NLP Driven Instant Language Translation

Dr M Shilpa[1], Tulasi Udupa[2], Sushma Jayaram[3], Navya[4], Navya G[5]

[1]Associate Professor, Bangalore Institute of Technology
[2,3,4,5]Student, Bangalore Institute of Technology

Abstract
The project aims to develop a multilingual chat application that supports real-time translation of text. The application enables communication across diverse languages and dialects by integrating powerful translation models. This ensures high accuracy in translation, which provides a seamless experience. The application includes a user-friendly interface that offers an attractive environment for chatting with people from diverse communities. It allows for audio-to-text translation for users who prefer speaking over typing. It also supports dynamic media playback, providing subtitles to the video sent to them. It can also generate translated text from an image, providing users with a reliable tool for multilingual communication. This versatile and inclusive solution helps break down language barriers, bringing people closer together and building a more connected global community.

Keywords: Natural Language Processing, Real-time Language Translation, Subtitle Generation, Machine Translation, Automatic Speech Recognition, Neural Network, Multimodal Communication

I. INTRODUCTION
Language barriers often make real-time communication between people from different cultures and backgrounds a challenge. Real-time language translation systems have become a game-changer, using advanced machine learning and natural language processing (NLP) to deliver instant translations. This project aims to break those barriers by creating a chat application that provides seamless multilingual communication.
The project focuses on the development of a chat application with real-time translation. The system uses state-of-the-art Transformer models, known for their speed and accuracy, to handle multiple languages and dialects. It can translate text instantly, convert audio messages into text, turn text into spoken audio, and even add subtitles to videos. These features allow users to communicate easily, whether they're typing, speaking, or sharing media. The project also has a user-friendly interface that makes the translation feature effortless to use, ensuring it works smoothly with minimal effort from the user.

A. Computational Requirements
The application is designed to be efficient and accessible. Transformer-based models ensure fast and accurate translations while maintaining low latency. The system is lightweight enough to run smoothly on devices with limited resources, making it ideal for real-time use.

B. Use Case Scenarios
The multilingual chat application has the potential to benefit a wide range of users across different scenarios. For travellers, the app provides a convenient way to connect with locals by translating text or speech into the local language. It also supports translating signboards, menus, and other visual content through its image-based translation features, ensuring smoother navigation and interaction during trips.

The app also enhances video experiences by generating subtitles, making it easier to understand shared content. For professionals, the app becomes an invaluable tool for multilingual meetings and collaborations and helps global teams communicate smoothly and work together without language barriers. It helps bridge cultural gaps and promote a more connected and inclusive global community.

II. LITERATURE SURVEY

Deep Learning for Machine Translation

M. Chen (2023): This paper introduces a method to estimate the quality of machine translation at the sentence level, utilizing a specialized bidirectional recurrent neural network (Double-RNN). By taking advantage of a large parallel corpus, this Double-RNN model improves the accuracy of quality detection in real-time applications. This approach supports ongoing debugging and optimization of translation systems, and contributes to more accurate and efficient machine translation processes.

Automatic Speech Recognition

J. Liu et al. (2024): This study introduces a computer-aided interpreting (CAI) system called "InterpretSIMPLE" designed for simultaneous interpretation (SI). The system uses automatic speech recognition (ASR) to extract key terms and other relevant details in real-time. The system allows interpreters to upload their commonly used Excel glossaries without any format conversion. It offers features like automatic retrieval and display of terms, numbers, making the interpreting process more efficient and accurate.

NLP in Speech Synthesis

L. Huang et al. (2020): This paper presents a deep learning approach for text normalization in speech synthesis, built on a recurrent neural network (RNN). Unlike traditional rule-based models, this method effectively uses contextual information to overcome their limitations. The model excels at capturing key context and prioritizing words closely related to the target word. Therefore it delivers greater accuracy and enhances the overall text normalization process.

Cross-Lingual Text and Image Recognition

Z. Chen et al. (2021): This paper introduces the concept of Cross-Lingual Text Image Recognition, which involves recognizing text in a source language and translating it directly into a target language without generating intermediate source language text. It employs two sequence-to-sequence learning methods: a convolution-based attention model and a Bidirectional Long Short-Term Memory.

Language Translation and Communication Apps

Hossain et al. (2017): The paper discusses the need for a chat application with an integrated auto-translation system. The proposed system allows users to select their preferred language for chatting and utilizes Google Translator API for real-time translation into different languages.

