML) and natural language processing (NLP) techniques
- leveraging existing translation APIs provided by companies such as Google Cloud Translation.

Each strategy provides various benefits and considerations based on individual use cases and requirements:

Creating Your Own Translation Model: Customisation and Control. Building a bespoke translation model enables fine-tuning and customisation depending on unique domain or language needs. Developers have control over the model's architecture, training data, and optimisation algorithms.

Tailored Solutions: Custom models may be trained on domain-specific datasets to provide more accurate translations for specialised vocabularies and circumstances. **Resource Intensive:** Developing and training a bespoke translation model involves extensive computer resources and access to large-scale information.

Figure shows pseudo code for the “Language Translation using APIs” model:


```

# Import required Libraries
Import Streamlit for the app interface
Import Google Translate (Translator) for text translation
Import Google Text-to-Speech (gTTS) for converting text to audio
Import BytesIO for managing in-memory data
Import tempfile for temporary file management
Import 'languages' dictionary with supported language codes

# Create Streamlit title
Display Streamlit title "Language Translation App"

# Text area for user input
Create text area for user to enter text to be translated

# Dropdown to select the target language for translation
Create select box with language options from 'languages'

# Button to initiate translation
Create a button "Translate"

# If the "Translate" button is clicked
If "Translate" button is pressed:
    - Create a 'Translator' instance
    - Translate the text from the source language to the selected target language
    - Store the translated text in 'out'
    - Display the translated text in Streamlit

# Generate audio from the translated text
- Create a 'gTTS' instance with the translated text and the target language code
- Save the generated audio to a temporary file
- Read the audio data into memory
- Convert the audio data to base64 for Streamlit playback

# Button to play the audio translation
Create a button "🔊 Click to listen to the translation"

# If the audio play button is clicked
If the play button is pressed:
    - Play the audio data in Streamlit using 'st.audio'

```

Fig24. Pseudo code for the Language Translation using APIs model

Below shown figure is text-to-text language translation output using google trans API:

```

Input Source Language: en
Input the Scentence you want to Translate: Translating text from english to hindi.
Input Destined Language: hi
The Translated Scentence: अंग्रेजी से हिंदी में अनुवाद करना।

```

Fig25. Text-to-text translation output using google trans API

Text translation functionality in our system allows users to translate either single sentences or lists of sentences to their desired target language. This functionality is implemented in the `translate_text` function within the `Main.py` script.

- The `translate_text` function first validates the input to ensure it is either a single sentence or a list of sentences. This is crucial for determining the appropriate translation process. If the input is invalid, a `ValueError` is raised to alert the user.
- Once the input is validated, the translation process begins. For each sentence in the input, the function utilizes the Google Translate API to translate the sentence to the specified target language. The translated sentences are then appended to a list, which is returned as the final output.
- The function is designed to handle different data types gracefully. If the input is a single string, it performs translation on that single sentence. If the input is a list of strings, it performs translation on each sentence in the list individually.
- Error handling mechanisms are implemented to handle exceptions that may occur during the translation process. This ensures that the system remains robust and resilient, providing a smooth user experience even in the face of unexpected errors.

Below shown image is the “Text Language Translation” we made in the application deployed:

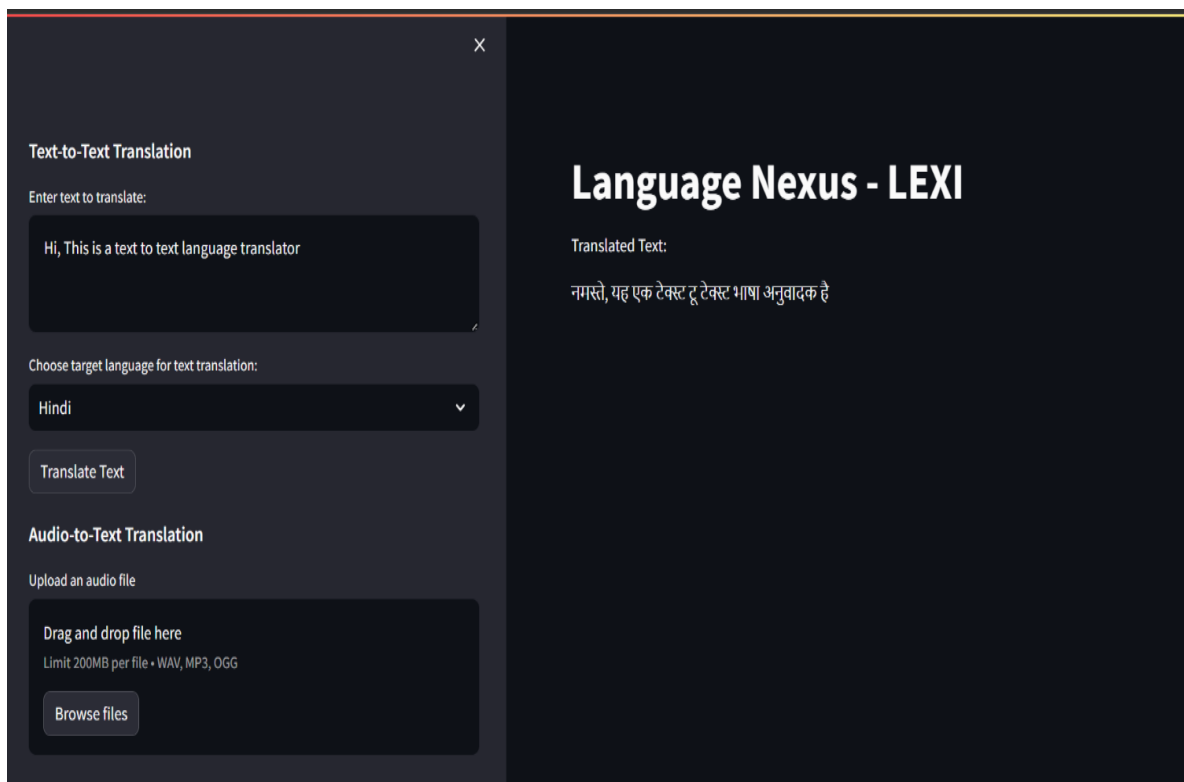


Fig26. The Text Language Translation we made in the application deployed

3.3 Audio-to-Text Translation:

Automatic speech Recognition, or ASR, is the application of Machine Learning or Artificial Intelligence (AI) technology to convert human voice into legible text.[21] Let's first understand the components of the model:

- **Audio Input:** It always begins with the data. We will take the data that is voice of the user or the voice document provided by the user as the audio input for the further process.

