Malayalam Automatic Speech Recognition

Submitted by

Basil K Raju, Mohammed Anshad K
(Register Number: 211109, 211116)

In partial fulfillment of the requirements for the award of Master of Science in Computer Science with Specialization in Machine Intelligence

of

School of Computer Science and Engineering
Kerala University of Digital Sciences, Innovation and Technology
(Digital University Kerala)
Technocity Campus, Thiruvananthapuram, Kerala – 695317

February, 2023

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **Malayalam Automatic Speech Recognition** submitted by **Basil K Raju (Reg. No: 211109)** and **Mohammed Anshad K (Reg No: 211116** in partial fulfillment of the requirements for the award of **Master of Science in Computer Science with Specialization in Machine Intelligence** is a bonafide record of the work carried out at **Kerala University of Digital Sciences, Innovation and Technology** under our supervision.

Supervisor	Course Coordinator
Dr. Elizabath Sherly	Sujouy Madhav
Distinguished Professor	Assistant Professor
DUK	DUK

Head of Institution

Prof. Saji Gopinath
Vice Chancellor
DUK

DECLARATION

We, **Basil K Raju and Mohammed Anshad K** , students of **MSc in Computer Science With Specialization in Machine Intelligence**, hereby declare that this report is substantially the result of our own work , except, where explicitly indicated in the text and has been carried out during the period **September 2022 - January 2023**.

Place:	Trivandrum
Date:	08-02-2023

Student's signature

ACKNOWLEDGEMENT

We are extremely pleased to present our work in Natural Language Processing, the project named ”Malayalam Automatic Speech Recognition.” We would like to extend my sincere thanks to those who greatly contributed to the completion of this project. We are especially grateful to Dr. Elizabeth Sherly of Digital University Of Kerala for the support and guidance throughout the project.

We appreciate the help and support from all members of my department. We are deeply grateful to my parents for their encouragement and love. Finally, We give thanks to the Almighty for his blessings that allowed me to complete this project.

ABSTRACT

Automatic speech recognition (ASR) has numerous applications in areas such as voice dialing, voice commands, voice to text conversion, and more. However, there is still room for improvement in terms of the speed and accuracy of these systems. Wav2Vec 2.0 is a pretrained machine learning model for ASR that has shown remarkable performance. Facebook AI introduced XSLR-Wav2Vec2, which has the ability to learn cross-lingual speech representations for speech recognition in multiple languages. XSLR-Wav2Vec2 uses advanced techniques, such as masked language modeling and large-scale training data, to provide powerful speech representations. In this study, XSLR-Wav2Vec2 was fine-tuned using Connectionist Temporal Classification (CTC) for Malayalam automatic speech recognition. A tokenizer and feature extractor were also developed to process speech into the model's input format and transcribe speech to text. The results showed that XSLR-Wav2Vec2 provides an accurate and efficient solution for Malayalam ASR

Table of Contents

List of Figures

Chapter 1

Introduction

An automatic speech recognition system aims to enable communication between human and computer.Recognition of speech signal has many applications, such as voice dialing, voice commands, voice to text converters,spoken database querying, hands busy applications etc. There lies a significant improvement based on the speech recognition application, but there are still doubts about how fast and reliable the system model is. Therefore, the success of ASR system depends on how accurate the system model. It is measured in terms of accuracy of speech recognition.Automatic speech recognition (ASR) in Malayalam language involves the use of technology to transcribe spoken words into written text. One approach to ASR in any language involves the use of pre-trained models, such as the wav2vec model.

1.1 Problem Statement

Malayalam is a complex and morphologically rich language spoken by over 38 million people in India. Despite its widespread use, there is currently a lack of accurate and reliable ASR systems for the Malayalam language. This poses a significant challenge for applications such as voice-based interfaces, dictation, and speech-to-text transcription, which are increasingly in demand in various industries.

The main challenges in developing a robust Malayalam ASR system include the highly agglutinative nature of the language, the presence of numerous homophones (words that sound the same but have different meanings), and the variability in pronunciation across speakers and dialects. Additionally, the lack of large, high-quality speech corpora for the Malayalam language presents a significant barrier to developing accurate models.