Attention Is All You Need

Vaswani, Ashish et al. (2017): The paper proposes a new simple network architecture, the Transformer, based solely on attention mechanisms. Traditional models like RNNs, including LSTMs and GRUs, process sequences sequentially, which limits their computational efficiency. The Transformer addresses these limitations by using a method called attention, which lets it focus on all parts of the input and output at once. This results in superior performance and reduced training costs.

Robust Speech Recognition via Large-Scale Weak Supervision

Radford, Alec, et al. (2022): This paper introduces Whisper, a large-scale speech recognition system trained on a massive 680,000 hours of multilingual and multitask audio data. Unlike most traditional models, Whisper doesn't need fine-tuning for specific tasks or datasets and works well in a zero-shot setting. It uses weakly supervised learning, where raw audio is paired with internet-based transcripts. This removes the need for complicated pre-processing or inverse text normalization. The model combines supervised and weakly supervised learning. It outperforms existing models, like Wav2Vec 2.0, which are trained on smaller datasets or extensive fine-tuning. It also delivers high accuracy without the need for human-validated transcripts

III. PROBLEM STATEMENT

The problem statement revolves around the challenges faced in real-time multilingual communication, particularly concerning the accessibility, accuracy, and convenience of translation tools. Current communication platforms often require users to manually translate messages using external services, disrupting the flow of conversation and making real-time communication cumbersome. Additionally, translations often miss the mark in terms of accuracy and context, leaving users with incomplete or confusing interpretations. When it comes to videos, extracting audio and creating subtitles in another language is challenging. It’s a complicated, time-consuming process often requiring separate software and technical expertise. These issues highlight the need for a smarter solution, one that can handle real-time multilingual communication effortlessly, offer accurate and context-aware translations, and make generating subtitles from videos quick and easy for everyone.

IV. PROPOSED SYSTEM AND ARCHITECTURE

The proposed system addresses the need for language translation integrated into a real-time chat application across multiple modalities: text, images, audio, and video. It consists of four major modules: Text Translation, Automatic Speech Recognition, Optical Character Recognition and Translated Subtitle Generation. The modules are then incorporated into a web-based chat interface, as illustrated in Fig. 1.

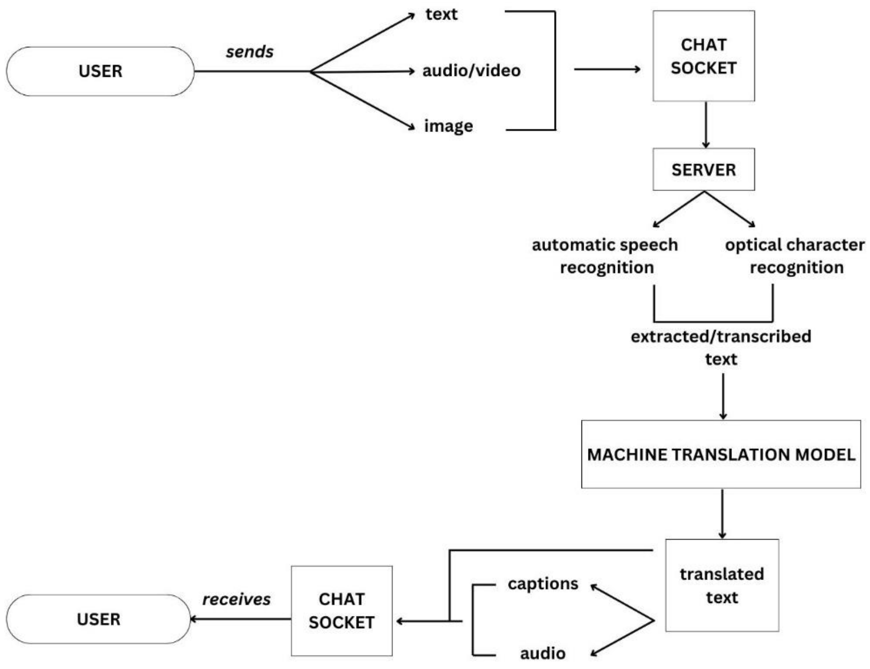

USER
sends
text
audio/video
image
CHAT SOCKET
SERVER
automatic speech recognition
optical character recognition
extracted/transcribed text
MACHINE TRANSLATION MODEL
translated text
captions
audio
CHAT SOCKET
receives
USER