- **Feature Extraction:** Feature extraction is a machine learning and data analysis technique that identifies and extracts significant characteristics from raw data. These traits are eventually utilised to generate a more informative dataset. In our model we have taken the analysis of the raw audio provided by the user to identify the features such as frequency of the voice, pitch of the voice and intensity of the voice. These will help in understand the voice much clear and accurate.
- **Acoustic Model:** The acoustic model typically works with raw audio waveforms from human speech, predicting which phoneme each waveform belongs to, usually at the character or sub-word level. Phoneme are any of the perceptually different sound units in a given language that separate one word from another, such as p, b, d, and t in the English words pad, pat, bad, and bat, etc.
- **Language Model:** Language models are useful for a variety of jobs, including speech recognition, machine translation, natural language generation, optical character recognition, handwriting recognition, grammar induction, etc. With the help of language models, we can recognize each word from the given text or audio or image -text etc. It helps us in providing context by predicting word sequences based on grammar, vocabulary, and usage patterns.
- **Decoder:** The decoder component in voice recognition systems is crucial for transforming audio input into text transcription by combining acoustic and linguistic models. This integration includes a series of important processes for correctly decoding spoken language:
 - **Acoustic Model Integration:** The decoder initially appoints an acoustic model, which is trained to map acoustic information (e.g., spectrograms) recovered from audio input to phonetic units or language representations. The acoustic model aids in identifying and matching speech segments with their appropriate phonetic units or sub-word representations.
 - **Integration of Language Models:** After receiving audio representations, the decoder uses a language model to forecast and construct credible sequences of words or linguistic units based on the recognised phonetic or sub-word representations. The language model increases transcription accuracy is improved by making use of linguistic context, vocabulary, and grammatical restrictions.
 - **Beam Search/Decoding Strategy:** The decoder regularly uses beam search or other decoding algorithms to provide the best likely transcription based on the audio input. Beam search investigates various stand by word sequences based on acoustic and language model scores, with the goal of determining which sequence has the highest overall likelihood.
 - **Lexicon and Pronunciation Knowledge:** The decoder may use a lexicon or pronunciation knowledge to handle out-of-vocabulary terms or abnormal pronunciations in the audio input. This aids in correctly mapping phonetic representations to appropriate words or linguistic units.
 - **Integration and Output:** Finally, the decoder combines the acoustic and language models' outputs to provide the final text transcription of the audio input. This

transcription offers the system's best assessment of the spoken material based on previously learnt audio patterns and linguistic context.

The flowchart below explains about the working of Automatic speech Recognition or ASR model:

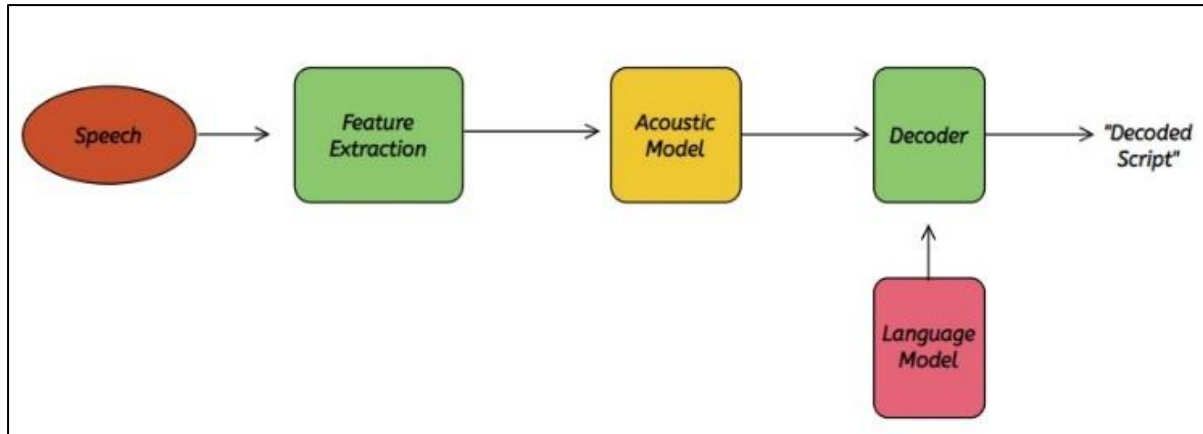


Fig27. Flowchart of ASR working

The Audio-to-Text translation functionality in our system allows users to upload audio files in various formats such as WAV, MP3, or OGG, and convert the spoken content into text format. This functionality is implemented in the Main.py script.

The model we have created has two types

- Live Translate (Real time Voice Translation)
- Audio document Voice Translation

Live Translate (Real Time Voice Translation)

The Live Translate technique, which focuses on real-time voice translation, includes many critical components that allow spoken words to be effortlessly translated into another language.

- **Speech Recognition:** Speech recognition captures live audio input, such as spoken language or speech, utilising microphones or recording devices.

Automatic Speech Recognition (ASR) technology is used to transcribe spoken audio into text. ASR systems use acoustic models to understand sound patterns and language models to convert phoneme sequences into words and sentences.

Handling different accents, background noise, and context-dependent speech changes are all significant issues in speech recognition. Advanced ASR systems use deep learning models such as recurrent neural networks (RNNs) or transformer-based architectures to increase accuracy.

- **Audio Transcription:** Later speech recognition, the recognised audio speech is converted into text in the language user has spoken. This text helps in further process.
- **Language Translation:** After audio transcription text is machine translated into another language.

Machine translation (MT) systems, to create correct translations, use advanced algorithms and neural network designs that have been trained on large multilingual corpora. These systems use contextual information, grammar, and syntax to provide meaningful translations.

State-of-the-art MT models use transformer-based designs such as BERT or T5, which allow them to handle complicated linguistic patterns and subtleties across several languages.

- **Text-to-Speech(TTS) Synthesis:** After acquiring the translated text, the next step is to turn the written output back into spoken language using Text-to-Speech (TTS) synthesis.
TTS systems use techniques like concatenative synthesis (combining pre-recorded speech units) and parametric synthesis (creating speech from acoustic characteristics) to produce synthetic speech that sounds realistic and human-like.
Neural network-based TTS models, such as WaveNet or Tacotron, have dramatically increased the quality and expressiveness of synthesised speech, creating it is better suited for real-time speech translation applications.

- **Integration and Real-Time Operation:** Live Translate systems incorporate these components into a pipeline that functions in real-time, allowing for smooth and immediate translation of spoken language.
The pipeline consists of continuous audio input processing, quick transcription, text translation, and instantaneous speech synthesis.
Challenges such as latency, accuracy, and naturalness of synthesised speech are addressed by optimising algorithms, accelerating hardware, and making ongoing advances in machine learning and voice technology.

- **Applications and Implications:** Live Translate technologies are useful for multilingual communication, cross-cultural relationships, travel, and accessibility for those with language problems.
They enable users to interact efficiently across languages in real time, bringing up new potential for worldwide cooperation and international trade and inclusive communication

```

Define 'languages' dictionary with language names and corresponding language codes

# Function to record audio from the microphone
Define 'take_command' function:
    - Initialize a speech recognizer
    - Record audio using the system's microphone
    - Attempt to transcribe the audio to text using Google's speech recognition
    - Return the transcribed text
    - Handle exceptions for any errors in transcription

# Function to translate a given text to a target language
Define 'translate_text' function:
    - Initialize the Google Translate API
    - Translate the input text into the target language
    - Return the translated text

# Streamlit application starts
Start Streamlit application with title "Voice-to-Voice Translation App"

# Sidebar for choosing translation language
Define 'lang_options' with the list of supported languages
Allow user to select a target language from 'lang_options' using a select box
If user clicks "Record and Translate" button:
    - Call 'take_command' to record and transcribe speech
    - If transcription is successful:
        - Display the transcribed text as "You said: {transcribed_text}"
        - Get the target language code for the chosen language from 'languages'
        - Translate the transcribed text to the chosen language
        - Use gTTS to convert the translated text to audio
        - Save the audio to a temporary file ("translated_audio.mp3")
        - Play the translated audio
        - Delete the temporary audio file after playback
        - Display the translated text as "Translated text: {translated_text}"
    - Else (if transcription fails):
        - Display a message indicating transcription failure

