International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 19, Number 1 (2023), pp. 37-50

https://dx.doi.org/10.37622/IJCIR/19.1.2023.37-50

Language Interpretation Using Machine Learning

***Dr. P.Sunitha Devi, **Chepuri Sai Tejaswini, **Minati Srinivas, **Manusree Cheruvu, **Modem Keerthana**

**Assistant Professor, **Students*
Narayanamma Institute of Technology & Science (For Women)
sunitha@gnits.ac.in

Abstract

Wouldn't it be cool for your device to translate English language into the language you speak in real life and vice versa? Understanding multiple languages can be a challenging task for humans. In a country like India, where there are over 22 recognized languages, it can be nearly impossible for an individual to learn and comprehend all of them. Consequently, communicating effectively in such a diverse linguistic environment can pose significant obstacles, forcing the need of human interpreters. An interpreter is a platform which translates what is being said in one language into another language without changing its original meaning. Google Translate is one such existing platform but it has drawbacks like not ideal for confidential documents, cannot provide a perfectly accurate translation and requires internet connection to translate. Our goal is to address the limitations mentioned earlier, and to accomplish this objective, we have come up with a project idea that involves real-time translation of spoken languages. The project aims to achieve this by using audio as an input. The proposed system will take real-time English audio as an input, translate it into Telugu and then will produce audio output in Telugu and vice versa. As English is the third most spoken native language in the world after Chinese and Spanish and also the most widely used, we chose to convert the text from English to Telugu and vice versa. Our intention is to utilize a Sequence-to-Sequence model as it is proficient in performing language translations. This model works by transforming a sequence of words from one language into a sequence of different words in another language. It uses neural network for sequence learning.

Keywords: Automatic Speech Recognition, LSTM.

Introduction

Individuals must interact with each other in order to exchange consciousness, emotional state, ideas, thoughts, and facts. India boasts of being one of the most culturally and linguistically diverse countries across the globe. With a vast population, it ranks second in the world in terms of the number of languages spoken by its citizens. This diverse population communicates using their respective regional languages. The two most widely spoken Indo-Aryan languages in the collection are spoken by people in India. The second additional official language is English. English is used differently around the world for human communication. The material made available on the Internet has a significant influence on English. English is spoken by 20% of the world's population, yet only 0.2% of people in India do so. This complicates communication between English and the regional Indian languages. There is a need for practical and accurate computational techniques to bridge this enormous language divide with the least amount of human involvement. This task can be successfully completed using computer perception. A technology known as machine translation (MT) is one that can translate human communication into another language with the least amount of human intervention. The goal of machine interpretation is to provide translations in the target language that are syntactically accurate and have the same meaning as the source phrase. For translating lengthy texts in scientific documents, news articles, and other materials, machine translation has shown to be a useful tool. Over the past decade, the growth of industries and the increased flow of information across various regional languages have significantly impacted the machine translation market, which requires access to information in all regional languages. In the 1950s, the primary funders of machine translation programs were the US military and intelligence organizations, who were interested in the rapid and accurate translation of relevant items. However, during the 1960s, the language barriers became more complex and challenging, and it became apparent that automated solutions alone were insufficient to resolve the translation issue.

Related Work

Machine translation systems were created during the early stages of research using bilingual dictionaries and some manually created rules[1]; nonetheless, it proved challenging to manage all linguistic abnormalities using these manually created rules[3]. As processing power improved in the 1980s, the rule-based approach gave way to statistical machine translation (SMT). The emergence of deep learning and the availability of vast parallel corpora[2] led to a paradigm shift from statistical to neural models[5]. The elimination of the language barrier is the primary goal of machine translation. Earlier studies in this area began by directly substituting target language for source language word[6] for word. Later, data-driven models like statistical and neural machine translation techniques underwent a paradigm shift as a result of advancements in computer and communication technologies. The primary contributions of one study were: (i) an encoder-decoder-based English to Urdu[11] machine translation model with attention; (ii) the development of a news parallel corpus; and (iii) the evaluation of the machine translation model using various

metrics. The primary driving force for conducting this study was the discovery that, despite multiple existing models being put up for other language combinations, the Urdu[12] language received relatively little consideration. It was concluded that a novel paradigm in machine translation research, known as neural machine translation, had been developed. An LSTMbased deep learning encoder-decoder[4] model for translating from English to Urdu[8] is proposed in this paper. They applied their suggested methodology in another study that used the English-Telugu[7] language pair. Utilising their neural model's attention mechanism, a framework was created. Their suggested model is based on NMT. The typical machine translation methods were found to be swift and efficient enough for processing, it was concluded. They have been demonstrated to be crucial in producing excellent results while having a limited operational capability. However, they have trouble creating corpora[10] or the desired language, which can be spoken fluently and with the help of humans.