Fig. 1: System Architecture

1. Text Translation

This module is responsible for converting input text from one language to another in real time, maintaining accuracy and preserving context. The system begins with preprocessing, where input text is normalized, tokenized and represented as token IDs. The Transformer model is employed for language translation. It consists of an encoder-decoder architecture enhanced using the self-attention mechanism.
The encoder transforms these token IDs into numerical vectors enriched with semantic information. Positional encodings are incorporated to preserve sequential information, allowing the model to consider the order of words in the input. The encoder comprises multiple layers of self-attention mechanisms and feed-forward neural networks, enabling the model to capture contextual relationships between words across the sequence. The decoder layer generates the translated sentence. It utilises both its previously generated tokens and the encoder's context vectors and employs masking to ensure that the decoder predicts each word sequentially without being affected by future tokens. Post-processing involves detokenizing the output into words and applying any necessary adjustments to ensure grammatical accuracy and fluency in the destination language.

2. Automatic Speech Recognition

The speech recognition module handles conversion of speech to text, which is then sent for translation. This is performed through a Hidden Markov Model. The idea is that a speech signal, when viewed on a time scale as short as ten milliseconds, can be approximated as a stationary process, i.e. its statistical properties do not change.

The audio is pre-processed to reduce noise and retain only speech segments. This is then divided into short frames and features such as cepstral coefficients spectrogram, pitch and formants are extracted. Using the extracted feature vectors, the model computes the likelihood of sequences of phonemes and decodes the most probable sequence, which is then mapped to text, completing the speech-to-text conversion.

3. Optical Character Recognition

The OCR module extracts text from images, which is then translated into the target language. The image is converted to grayscale and pre-processed to remove noise and smoothen the image. Text regions are identified by detecting sharp intensity changes and are organised into bounding boxes or polygons based on their geometric properties.
The model comprises three layers: convolutional neural network, pooling, and fully connected neural network. Convolutional layers identify features such as edges and patterns in the image, while pooling layers reduce the dimension by minimising the number of pixels, retaining only the required features. Since CNN cannot maintain context, the processed data is then passed to a recurrent neural network (RNN). Being suitable for processing sequences of inputs that have variable lengths, it recognises characters by maintaining context across the input.

4. Translated Subtitle Generation

The subtitle generation module focuses on creating an English transcription of the input video and embedding it as subtitles into the video. Audio is extracted from the video and then sent to a Transformer sequence-to-sequence model for speech translation. Similar to the speech recognition module, the audio is divided into smaller frames and the features are extracted. These features are processed by an acoustic model to convert audio signals into phonemes, which are then passed to a model that is pre-trained to generate English text directly from the source speech, without the intermediate transcription stage in the original language. Timestamps are maintained by aligning the audio frames with the output tokens using connectionist temporal classification. Using the text and timestamps, subtitles are generated and overlaid onto the input video.

System Integration

The proposed system leverages WebSocket technology to enable real-time bi-directional communication between the client and server. By establishing a persistent connection, WebSockets eliminate the need for repeated HTTP requests and enable low-latency data transmission. The chat application integrates the translation modules with the WebSocket infrastructure. Messages are sent from the client to the server via WebSockets, where they are processed by the appropriate translation module. The client then receives the translated output via the same connection.