```

Fig28. Pseudo code for the Live Translate model

The next image is the “Live Translate (Real Time Voice Translation)” we made in the application deployed:

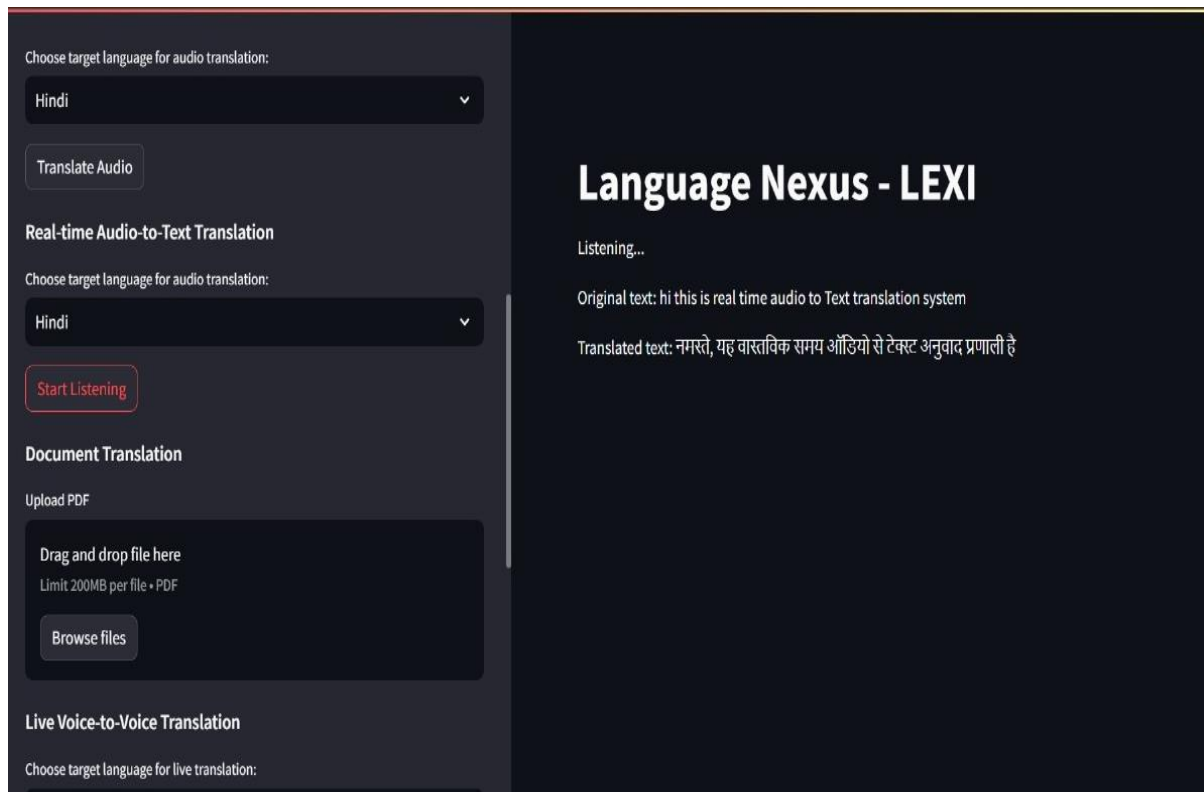


Fig29. Live Translate we made in the application deployed

Audio Translation

Audio Translation is a useful tool for translating audio recordings or documents from one language into another. This technology makes it easier to convert spoken material into written text, which can then be translated into a target language, allowing for more effective communication and content distribution across varied linguistic audiences. Here is how Audio Document Voice Translation usually works:

- **Audio Input Processing:** - The procedure starts with receiving audio recordings or papers with spoken material in a source language. These recordings may comprise interviews, lectures, talks, podcasts, or other forms of oral communication.
- **Speech-to-Text Transcription:** The audio input is analysed using Automatic Speech Recognition (ASR) technology to turn spoken material into text. ASR systems analyse audio data to recognise and transcribe spoken words into text.

The transcription process entails segmenting the audio input, detecting spoken words, and creating a written transcript that properly represents the spoken information.

- **Text Translation:** After transcribing oral material into text, the next stage is language translation. The recorded text is translated using a machine translation system, such as Google Translate or bespoke neural machine translation models, into the target language. Machine translation algorithms analyse textual input, taking into account

linguistic patterns, semantics, and context, to generate intelligible and coherent translations.

- **Generating Translated Audio:** After translating the text, the final output may be converted back into audio using Text-to-Speech (TTS) technology. TTS systems turn written text into spoken voice, enabling the translated material to be spoken in the target language. The synthesised audio keeps the target language's tone, rhythm, and pronunciation qualities, resulting in a natural and fluent portrayal of the translated information.
- **Applications and Use Cases:** Audio Document Voice Translation is useful for multilingual communication and content localization.
 - Media transcription is the process of translating spoken information from interviews, podcasts, or videos into different languages in order to reach a larger audience.
 - Instructional materials: Translating instructional information, training films, and e-learning tools into several languages to help global learners.
 - Content localization is the process of translating marketing materials, announcements, or business communications into many languages in order to reach a worldwide audience.

Figure shows pseudo code for the “Audio Translation” model:

```
# Define the list of supported languages and their codes
Define 'languages' dictionary with language names and corresponding language codes

# Initialize the speech recognizer
Initialize recognizer using the 'speech_recognition' module

# Initialize the Google Translate API client
Initialize translator using the 'googletrans' module

# Start the Streamlit application
Start Streamlit application with title "Audio-to-Text Translation App"

# File upload widget for audio files
Allow user to upload an audio file of type WAV, MP3, or OGG

# Dropdown to select the target translation language
Allow user to select a language from the 'languages' list
```

Fig30. Pseudo code for the Audio Translation model (1)


```

# If the user has uploaded a file:
If uploaded_file exists:
    # Save the uploaded file to a temporary location
    Write the uploaded_file data to a temporary file
    Store the temporary file path in 'temp_audio_path'

    # Load the audio file into the speech recognizer
    Load the audio file from 'temp_audio_path' using sr.AudioFile
    Record the audio data

    Try:
        # Recognize the speech in the audio data using Google's speech-to-text
        Recognize the speech from the audio data
        Store the recognized text in 'original_text'

    # Display the original recognized text
    Output "Original text" followed by the 'original_text'

    # Translate the recognized text into the selected language
    Retrieve the target language code from 'languages'
    Translate the 'original_text' into the target language
    Store the translated text in 'translated_text'

    # Display the translated text
    Output "Translated text" followed by 'translated_text'

    # Convert the translated text to speech (TTS) in the target language
    Use gTTS to convert 'translated_text' to speech
    Save the TTS output to a temporary audio file

    # Display an option to play the translated audio
    Output a message with a speaker icon indicating audio playback option

    # If the user presses the play button:
    If play button is pressed:
        # Play the TTS audio file
        Play the temporary audio file using 'playsound'

# Handle errors during speech recognition
Catch 'sr.UnknownValueError':
    Output "Could not understand the audio"

Catch 'sr.RequestError':
    Output "Could not request results; check your network connection"

# Clean up temporary files
Delete 'temp_audio_path'
Delete the temporary TTS audio file