Neural machine translation overcomes the limitations of conventional machine translation[9] methods. Recent NMT models, like Seq-2-Seq, have produced the desired language with excellent results. Although the model drastically reduces its accuracy in some real-time situations, particularly when it must go through rare words. The most current NMT models, which create a context vector via an attention mechanism, circumvent this problem. The model[14] won't lose accuracy even when it encounters unidentified words. For the first time ever, a translator translated Bangla from Arabic[13] using Sequence-to-Sequence (seq2seq) models. A typical Seq2Seq model typically consists of two main parts: an encoder and a decoder. Mistranslations primarily happened for no vocalisations, it was concluded. Because there were few diacritics, they discovered[15] many homographs, which made it challenging to determine the meaning. Homonyms posed a significant additional issue.

Existing Systems

Google Translate is a neural machine translation service created by Google, designed to translate text, documents, and websites from one language to another. It features a web interface, an iOS and Android mobile app, and an API for programmers to create browser add-ons and software apps. According to recent data from December 2022, Google Translate supports 133 languages at various levels and has claimed over 500 million total users, translating more than 100 billion words daily.

On the other hand, Shabdkosh.com offers one of the world's most popular English to Indian Language Dictionary services. This site, which began in 2003, has a comprehensive database and user-friendly interface, including voice pronunciations in multiple accents, to ensure that Indian language resources are on par with those of any other language worldwide. The site has been a forerunner in the field, having published the world's first Indian language online dictionary without requiring any font installation and providing spoken pronunciations in English and several Indian languages. The layout and style of the website have also served as a model and source of inspiration for many of the rivals.

Architecture

The proposed work flow is as shown in Fig 4.1. The system makes use of sequence-to-sequence model with encoder/decoder. The architecture comprises of three modules namely Automatic Speech Recognition, Machine Translation and Text-to-Speech. The main emphasis is on Machine Translation module to enhance the interpretation.