V. IMPLEMENTATION

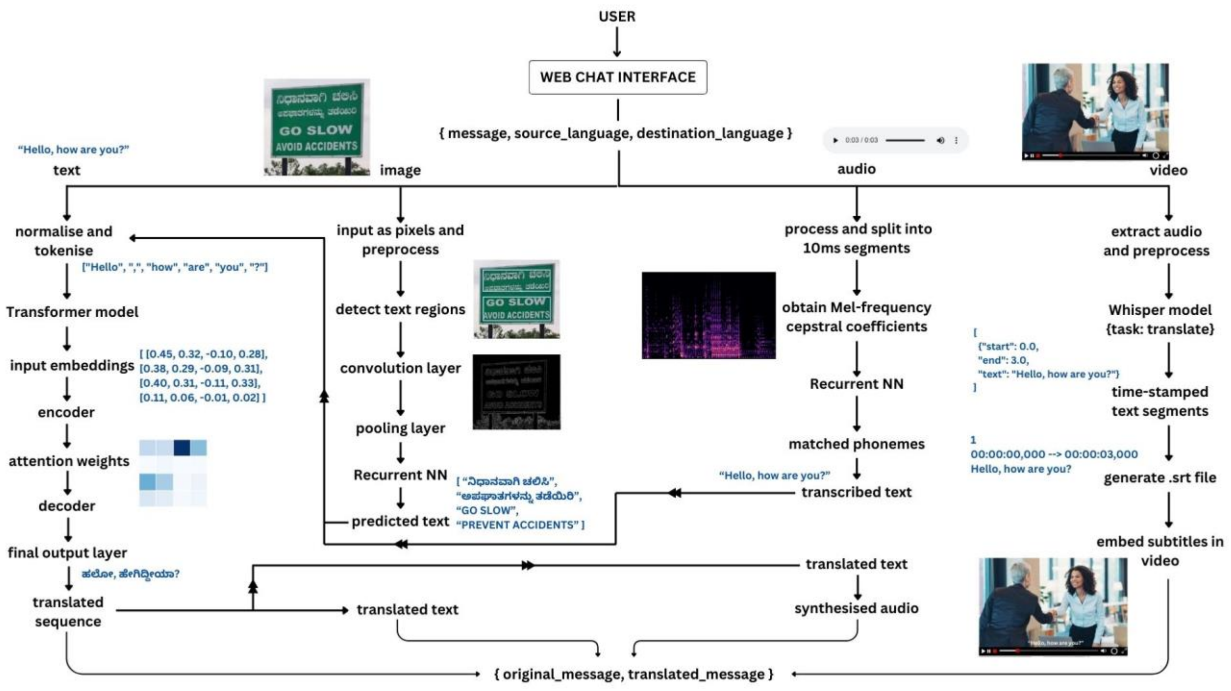


Fig. 2: Flow Diagram

User Authentication & Chat Interface

The application uses a web-based interface using React.js with Next.js, integrated with a Python backend built with Flask, communicating via HTTP. The server leverages Flask-SocketIO and the frontend uses the socket.io library in Node.js, to set up the WebSocket framework. The WebSocket connection is initialized by emitting a "connect" event from the client. Messages are sent as JSON objects containing metadata such as IDs of the sender and receiver, message content, and timestamp. Custom events like "message" and "disconnect" are defined to handle specific actions during a chat session.
User registration and authentication is performed using JSON web tokens and passwords are encrypted using bcrypt. After successful login, the server upgrades the protocol from HTTP to WebSocket, establishing a persistent connection. JWT tokens are passed as part of the headers and validated on the server before upgrading the protocol. The server manages multiple client connections simultaneously and

maintains their state using unique session identifiers. Real-time UI updates are handled through React hooks like useState, useEffect as well as Redux for global state management.
The user is provided a list of available chat participants along with their respective preferred language of communication. Selecting a user directs them to the chat screen with that user, populated with previous messages from the database. The database has three collections: users, messages, and chats. A cloud store is maintained through the client-side API provided by Cloudinary. When the sender sends a message, the attached media – image, audio or video – is uploaded to the cloud store and its corresponding URL is appended to the payload. Using this URL, the resource is downloaded on the server and sent for further translation processing.

Text Translation

This is implemented using the Google Cloud Translation API. The input text is pre-processed first: this involves lowercasing and removing unnecessary punctuation marks that do not contribute to the meaning. Numerical values are converted into their corresponding word forms, i.e. '3' becomes 'three'. Lastly, common spelling mistakes are corrected. The resultant, normalised text is then tokenized by breaking it down into words and decomposing compound words into their subunits. For example, the word 'intelligently' is tokenized into 'intelligent' and '-ly'. The sequences of token IDs generated are fed into the Transformer model, along with the source and destination languages as specified by the user.

Encoder

The encoder layer primarily transforms the input tokens vectors using embedding layers that capture the semantic meaning of the tokens with respect to the entire sequence and convert them into numerical vectors. They employ positional encodings which are created using a combination of various sine and cosine functions, where each dimension is represented by unique frequencies and offsets of the wave, with the values ranging from -1 to 1 to effectively represent the position.
The Transformer encoder comprises a stack of 6 identical layers, each containing a multi-headed attention mechanism and a fully connected neural network. For each word, the model computes three vectors: query **Q**, key **K**, and value **V**.