```

Fig31. Pseudo code for the Audio Translation model (2)

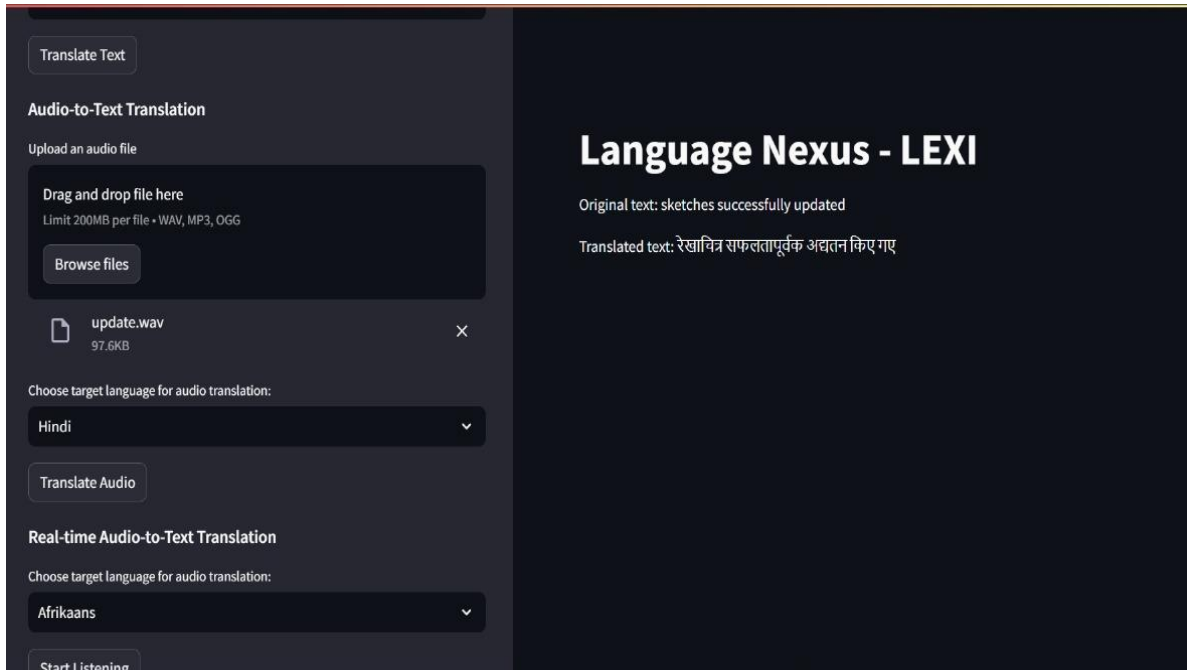


Fig32. Working of the Audio Translation model

Both sorts of translation features use voice processing, natural language processing (NLP), and machine translation techniques to allow for smooth communication and comprehension across several languages. The model's adaptability in dealing with both real-time voice interactions and audio document translations highlights its application in varied use scenarios, ranging from multilingual communication apps to automatic transcription and translation of audio material.

3.4 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a crucial component of our language translation system, enabling the extraction of text from images and live video streams.[22] The OCR functionality was implemented using the EasyOCR library in Python, which provides robust text recognition capabilities.

In the deployed application we have made three models from the usage of Optical Character Recognition modules and libraries, those are:

- Live Capture Language Translation.
- Image document to Text Language Translation.
- Document Language Translation

Live Capture Language Translation:

This approach allows for real-time translation of text recorded by live camera input.

- The programme uses OCR technology to extract text from live video feeds or camera input in real time.
- The retrieved text is next analysed for language detection to determine the source language.

- The detected text is then translated into the intended target language using machine translation techniques.
- Live Capture Language Translation is suitable for quick translation of printed text, signs, or documents observed through a camera, allowing for on-the-spot language comprehension and conversation.

Figure shows pseudo code for the “Live Capture Language Translation” model:

```
# Import necessary libraries
Import OpenCV (cv2)
Import EasyOCR for Optical Character Recognition (OCR)
Import NumPy
Import Streamlit for web interface
Import TextBlob for language detection
Import Google Translate for text translation
Import LANGUAGES from Google Translate for supported languages

# Initialize EasyOCR reader for English
Create 'reader' object for EasyOCR with 'en' as the language

# Initialize Google Translate
Create 'translator' object

# Function to translate a given text into a specified target language
Define 'translate_text(text, target_lang)':
    - Translate the text using Google Translate
    - Return the translated text

# Function to add semi-transparent text onto an image
Define 'add_transparent_text(image, text, position, detected_lang)':
    - Create a copy of the image
    - Determine text size and position
    - Define the position of the semi-transparent background rectangle
    - Draw a semi-transparent grey rectangle as a background for the text
    - Add the translated text onto the image
    - Display the detected language on the top-left corner of the image
    - Return the modified image with the text overlay

# Function to capture video from a webcam and process it for OCR translation
Define 'capture_and_process(target_lang)':
    - Create a Streamlit title "Live OCR Translation"
    - Open a video capture using OpenCV (default webcam)
```

Fig33. Pseudo code for the Live Capture Translation model(1)

```

Loop while video is capturing:
    - Capture a frame from the webcam
    - If no frame is captured, break the loop
    - Use EasyOCR to detect text in the frame
    - Create a copy of the frame for processing
    - For each detected text in the frame:
        - Get the bounding box of the text
        - Extract the detected text
        - Try to detect the language using TextBlob
        - If language detection fails, default to 'en'
        - If the detected language is different from 'target_lang':
            - Translate the text to the target language
        - Else:
            - Use the original text
        - Add the translated text onto the processed frame using 'add_transparent_text'
    - Display the processed frame on Streamlit
    - Check for key press (if 'q' is pressed, exit the loop)

- Release the video capture

# Function to start the live OCR translation process with the specified target language
Define 'start_translation(target_lang)':
    - Call 'capture_and_process(target_lang)'

# Main section of the code
If '__name__' equals '__main__':
    - Retrieve the supported languages for translation from Google Translate
    - Create a Streamlit sidebar to select the target language
    - Call 'start_translation(selected_language)' with the selected language

```

Fig34. pseudo code for the Live Capture Translation model(2)