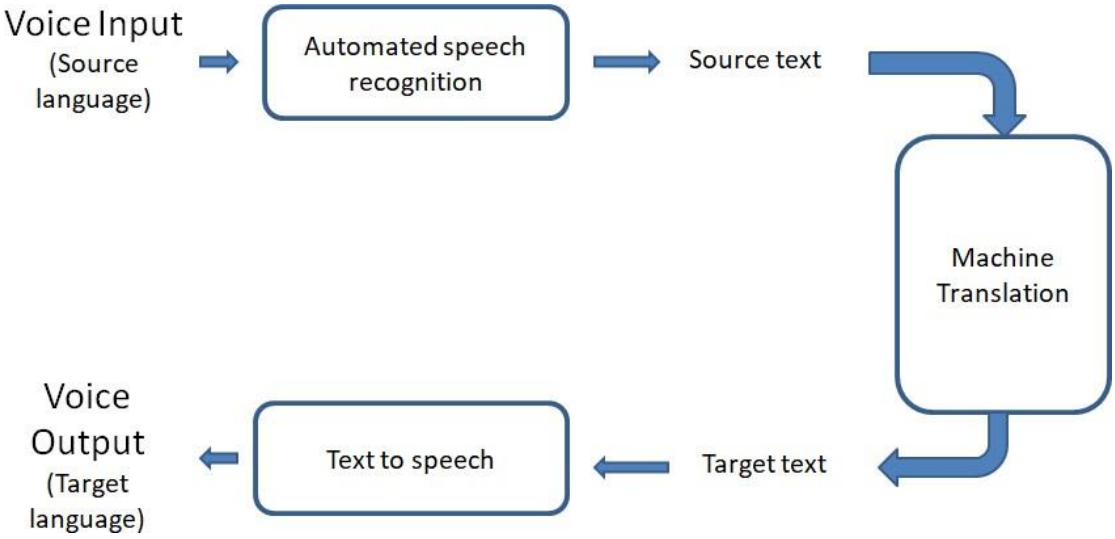

Fig 4.1: Proposed work flow

Dataset

The dataset is collected from indiccorp. IndicCorp consists of datasets belonging to different languages. IndicCorp has been developed by discovering and scraping thousands of web sources-primarily news, magazines, and books, over a duration of several months. IndicCorp is one of the largest publiclyavailable corpora for Indian languages. For this project the English and Telugu datasets as shown in Fig 4.2 and Fig 4.3 respectively are chosen. Each dataset is a file consisting of 1.5 lakhs sentences.

Fig 4.2: English Dataset

Fig 4.3: Telugu Dataset

Automatic Speech Recognition

Speech recognition, commonly referred to as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, is a feature that allows software to convert spoken language into written text. The three types of modelling used by ASR algorithms are acoustic modelling, language modelling, and pronunciation modelling. ASR deals with the link between auditory signals and linguistic components of speech, such as phonemes. Language modelling in ASR searches for patterns in word sequences and so aids in the differentiation of various words with the same sound. An acoustically/phonetically motivated transcript of speech can be mapped to a conventional symbolic transcript of speech that can display variable degrees of arbitrariness. This is known as pronunciation modelling in automatic speech recognition (ASR). The speech recognition pipeline can be described by the Fig 4.4.

Fig 4.4: Speech Recognition Pipeline

The input consists of an audio stream that has been converted into a series of audible features appropriate for decoding. By utilising the PM (Pronunciation Model), AC (Acoustic Model), and LM (Language Model) models, the decoder creates a list of words that are recognised from the input acoustic data. From the provided transcript sequence, PM generates a sequence of phones or potential pronunciations, AC maps acoustic properties to a sequence of phones, and LM adds context by calculating the likelihood that specific words will follow one another in a particular phrase. The decoded sequence has a MAP (Maximum A Posteriori) probability since the conventional statistical approach employs a straightforward probabilistic model to find the word sequences that are most likely to be present in the acoustic observation, P(w, a).

Machine Translation

The encoder/decoder sequence-to-sequence model is used in the proposed system. Even if the sequence length increases, the encoder/decoder's LSTM networks can predict the next value in the sequence. Recurrent Neural Networks (RNN) had a short-term memory problem that LSTM was designed to address. If the sequence was sufficient, the RNN could not store the information. While LSTM have internal systems known as gates that can control the information flow. As depicted in Fig. 4.6, a Seq2Seq model takes a sequence of objects (words, letters, time series, etc.) and produces another sequence of items. With neural machine translation, a set of words serves as the input and a set of translated words serves as the output. An encoder and a decoder make up the model. In the form of a hidden state vector, the encoder records the context of the input sequence and passes it to the decoder, which subsequently creates the output sequence. Both the encoder and the decoder often use some type of RNNs, LSTMs, GRUs, etc. because the task is sequence-based. By design, LSTMs take two inputs: the most recent example they have access to and a

representation of the input from before. As a result, both the input at time t-1 and the present input affect the output at time step t. The network's hidden state stores the sequential data, which is then utilised in the subsequent instance.