Attention

The self-attention mechanism calculates the attention score between two words as:

$$\mathrm{Attention}(Q, K) = \frac{Q.K^{T}}{\sqrt{d_k}}$$

Where:

- **Q** is the query vector for the current word.
- **K** is the key vector for the other words.
- **d_k** is the dimension of the key vectors, used for normalization to prevent excessively large dot products.

A softmax function is applied to these scaled scores to generate attention weights, which determine how much focus each word should place on others in the sequence. These weights are multiplied by the corresponding value vectors **V** to produce an output, which is then passed through a linear transformation.

$$Z = \mathrm{softmax}\big(\mathrm{Attention}(Q, K)\big) \cdot V$$

This process occurs in parallel for all words in the sentence, with each word attending to every other word. The attention mechanism is followed by a position-wise feed-forward network, which consists of two linear layers with a ReLU activation in between. After each sub-layer, layer normalization and residual connections are applied to ensure stable learning and handle the vanishing gradient problem.

Decoder

The decoder layer contains two multi-headed attention layers, a point-wise feed-forward neural network, and incorporates both residual connections and layer normalization after each sub-layer. Similar to the encoder, the input passes through an embedding layer followed by the positional encoding layer. The difference begins in the self-attention mechanism: using masking, current positions are prevented from attending to subsequent positions, meaning that predictions for a particular token (position) can only depend on known outputs at positions before it. Outputs from the first multi-headed attention layer of the decoder serve as values for the second layer, with outputs from the encoder becoming both queries and keys.

Mathematically, the output at each step t is given by:

$$y_t = \text{softmax}\left(W_o \cdot \text{Decoder}(y_{t-1}, Z)\right)$$

Where:

- **y_t** is the predicted word at time step t.
- **W_o** is the output weight matrix.
- **Decoder(y_{t-1}, Z)** is the decoder's operation on the previous word and the context vectors.

The final layer's output is transformed into a predicted sequence, through a linear layer followed by a softmax to generate probabilities over the vocabulary.

AUDIO TRANSLATION & SYNTHESIS

The SpeechRecognition library in Python is employed. The input audio is pre-processed to simplify the speech signal using feature transformation and dimensionality reduction, as well as voice activity detection to reduce it to only portions that contain speech. The resultant is then divided into segments of ten milliseconds each. The power spectrum of each segment – essentially a plot of the signal's power vs. frequency – is mapped to a vector of real numbers known as Mel-frequency cepstral coefficients. To decode the speech into text, a recurrent neural network (RNN) is used to match vectors to one or more phonemes, i.e. a fundamental unit of speech.

We begin with an initial probability distribution for the first segment. At each step, each candidate path is expanded by adding possible phonemes, based on the RNN output probabilities. After expanding, only the top paths are kept, and the rest are discarded. This process is repeated until the end of the audio is reached, resulting in a few final candidate sequences. The one with the highest probability is chosen as the final transcription.

This is passed to the translation module, whose result is then synthesised back into audio of the target language using the gTTS library and sent to the receiver.

IMAGE TEXT TRANSLATION

Using opencv-python, the image is read as a 3-dimensional matrix of pixels. A grayscale version is also created, to increase contrast between text and background. Preprocessing is performed to remove any noise, marks and scratches. This involves Gaussian blurring: it smoothens the image by averaging pixel intensities in a local neighbourhood, which removes minor variations in intensity that might not be part of the text. Following this, text regions are detected, bounding boxes and polygon-shaped areas, which are then organised and merged using their geometric properties. This is done by finding sharp changes in pixel intensity, which are often the borders of text characters.

We use the easyocr package in Python, which is a wrapper around pytorch, for text detection and recognition. The OCR model specific to the source language is loaded. In the convolution (CNN) layer, a feature detector (kernel) – a two-dimensional array of weights – is moved across the receptive fields of the input matrix, checking if the feature is present. The final output is the series of dot products from the input and the kernel, known as a feature map. The map undergoes dimension reduction in the pooling layer. A function such as max pooling or average pooling is applied, where either the maximum value or average value within each patch is computed respectively. This helps in minimizing computational complexity and retains only the most relevant features. Finally, the matrix is flattened to a one-dimensional array and passed through a fully connected recurrent neural network. RNNs are used to identify the relationship between the characters, where output from the network is fed back into the network as the next input, to maintain hidden state and context. The final output of the network is a sequence of character predictions, which are decoded into readable text using connectionist temporal classification (CTC).