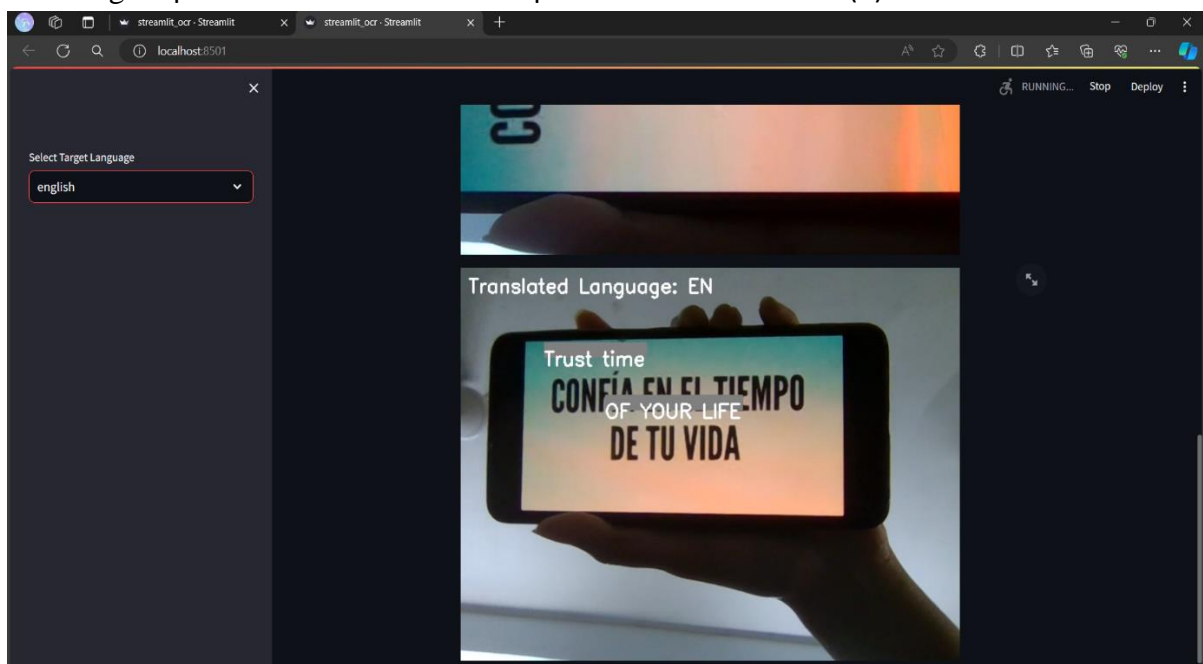


Fig35. Live Capture Language Translation model working

This image shows, after showing a text to the camera, it captures the image in frames one at a time extracts the text and converts into the user desired language on real time without costing us any memory. Here the text is recognised given by user and converted it into English language as user requirement.

Image Document to Text Language Translation

This approach allows for system to extract text from an image and then translate it.

- **Image Preprocessing:**
 - Capture Image: The OCR process begins with capturing an image containing text.[23]
 - Preprocessing: Preprocess the image to enhance text visibility (e.g., resizing, adjusting brightness/contrast).
- **Text Detection:** Detect text regions within the preprocessed image using techniques like edge detection and contour analysis.
- **Text Recognition:** Extract text from the detected regions using Optical Character Recognition (OCR) algorithms.
- **Language Detection:** Determine the language of the extracted text to guide translation.
- **Translation:** Translate the extracted text if it's in a different language from the target language.

Below shown figure is the working of the “Image Document to Text Language Translation” model:

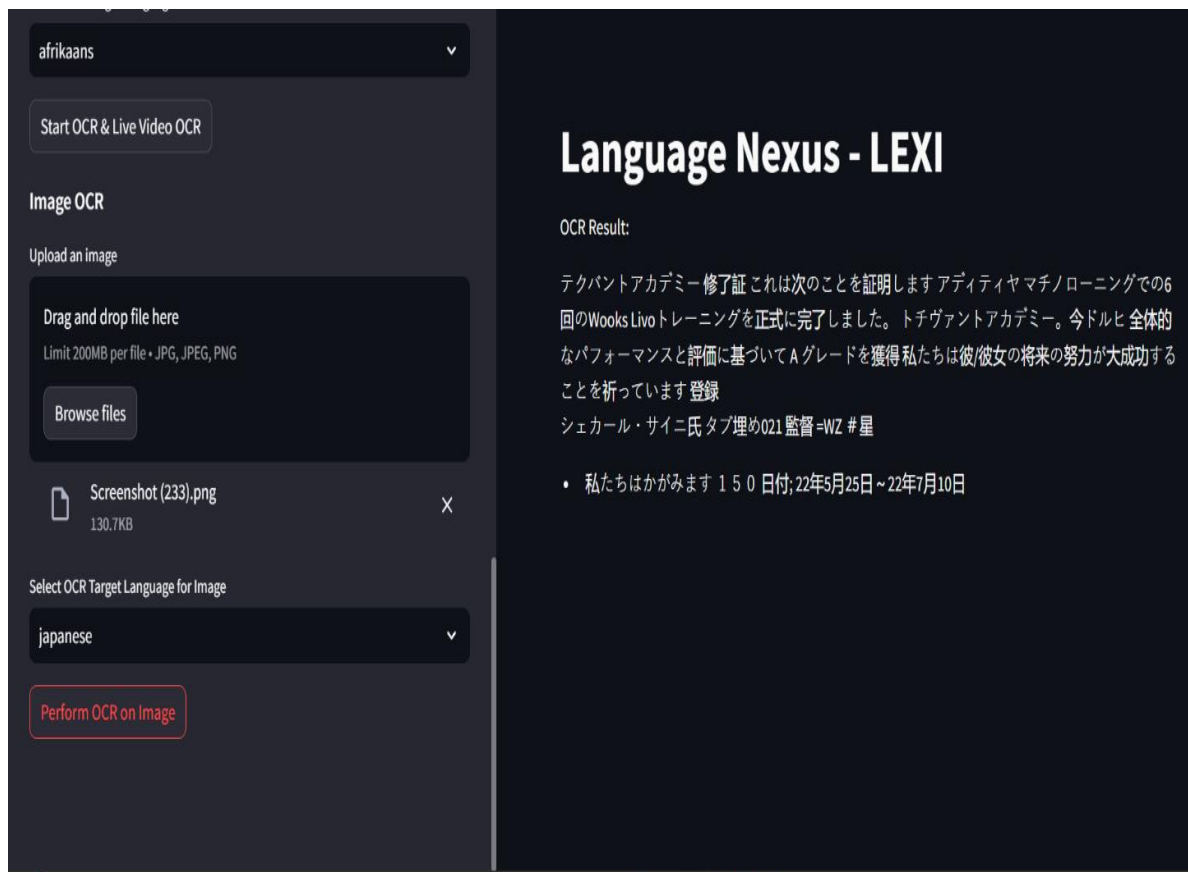


Fig36. Working of the Image Document to Text Translation model

Document **Language Translation:** This approach is designed to translate text from electronic or scanned documents.[24] Users can upload text-containing electronic documents (PDFs, etc.).

OCR technology extracts text from uploaded documents or scanned images. Similar to the previous models, the extracted text is subjected to language identification and translation to translate the material into the appropriate target language.