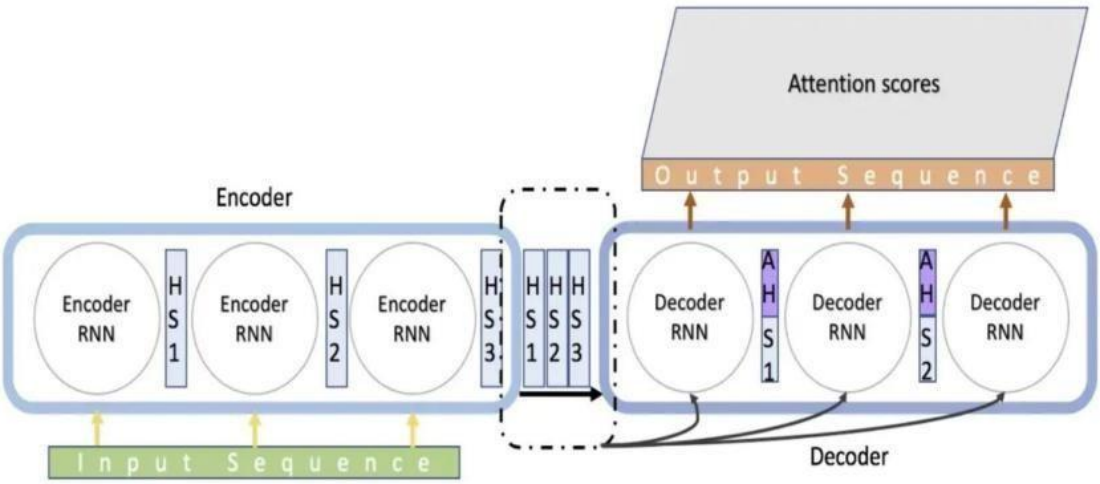

Fig 4.6: Encoder-Decoder

A method known as "Attention" enables the model to concentrate on different elements of the input sequence at each stage of the output sequence, preserving the context from start to finish. The attention mechanism, when used in the context of encoder-decoder models, is a strategy that aids the decoder in concentrating on particular elements of the input sequence (coded by the encoder) when producing each output token. A set of weights that reflect the relevance of each input sequence element to the current decoding step are computed by the attention mechanism during the decoding process in more detail. After computing a weighted total of the encoded input sequence using these weights, the decoder uses this sum as an input to produce the subsequent output. This approach enables the model to preferentially concentrate on the most crucial portions of the input sequence at each decoding stage, which can improve performance in tasks like speech recognition or machine translation.

4.4 Text-to-Speech

Python's Pyttsx3 module converts text to speech. It works offline and is appropriate for Python versions 2 and 3, in contrast to competing libraries. The pyttsx3.init() factory method is used by an application to obtain a reference to a pyttsx3. Engine instance. The tool that turns the entered text into speech is fairly simple to use. The "sapi5" for Windows programme provides the first voice, which is a female voice, and the second voice, which is a male voice. The key attributes of the pyttsx3 library are: it operates entirely offline; you can select from a variety of voices installed on your computer; you can regulate speech speed; you can adjust loudness; and you can save the speech audio into a file. The modular architecture of Pyttsx3 makes switching between several TTS engines simple. Fundamentally, it offers a standard API that

abstracts away the peculiarities of each TTS engine, allowing you to use the same code to produce speech with several engines.

Implementation

Automatic Speech Recognition

Python's speech recognition module is a package that makes it simple to recognise speech from audio input. The module gives access to a number of speech recognition APIs, such as the Wit.ai API, CMU Sphinx API, and Google Speech Recognition API. You can use this module to recognise speech coming from a variety of sources, including streaming audio, audio files, and microphone input. Additionally, it gives users the option to change the language, grammar, and audio source settings for recognition. The Speech Recognition library needs to be installed in order to use speech recognition in Python to detect voice. It is necessary to import the installed library before creating the "Recognizer" object. After that, the voice is recorded using the microphone feature. The Speech Recognition library's Microphone class has a constructor called Microphone () that is used by this function. It initialises a Microphone class instance when it is invoked so that audio from a microphone source can be captured. If the requested microphone is unavailable or if there are problems with the audio stream, this function may throw an error. In some circumstances, the issue might be resolved by specifying a new microphone index or by investigating the audio stream. It is possible to record audio from an Audio Source object and save it as an Audio Data object by using the listen () method of the Recognizer object. A new instance of the Microphone class is created by the listen() function to initialise the microphone. It starts capturing audio data and opens the microphone source's audio stream. It continuously reads information from the microphone stream and adds it to a buffer during recording. The audio stream is stopped after the recording is finished, and the recorded audio data is returned as an instance of the Audio Data class. Additionally, a method is used to transform the audio to text. In order to translate spoken language into text, the process of voice recognition generally combines signal processing, statistical modelling, and machine learning approaches.

Machine Translation

Pre-processing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. Pre-processing is done by the conversion of all the characters to lower case, removing special characters and white-spaces. Then adding '<start>' and '<end>' tokens to the sentences.