Addressing these challenges requires the development of specialized techniques for modeling the unique properties of the Malayalam language, as well as the collection and annotation of large amounts of speech data for training and testing ASR models. Developing a robust Malayalam ASR system has the potential to unlock new opportunities for speech-based applications in education, healthcare, entertainment, and other domains, and can help bridge the digital divide by enabling better access to information and services for Malayalam speakers.

1.2 Objective

The main objective of building an Automatic Speech Recognition (ASR) system for Malayalam language using Wav2Vec-XLSR is to enable speech-to-text transcription in Malayalam language for various applications. ASR technology has many practical uses, such as transcribing audio recordings, enabling hands-free interaction with devices, and creating subtitles for videos.

By building an ASR system for Malayalam language using Wav2Vec-XSLR, we can enable speech-to-text transcription for Malayalam speakers, which can have many practical applications, including:

- Enabling people with hearing impairments to participate in communication more easily.
- Creating transcripts of speeches, lectures, and other spoken content for accessibility and reference purposes.
- Improving the accuracy and efficiency of customer service systems that use voice recognition.
- Developing voice-activated digital assistants and other hands-free devices that can understand Malayalam language.
- Enabling the creation of Malayalam subtitles for videos and other multimedia content.

Overall, the goal of building an ASR system for Malayalam language using Wav2Vec-XSLR is to make speech recognition technology more accessible and inclusive for Malayalam speakers, and to improve the efficiency and accuracy of speech-to-text transcription for various applications.

1.3 Motivation

Since there are many well established ASR systems available for English, there are no good system available for Malayalam language.This project was started with the aim of building an Alexa in Malayalam.Thus we came to know the importance of a good ASR system for Malayalam language.As a first step building a reliable Malayalam ASR will help to create an ALexa in Malayalam language.Building an ASR system for Malayalam would help to promote and preserve the language and its cultural heritage.An ASR system for Malayalam would make it easier for people who speak the language to interact with technology, such as smartphones, computers, and voice assistants. This would also enable them to access information and services more easily, which can have a positive impact on education, business, and other aspects of life.

1.4 Automatic Speech Recognition

ASR stands for Automatic Speech Recognition, which is the process of converting spoken language into written text. ASR is used in a wide range of applications, including virtual assistants, transcription services, and automated customer service systems.

ASR systems typically consist of two main components: an acoustic model and a language model. The acoustic model takes the audio signal as input and produces a sequence of acoustic feature vectors that represent the speech signal. These feature vectors are typically based on spectral analysis of the audio signal, and they capture information such as the frequencies and amplitudes of the speech signal over time.

The language model takes the sequence of acoustic feature vectors as input and produces a sequence of text tokens that represent the spoken words. The language model uses statistical models or machine learning techniques to predict the most likely sequence of text tokens given the input features.

In order to accurately recognize speech, ASR systems require a large amount of training data. This training data typically consists of audio recordings and corresponding transcripts, which are used to train the acoustic and language models.

ASR is a challenging task because of the variability and complexity of spoken language. Speech signals can be affected by factors such as background noise, speaker accents, and speech rate, which can make it difficult for ASR systems to accurately recognize speech. Additionally, speech signals are highly dependent on context, so the language model must take into account the surrounding words and the overall meaning of the spoken language.

To overcome these challenges, ASR systems use a variety of techniques, such as speaker adaptation, noise reduction, and language modeling. For example, speaker adaptation techniques can be used to personalize the ASR system to a specific speaker, which can improve recognition accuracy. Noise reduction techniques can be used to remove background noise from the audio signal, which can make it easier for the ASR system to identify speech sounds. And language modeling techniques can be used to incorporate knowledge about the structure of the language and the context of the speech signal into the ASR system.