Document Language Translation is appropriate for large-scale textual document translation, allowing for effective cross-language communication and digital file content localization.[25]

```
# Import necessary libraries
import Streamlit
import PdfReader from PyPDF2
import Translator from googletrans
import tempfile for temporary file management

# Function to translate PDF content to a specified language
Define 'translate_pdf(pdf_reader, target_language)':
    - Initialize the Google Translate API
    - Create an empty list 'translated_pages'

    # Iterate over all pages in the PDF
    For each page in 'pdf_reader.pages':
        - Extract text from the current page

        # If text is found on the page
        If text exists:
            - Translate the extracted text to the target language
            - Append the translated text to 'translated_pages'

        # If no text is found
        Else:
            - Append "No text found on this page." to 'translated_pages'

    - Return 'translated_pages'

# Streamlit application starts
Set Streamlit title to "PDF Translator"
Display a message indicating users can upload a PDF to translate it

# File uploader to upload PDF files
Create a file uploader for PDF files

# Dropdown to select the target language for translation
Define 'language_options' dictionary with language names and corresponding language codes
```

Fig37. Pseudo code for the Image file to Text Translation model(1)


```

Create a dropdown menu to select the translation language

# If a PDF file is uploaded
If 'uploaded_file' exists:
    - Create a temporary file to save the uploaded PDF content
    - Write the uploaded file content to the temporary file
    - Store the temporary file path

    # Create a PDF reader using the temporary file
    Create 'pdf_reader' using PdfReader and the temporary file path

    # Translate the content of the PDF to the selected language
    Call 'translate_pdf(pdf_reader, language_options[selected_language])'
    Store the result in 'translated_content'

    # Display the translated content on Streamlit
    Output a message "Translated content into {selected_language}:"
    For each 'content' in 'translated_content':
        - Output the content with a corresponding page number

    # Clean up by removing the temporary file
    Delete the temporary file

Else:
    # If no PDF is uploaded
    Output a message indicating that a PDF must be uploaded for translation

```

Fig38. Pseudo code for the Image file to Text Translation model(2)

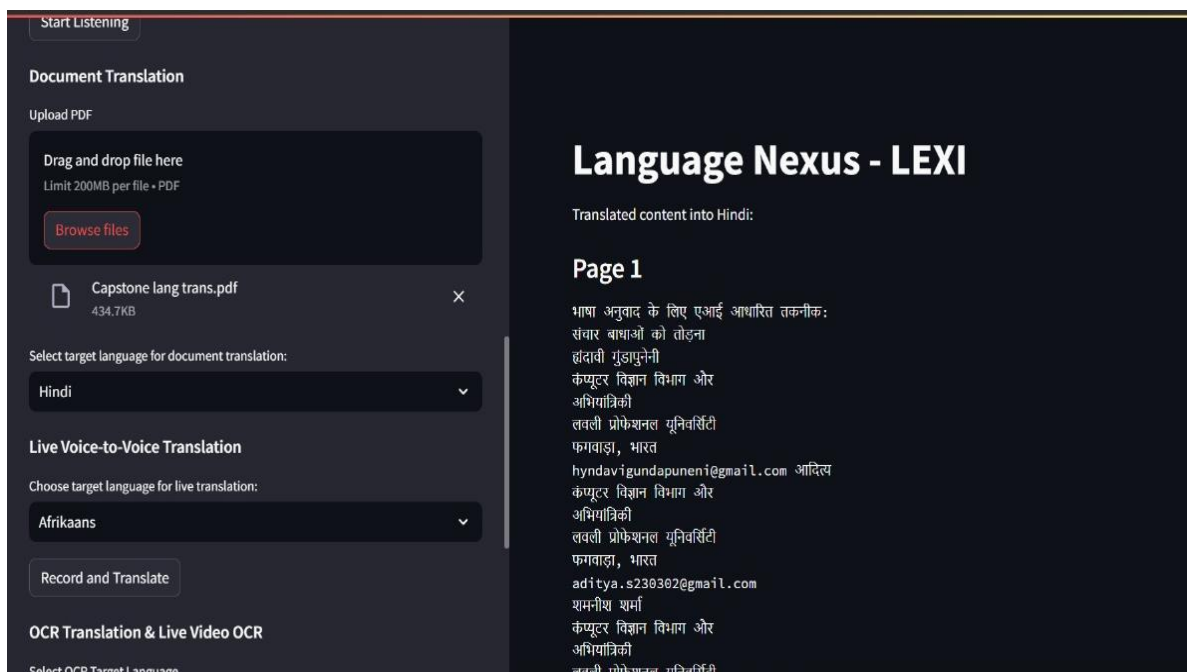


Fig39. Working of document language translation

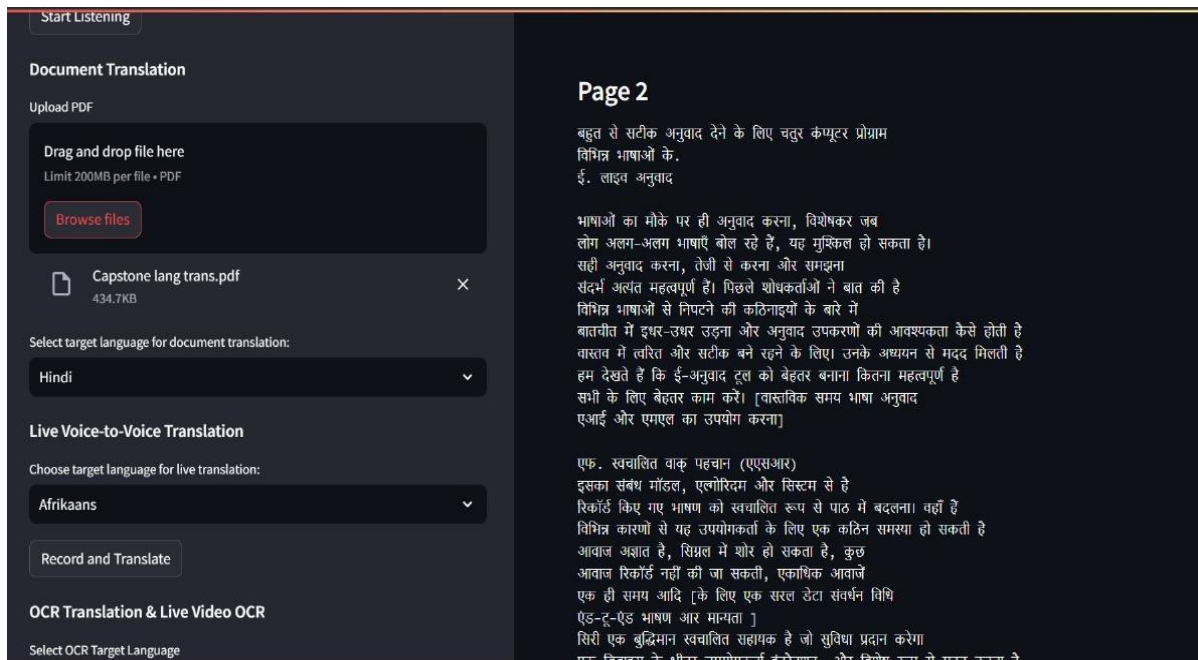


Fig40. Pages division in document translation

3.5 Integration and User Interface Development:

- **Integration of Translation Functionality:** Integration involved incorporating the various translation functionalities into a cohesive system. This was achieved by organizing the codebase into modular components and ensuring seamless communication between them.
- **User Interface Development:** The user interface was developed using Streamlit, providing an intuitive and interactive platform for users to interact with the translation system. Various input widgets and visualizations were utilized to enhance user experience and facilitate seamless translation.