Model training

Tokenization is done on the pre-processed text. Tokenization is the process of breaking larger text corpora, essays, or data sets into smaller pieces. These more

compact units could be lines of text or smaller documents. They could also serve as a word dictionary. We may vectorize a text corpus using the Keras Tokenizer, which converts each text into a series of integers, each of which represents the index of a token in a dictionary. A Tokenizer instance is created, and the fit_on_text method is called on the instance. Based on a collection of texts, fit_on_texts revises internal vocabulary. Using this strategy, the vocabulary index is generated based on word frequency. The size of the dictionary matches the amount of the vocabulary. Then the maximum length of source and target languages are computed. The most common architecture used to build Seq2Seq models is Encoder-Decoder architecture as shown in Fig 5.1.

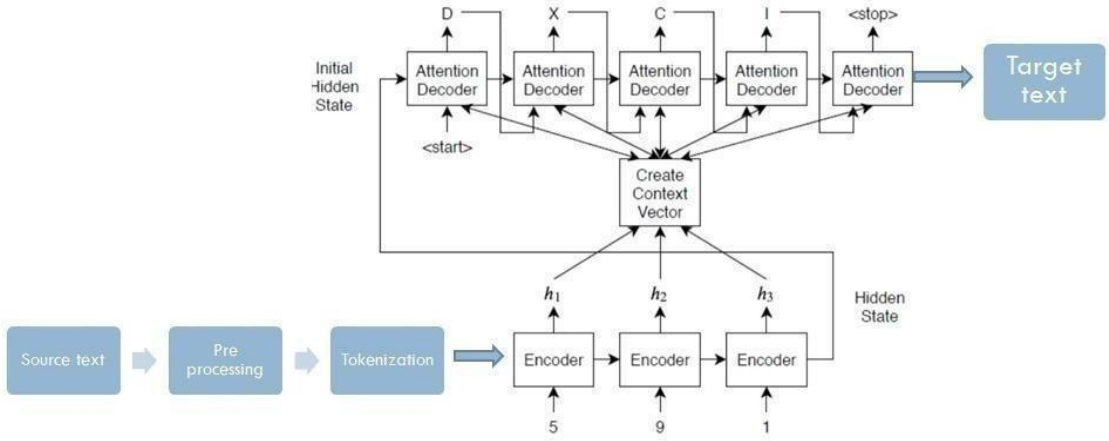

Fig 5.1: Encoder-Decoder architecture

The dataset is split into two different sets using the "train_test_split" function offered by the Python Scikit-learn module, one for training the model and the other for assessing its performance. The function receives a dataset (often a numpy array or a pandas DataFrame) and the appropriate test size (0.2) or proportion (20%), and it outputs two sets of data: one for training and one for testing. The datasets' input-output pairs are produced using the "from_tensor_slices" function. A dataset is produced from a tensor using this TensorFlow method. When a tensor is provided as input, the method returns a Dataset object that represents each tensor element as a separate entry in the dataset. After that, batches are created from the dataset. To increase the effectiveness of training models on huge datasets, machine learning practitioners frequently divide data into batches. There are defined classes for encoders and decoders. The input sequence must be encoded by the encoder into a fixed-size vector representation. Typically, the encoder is implemented as a recurrent neural network (RNN), such as a Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM) network. The size of the input vocabulary, the dimensionality of word embeddings, and the number of units in the encoder RNN are the three hyperparameters that are input to the Encoder class. The decoder is in charge of producing the output sequence while taking the attention mechanism into

consideration in a sequence-to-sequence (seq-toseq) paradigm. The decoder is built as a recurrent neural network (RNN) like a Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) network. The three hyperparameters that the Decoder class accepts as input are vocab_size, which is the size of the output vocabulary, embedding_dim, which is the dimensionality of the word embeddings, and dec_units, which is the number of units in the decoder RNN. Encoder and decoder instances are made available to save checkpoints. During the training phase, checkpoint saving is a machine learning technique used to save the model parameters at regular intervals.

Text-to-Speech

Pyttsx3 is a Python library for text-to-speech conversion that uses platform-specific text-to-speech engines. Here's a detailed overview of how it works internally with its architecture: Text input: The pyttsx3 library first receives text input, which is typically in the form of a string.