TRANSLATED SUBTITLE GENERATION

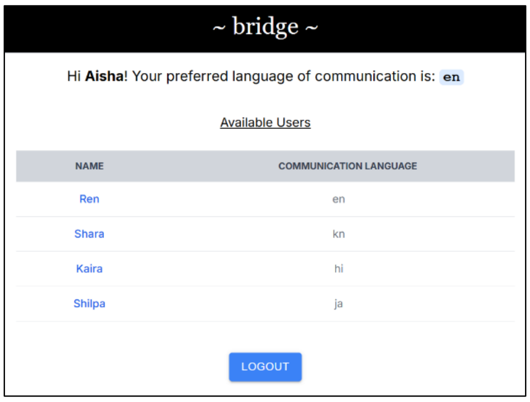
The ffmpeg-python library is used to extract audio from the video input by the user. The input video file is decoded, separating the audio stream from the video frames. By specifying the ".wav" extension, we create a copy of the input file where only the audio stream is saved. To generate time-stamped text segments from audio, we use the faster-whisper library, which is a reimplementation of OpenAI's Whisper model using CTranslate2 for almost four times faster performance, while retaining the original accuracy level.

The audio is split into 30-second chunks for processing, each then converted into a log-Mel spectrogram, which plots the variation of sound's frequency and intensity with time. The spectrogram is passed into Whisper's encoder to generate latent representations of the audio. These are used by Whisper's decoder to generate text tokens. For each token after the first, previously generated tokens are also used as input. Employing a pre-trained multilingual language model, we specify the task as 'translate' to perform speech-to-text translation to English, so a special token <|translate|> guides the model to perform translation. For each token, using voice activity detection, the begin time and end time are also calculated, aligning translated text segments with the duration when they were spoken. This result is formatted into a subtitle file. The subtitles are added as an embedded filter to the video using ffmpeg.

VI. RESULTS AND DISCUSSION

Execution of the code produces a real-time chat application where users can perform multimodal communication in their preferred languages, as shown in Fig. 3–6. The system effectively demonstrates the capacity to handle language translation, text extraction, and speech recognition in multiple modes. Evaluations show accuracy in transcriptions and translations, performing well even with challenging elements like low-quality images and noisy backgrounds. Feature extraction using MFCCs for ASR and convolutional layers for OCR are some key improvements that have contributed to the robust system. The attention-based architecture for translation aids in preserving context while adapting to different language pairs.

By combining speech-to-text, OCR, and translation into a single process, this solution offers a benefit over current applications that only offer isolated translation capabilities. Unlike other platforms with integrated translation features, which require a continuous network connection to perform translations and restrict subtitle access to within the app, here we deliver translated output to the user, eliminating the need for further network connection and enabling access even when viewing media outside the application.