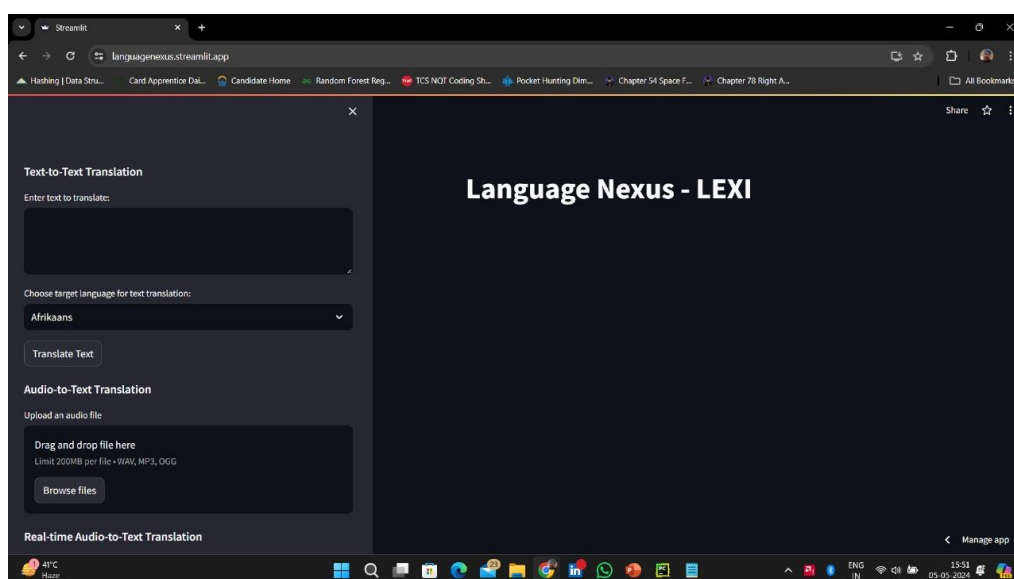


Fig41. Model after deployment(1)

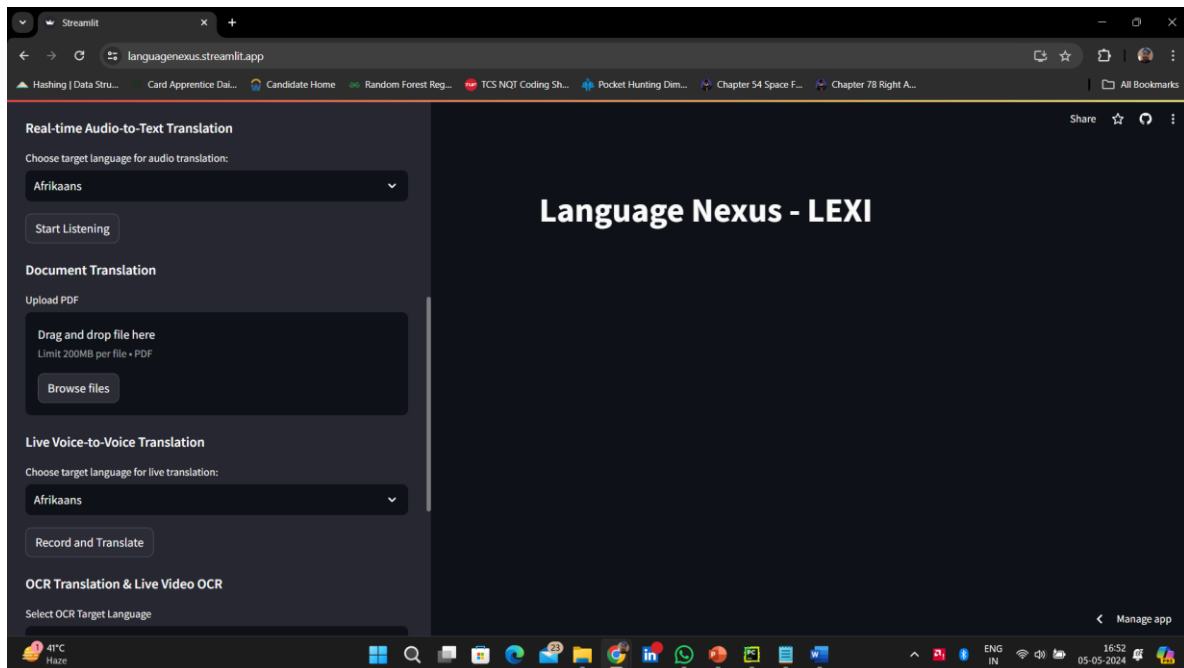


Fig42. Model after deployment(2)

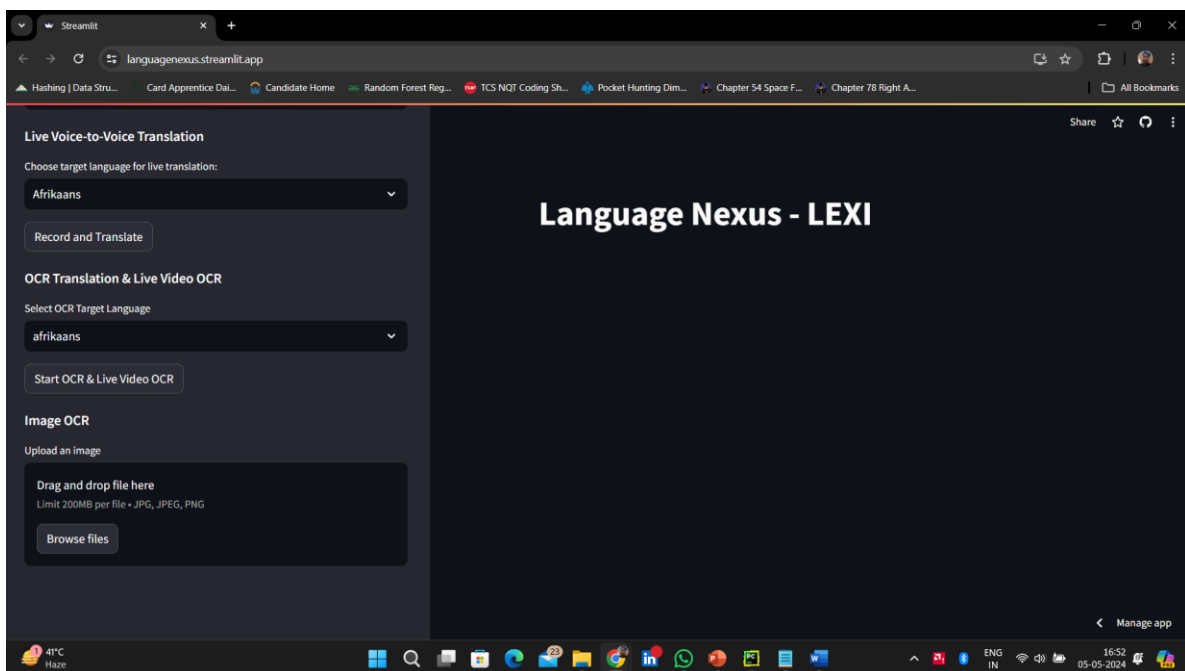


Fig43. Model after deployment(3)

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

In conclusion, this study delved into the realm of language translation technologies, aiming to address the challenges posed by the language barrier in today's interconnected world. With a growing interest in diverse cultures and literature across borders, efficient language translation systems have become indispensable.