1. *Initialization:* The library initializes a text-to-speech engine, such as the Microsoft Speech API or the eSpeak speech synthesizer. This is done using the platform-specific Text-to-Speech (TTS) API, which provides a way for programs to generate synthesized speech.
2. *Configuration:* The library allows for the configuration of various properties of the text-to-speech engine, such as the voice, rate, volume, and pitch. These properties can be set using the library's setProperty() method.
3. *Synthesis:* The library passes the text input to the text-to-speech engine, which synthesizes speech output based on the specified configuration. The engine generates an audio signal that represents the synthesized speech.
4. *Playback:* The library then plays back the synthesized speech through the computer's audio output, which can be a speaker or headphones. This is done using the platform-specific audio output API, which provides a way for programs to play the audio. It provides a simple interface for performing text-tospeech synthesis. The pyttsx3 library is first initialized by creating an instance of the Engine class. This initializes the underlying text-to-speech engine and sets default properties, such as the voice and rate. The Engine class provides methods for configuring various properties of the text-to-speech engine, such as the voice, rate, volume, and pitch. These properties can be set using the setProperty() method. The Engine class provides several methods for synthesizing speech from text. The simplest method is the say() method, which takes a string of text as input and generates spoken output.The Engine class uses a platform-specific audio player to play back the synthesized speech.

Results

The model architecture used in this project was a sequence-to-sequence model with attention mechanism. The encoder consisted of a bidirectional LSTM layer, followed by a decoder with a single LSTM layer and attention mechanism. The model was

trained on a dataset of parallel English and Telugu sentences. The training was performed for 15 epochs. During training, the model achieved a training loss of 0.04.The goal of machine translation is to produce results equal to those of a professional human translator. BLEU is an algorithm which is designed to measure exactly that. Because it is fast and cheap to carry out, BLEU has stayed one of the most popular automated metrics for determining the quality of Machine Translation. BLEU stands for Bilingual Evaluation Understudy. A BLEU score is a quality metric assigned to a text which has been translated by a Machine Translation engine. The goal with MT is to produce results equal to those of a professional human translator. BLEU is an algorithm which is designed to measure exactly that. Because it is fast and cheap to carry out, BLEU has stayed one of the most popular automated metrics for determining the quality of Machine Translation. A BLEU score runs on a scale from 0 to 1. A BLEU score measures a given machine-translated text against a test reference text which has been human-translated. On the BLEU scale: 1 (or 100) is the top end of the scale. The closer a BLEU score is to 1, the more it resembles the reference text from the human translator. If BLEU scores hit 1, it would be identical to the specific human reference work supplied. However, this is so unlikely that it is essentially impossible. 0 (zero) is the bottom end of the scale. Results towards this end of the scale indicate that there is much less overlap with the test human reference text.