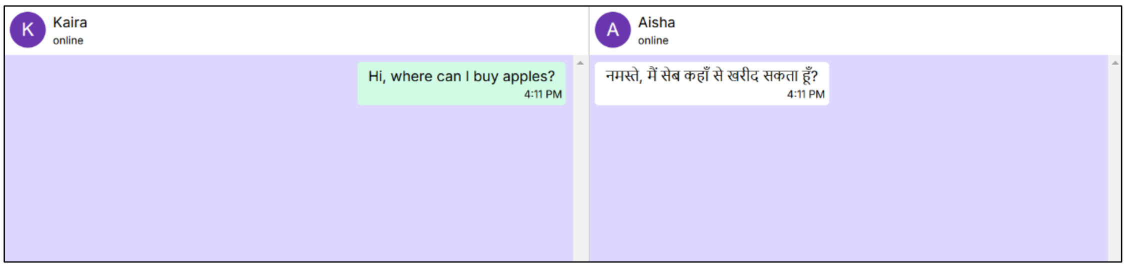
~ bridge ~

Hi **Aisha**! Your preferred language of communication is: `en`

Available Users

NAME	COMMUNICATION LANGUAGE
Ren	en
Shara	kn
Kaira	hi
Shilpa	ja

Fig. 3: User Home Page



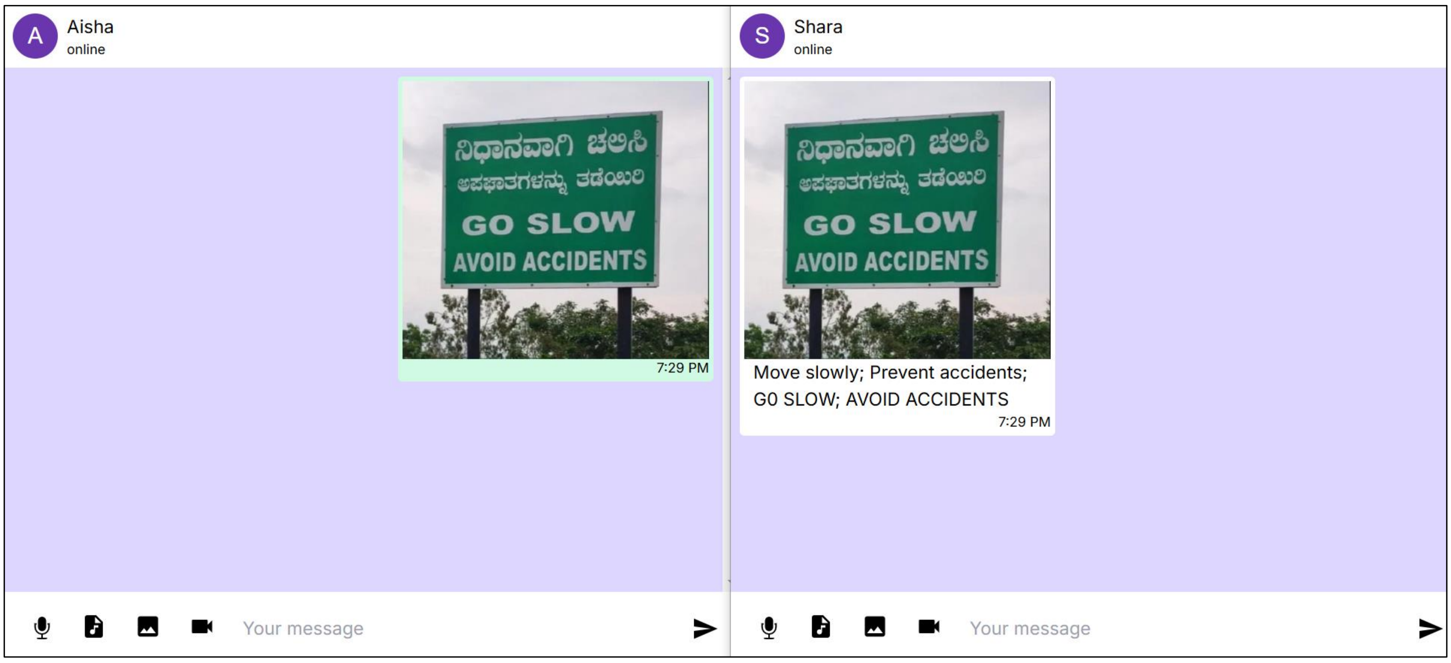
Kaira
online

Hi, where can I buy apples?
4:11 PM

Aisha
online

नमस्ते, मैं सेब कहाँ से खरीद सकता हूँ?
4:11 PM

Fig. 4: Text Translation from English to Hindi



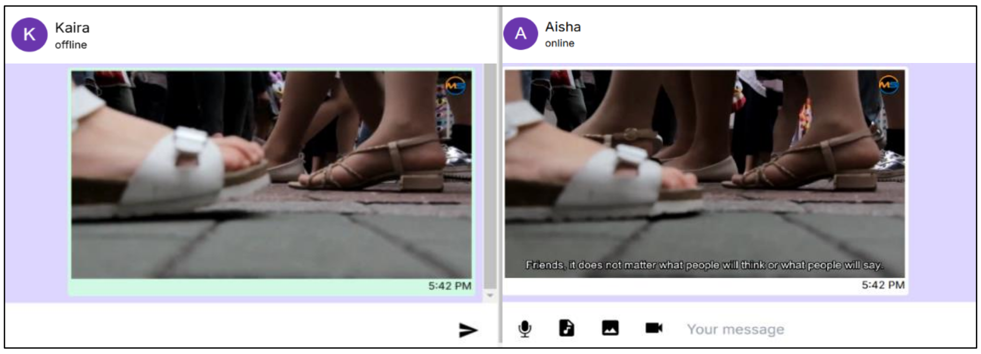
Aisha
online

7:29 PM

Shara
online

Move slowly; Prevent accidents;
G0 SLOW; AVOID ACCIDENTS
7:29 PM

Fig. 5: Image Text Translation from Kannada to English



Friends, it does not matter what people will think or what people will say.