Through the utilization of advanced technologies and methodologies, such as Google Cloud Translation, Multilingual T5 models, and APIs provided by leading cloud service providers like Google and Microsoft, this research endeavoured to create an accurate, scalable, and versatile language translation system.

The methodology employed in this study encompassed comprehensive data collection and preparation, leveraging a multi-faceted approach to ensure extensive language coverage and robust translation capabilities. Techniques like language translation using Transformers and APIs, audio-to-text translation, document translation, real-time translation, and optical character recognition were intricately integrated into the system.

The testing and evaluation phase yielded promising results, showcasing a high degree of accuracy with a test accuracy of 99.5%. Despite encountering some erroneous predictions, particularly in language pairs like Urdu and Hindi, and Russian and Bulgarian, the overall performance of the model remained commendable.

The progress of language translation technology is swiftly advancing because of advancements in the field of artificial intelligence, machine learning, and natural language processing. Potential advancements encompass improved precision and fluency, translation tailored to specific domains, translation that incorporates several modes of communication, translation that adapts to individual needs, and solutions to the difficulties faced by languages with limited resources and representation. Present challenges encompass attaining translation accuracy comparable to that of humans, incorporating various modes such as text, speech, visuals, and gestures, and creating adaptable systems that respond to individual linguistic requirements and preferences. Subsequent investigations should prioritize the resolution of these obstacles to better the performance of translation systems and promote more accessibility and inclusivity.

In essence, this study underscores the significance of leveraging modern technologies to overcome linguistic barriers, facilitating seamless communication and collaboration across diverse linguistic landscapes. By harnessing the power of language translation systems, individuals and organizations can bridge cultural divides, access global content, and foster meaningful interactions in an increasingly interconnected world.

REFERENCES

- [1] G. Lample and A. Conneau, "Cross-lingual language model pretraining," arXiv Prepr. arXiv1901.07291, 2019.
- [2] P. Singh, S. Verma, I. Khan, and S. Sharma, "Machine Learning: A Comprehensive Survey on Existing Algorithms," J. Comput. Sci. Eng. Softw. Test., vol. 7, no. 3, pp. 1–9, 2021.
- [3] M.-T. Luong and C. D. Manning, "Stanford neural machine translation systems for spoken language domains," in Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign, 2015, pp. 76–79.
- [4] P. A. Khan, S. Sharma, I. A. Khan, and P. Singh, "Image Detection based technique for shutting down the Applications opened in Windows Operating Environment," Contemp. Issues Soc. Sci., p. 290.
- [5] S. A. Kamaraj, M. Gautham, S. Karthikeyan, and R. Parkavi, "Enhancing Automatic Speech Recognition and Speech Translation Using Google Translate," in Handbook of Research on Data Science and Cybersecurity Innovations in Industry 4.0 Technologies, IGI Global, 2023, pp. 220–241.
- [6] S. Singh, C. Sharma, S. Sharma, and N. K. Verma, "Re-Learning Emotional Intelligence Through Artificial Intelligence," in 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2021, pp. 1–5.
- [7] P. S. V. Pamudhurthy, R. Komali, A. B. Panyam, A. Katarukonda, and H. Raithatha, "Real Time Language Translation Using AI and ML," 2024.
- [8] X. Song, Z. Wu, Y. Huang, D. Su, and H. Meng, "SpecSwap: A Simple Data Augmentation Method for End-to-End Speech Recognition.," in Interspeech, 2020, pp. 581–585.
- [9] A. Kalyani and P. S. Sajja, "A review of machine translation systems in india and different translation evaluation methodologies," Int. J. Comput. Appl., vol. 121, no. 23, 2015.
- [10] S. K. Saksamudre, P. P. Shrishrimal, and R. R. Deshmukh, "A review on different approaches for speech recognition system," Int. J. Comput. Appl., vol. 115, no. 22, 2015.
- [11] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 12, no. 8, pp. 787–808, 1990.
- [12] M. A. Radwan, M. I. Khalil, and H. M. Abbas, "Neural networks pipeline for offline machine printed Arabic OCR," Neural Process. Lett., vol. 48, no. 2, pp. 769–787, 2018.
- [13] M. Kumar, C. Sharma, S. Sharma, N. Nidhi, and N. Islam, "Analysis of Feature Selection and Data Mining Techniques to Predict Student Academic Performance," in 2022 International Conference on Decision Aid Sciences and Applications (DASA), 2022, pp. 1013–1017.

- [14] A. Sharma, Tushar; Sharma, Shamneesh; Sharma, Ajay; Kumar, Aman; Malik, Arun, Sharma, “Asteroid Hazard Prediction Using Machine Learning: A Comparative Analysis of Different Algorithms,” in 2023 3rd International Conference on Advancement in Electronics & Communication Engineering (AECE), IEEE, 2024, pp. 673–677. doi: 10.1109/AECE59614.2023.10428633.
- [15] S. R. Satti, J. S. K. Lankadasu, A. Sharma, S. Sharma, and S. Gochhait, “Deep Learning in Medical Image Diagnosis for COVID-19,” in 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETISIS), 2024, pp. 1858–1865.
- [16] Ali, P.J.M., Faraj, R.H., Koya, E., Ali, P.J.M. and Faraj, R.H., 2014. Data normalization and standardization: a technical report. Mach Learn Tech Rep, 1(1), pp.1-6.
- [17] Naselaris, T., Kay, K.N., Nishimoto, S. and Gallant, J.L., 2011. Encoding and decoding in fMRI. *Neuroimage*, 56(2), pp.400-410.
- [18] Chi, Z., Dong, L., Ma, S., Mao, S.H.X.L., Huang, H. and Wei, F., 2021. mT6: Multilingual pretrained text-to-text transformer with translation pairs. arXiv preprint arXiv:2104.08692.
- [19] Zhang, J., Luan, H., Sun, M., Zhai, F., Xu, J., Zhang, M. and Liu, Y., 2018. Improving the transformer translation model with document-level context. arXiv preprint arXiv:1810.03581.
- [20] Nguyen, A.T., Rigby, P.C., Nguyen, T., Palani, D., Karanfil, M. and Nguyen, T.N., 2018, September. Statistical translation of English texts to API code templates. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 194-205). IEEE.
- [21] Gaikwad, S.K., Gawali, B.W. and Yannawar, P., 2010. A review on speech recognition technique. *International Journal of Computer Applications*, 10(3), pp.16-24.
- [22] Mori, S., Suen, C.Y. and Yamamoto, K., 1992. Historical review of OCR research and development. *Proceedings of the IEEE*, 80(7), pp.1029-1058.
- [23] Singh, A., Bacchuwar, K. and Bhasin, A., 2012. A survey of OCR applications. *International Journal of Machine Learning and Computing*, 2(3), p.314.
- [24] Oard, D.W., 1998, October. A comparative study of query and document translation for cross-language information retrieval. In *Conference of the Association for Machine Translation in the Americas* (pp. 472-483). Berlin, Heidelberg: Springer Berlin Heidelberg.
- [25] Donato, D., Yu, L. and Dyer, C., 2021. Diverse pretrained context encodings improve document translation. arXiv preprint arXiv:2106.03717.