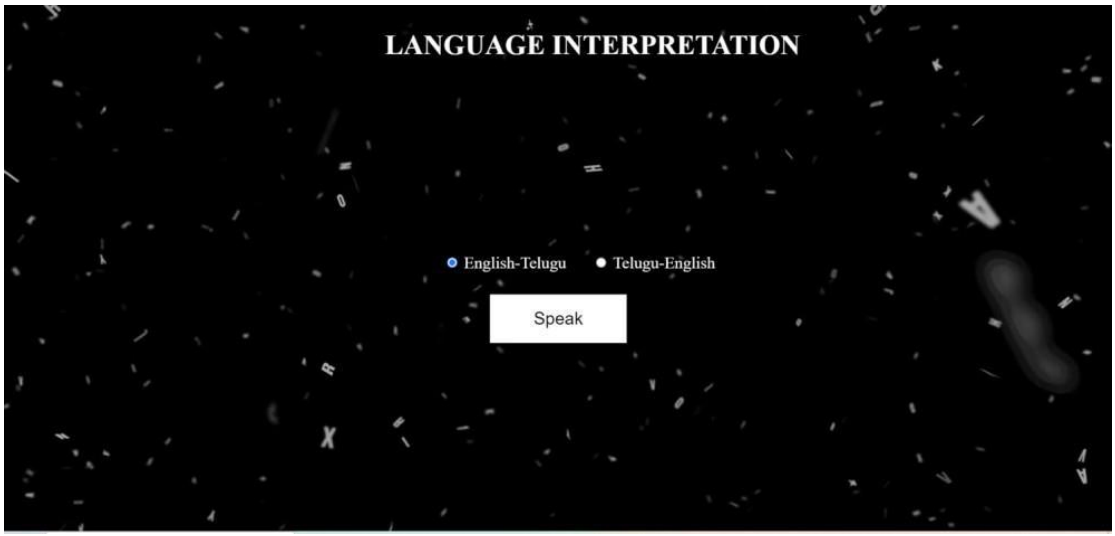

Fig: 5.1 English-Telugu Interpretation

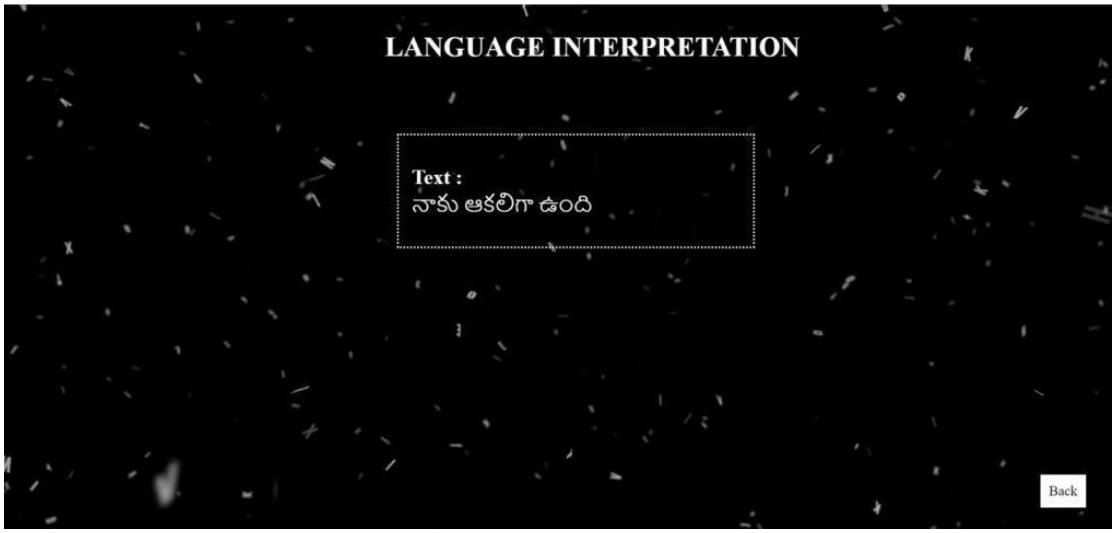

Fig: 5.2 Telugu Interpretation

Fig: 5.3 Telugu-English Interpretation

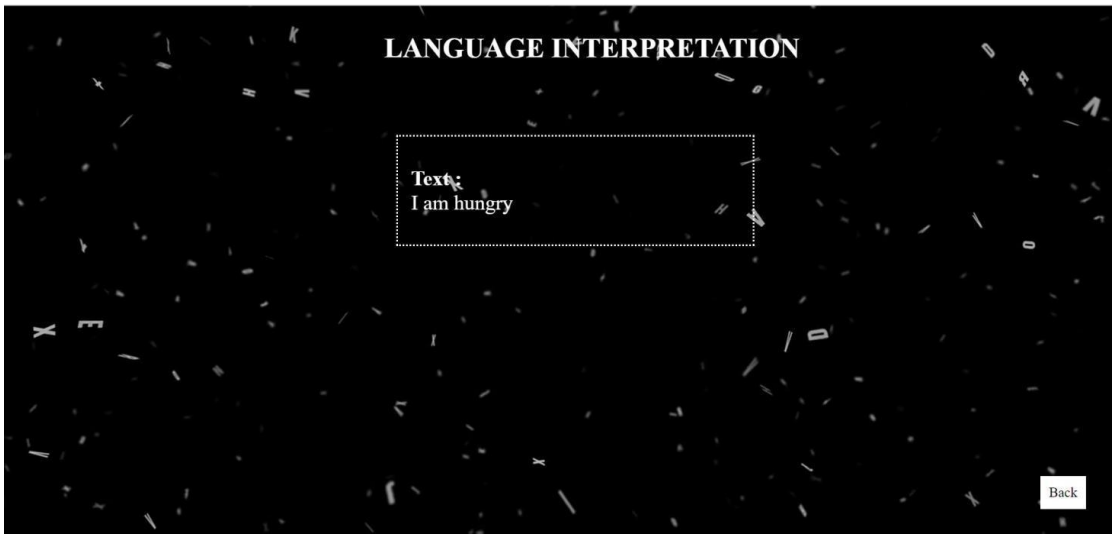

Fig: 5.4 English Interpretation

Conclusion

The main aim of machine translation is to remove the language barrier. An interpreter is a platform which translates what is being said in one language into another language without changing its original meaning. With this project we were able to convert Source speech to source text and then interpret the source text to most accurate target text. We were able to perform this interpretation offline which is a major shortcoming of the existing systems. The interpretation is done using Sequence-to-Sequence model. The Speech Recognition model chosen can convert speech to text so that it can be given as input to the Machine Translation model. The built model can accurately interpret the sentences which is given as combination of words from the dataset. The model is also able to capture longer sentences and provide appropriate translation.

References

[1] A. Ali, A. Hussain, and M. K. Malik, "Model for English-Urdu statistical machine translation," World Applied Sciences Journal, vol. 24, no. 10, pp. 1362–1367, 2013.

[2] A. Ali, S. Siddiq, and M. K. Malik, "Development of parallel corpus and English to Urdu statistical machine translation," International Journal of Engineering, vol. 10, no. 05, pp. 3–6, 2010.

[3] Bart Kahler, Brian Bacher, and K.C. Jones "Word-order issues in English-toUrdu statistical machine translation," Be Prague Bulletin of Mathematical Linguistics, vol. 95, no. 1, pp. 87–106, 2011.

[4] C. Adak, "A bilingual machine translation system: English & Bengali," in Proceedings of the 2014 FirstInternational Conference on Automation, Control,

Energy and Systems (ACES), pp. 1–4, Adisaptagram, India, February 2014.
[5] Manuel A. Pérez-Quiñones, Olga I. Padilla-Falto, Kathleen McDevitt "Automatic Language Translation for User Interfaces"