Fig. 6: English Subtitle Generation for Hindi Video

ACCURACY RESULTS

Table 1: Evaluation Metrics for Translation and Recognition Models

Evaluation Metric	Translation	Image Text Extraction	Automatic Speech Recognition	Translated Transcription
Word Error Rate (WER)	–	–	~0.05	~0.11
Character Error Rate (CER)	–	~0.1	~0.05-0.1	~0.1-0.2
BLEU	~0.4	–	–	~0.3
F1-Score	–	~0.8-0.9	–	–
Accuracy	~0.94	~0.8-0.9	~0.9	~0.9
Real-Time Factor (RTF)	–	–	~0.2-0.3	–

As demonstrated in Table 6.1, each model was assessed using the following standard metrics:
Error Rate: WER measures the difference between the predicted text and the actual text, calculated as the number of substitutions, deletions, and insertions divided by the total number of words in the reference text, while CER performs the same at character-level.
BLEU: It is a metric for evaluating the quality of text translation by comparing the n-grams in the predicted translation to the reference translation.
F1-Score: The harmonic mean of precision and recall, offering a balance between the two for evaluating the accuracy of predictions.
Accuracy: The proportion of correct predictions out of all predictions, indicating the overall correctness of the model.
Real-Time Factor (RTF): The ratio of processing time to the duration of the audio being processed, indicating how fast a model performs in real-time settings.

VII. FUTURE WORK

This project has the potential to go beyond just being a chat application. It can be integrated with IoT devices, which can help travellers, differently-abled individuals, and even professionals who need quick and reliable translations.

For instance, a camera-equipped device could read street names, signboards, or instructions and then provide spoken translations in real-time. This could be life-changing for visually impaired people, giving them the confidence to navigate their surroundings independently. Travellers could also benefit by using the app to understand signboards, menus, or the spoken information of people when visiting new places, making their trips smoother and more enjoyable.

In addition, the app could make communication even easier by converting translated text into synthesized speech. Travellers could type something in their language, and the app could synthesize an audio translation in the local language, making it easier to communicate with locals without any language barriers. These features could make the app an incredible tool for accessibility and inclusivity, and be a part of people's everyday life.

REFERENCES

1. M. Chen. (2021). A Deep Learning-Based Intelligent Quality Detection Model for Machine Translation. IEEE Access, vol. 11, pp. 89469-89477.
2. J. Liu, C. Liu, B. Shan and Ö. S. Ganiyusufoglu. (2024). A Computer-Assisted Interpreting System for Multilingual Conferences Based on Automatic Speech Recognition. IEEE Access, vol. 12, pp. 67498-67511.
3. L. Huang, S. Zhuang and K. Wang. (2020). A Text Normalization Method for Speech Synthesis Based on Local Attention Mechanism. IEEE Access, vol. 8, pp. 36202-36209, 2020.
4. Z. Chen, F. Yin, X. -Y. Zhang, Q. Yang and C. -L. Liu. (2021). Cross-Lingual Text Image Recognition via Multi-Task Sequence to Sequence Learning. 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 2021, pp. 3122-3129.
5. Hossain, MD & Ismail, Mohamed. (2017). Auto Translation Chat System. International Journal of Information Systems and Engineering. 5. 88-104.
6. Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All You Need. Neural Information Processing Systems. https://arxiv.org/abs/1706.03762
7. Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023, July). Robust Speech Recognition via Large-Scale Weak Supervision. In International Conference on Machine Learning (pp. 28492-28518). PMLR. https://arxiv.org/abs/2212.04356
8. J. Andhale, C. Dadi and Z. Fei. (2017). A Multilingual Video Chat System Based on the Service-Oriented Architecture. IEEE Symposium on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA, 2017, pp. 126-131.
9. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv.org. https://arxiv.org/abs/1810.04805