[6] R. M. K. Sinha and A.)akur, "Machine translation of bilingual Hindi-English (hinglish) text," in Proceedings of the 10th Machine Translation summit (MT Summit X), pp. 149– 156, Phuket,Tailand, September 2005.
[7] R. M. K. Sinha and A. Jain, "AnglaHindi: an English to Hindi machine-aided translation system," in Proceedings of the Ninth Machine Translation Summit, New Orleans, LA, USA, September 2003.
[8] R. M. K. Sinha, K. Sivaraman, A. Agrawal, R. Jain, R. Srivastava, and A. Jain, "ANGLABHARTI: a multilingual machine aided translation project on translation from English to Indian languages," in Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, vol. 2, pp. 1609– 1614, Vancouver, UK, October 1995.
[9] R. Udupa and T. A. Faruquie, "An English-Hindi statistical machine translation system," Lecture Notes in Artificial Intelligence (Subseries Lecture Notes in Computer Science, vol. 3248, pp. 254–262, Springer, Berlin, Germany, 2004.
[10] S. A. B. Andrabi and A. Wahid, "A review of machine translation for south asian low resource languages," Turkish Journal of Computer and Mathematics Education, vol. 12, no. 5, pp. 1134– 1147, 2021.
[11] S. H. Kumhar, W. Qadir, M. Kirmani, H. Zargar, and M. Hassan, "Effectiveness of methods and tools used for collection of roman Urdu and English multilingual corpus," Design Engineering, vol. 7, pp. 13428– 13438, 2021
[12] S. I. Rai, M. U. S. Khan, and M. Waqas Anwar, "English to Urdu: Optimizing sequence learning in neural machine translation," in Proceedings of the 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1–6, Sukkur, Pakistan, January 2020.
[13] Toshiba Kamruzzaman," Arabic To Bangla Machine Translation UsingEncoder Decoder Approach" IEEE Region 10 Symposium (TENSYMP), 5-7 June 2020, Dhaka, Bangladesh
[14] V. Goyal and G. S. Lehal, "Hindi to Punjabi machine translation," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations, pp. 1–6, Portland, OR, USA, June 2011. [15] W. J. Hutchins, Machine Translation: A Brief History, Elsevier Science Ltd, Amsterdam, Netherlands, 1995.