1.5 Wav2vec 2.0

Wav2Vec2 is an innovative pretrained machine learning model for Automatic Speech Recognition (ASR) that was released by Alexei Baevski, Michael Auli, and Alex Conneau in September 2020[2]. This model showed remarkable performance on the English ASR dataset LibriSpeech and quickly attracted the attention of the research community. The wav2vec model is a neural network architecture that was pre-trained on a large amount of audio data using a self-supervised learning approach. The model can be fine-tuned on smaller datasets specific to a particular language, such as Malayalam, to improve its accuracy in transcribing speech.The core of wav2vec 2.0 is its Transformer encoder, which takes as input the latent feature vectors and processes it through 12 Transformer blocks for the BASE version of the model, or 24 blocks for the LARGE version. To match the inner dimension of the Transformer encoder, the input sequence first needs to go through a feature projection layer to increase the dimension from 512 (output of the CNN) to 768 for BASE or 1,024 for LARGE.

The Wav2Vec model is built on top of the transformer architecture. Specifically, it uses a variant of the transformer known as the convolutional transformer, which incorporates convolutional layers along with the self-attention mechanism used in traditional transformers.

The convolutional transformer architecture is well-suited for processing audio signals, which are typically much longer than the sequences of text that transformers are typically used for. By incorporating convolutional layers, the Wav2Vec model is able to efficiently process audio signals of variable length, while still taking advantage of the self-attention mechanism to capture long-range dependencies.

In addition to the convolutional transformer architecture, Wav2Vec also uses several other innovations to improve its performance on speech recognition tasks, such as a contrastive loss function during pretraining, and a quantization-based approach for reducing the computational cost of inference.

Overall, the combination of the transformer architecture with these additional innovations has resulted in a highly effective self-supervised model for speech recognition.

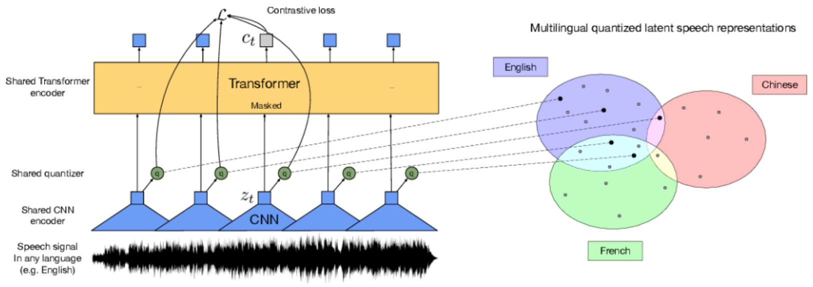

$\mathcal{L}$
Contrastive loss
c_t
Multilingual quantized latent speech representations
Shared Transformer encoder
Transformer
Masked
English
Chinese
Shared quantizer
q
z_t
Shared CNN encoder
CNN
Speech signal In any language (e.g. English)
French

Figure 1.1: wav2vec2 Architecture

1.6 XSLR-Wav2Vec2

Based on the sucess of Wav2vec2 XSLR-Wav2Vec2 is introduced. XSLR-Wav2Vec2 is a pre-trained cross-lingual speech recognition model developed by Facebook AI Research (FAIR). It is a variation of the Wav2Vec2 model, which is a state-of-the-art self-supervised model for speech recognition.

The XSLR-Wav2Vec2 model is designed to support multilingual speech recognition by leveraging pre-existing cross-lingual knowledge. It is trained on a large amount of speech data from multiple languages and is fine-tuned on a target language to improve its performance on that language.The XSLR-Wav2Vec2 model works by first pretraining the model on a large corpus of unlabeled speech data using a self-supervised learning approach.

During pretraining, the model learns to extract features from the raw audio waveform, which are then used to predict a discrete latent representation of the audio. This pretraining step allows the model to capture robust and transferable representations of speech.The model is then fine-tuned on a target language by using labeled data from that language. During fine-tuning, the model is trained to predict the text transcription of the speech signal. The model also incorporates a language model, which provides additional context during the transcription process and improves the model's accuracy.

One of the key benefits of XSLR-Wav2Vec2 is that it can be used to develop speech recognition systems for low-resource languages where labeled data is scarce. By leveraging the pre-existing cross-lingual knowledge captured in the model's pretraining step, the model can be fine-tuned on a smaller amount of labeled data from a low-resource language, resulting in improved accuracy compared to traditional approaches.Similar to the masked language modeling technique used in BERT, XLSR-Wav2Vec2 learns contextualized speech representations by randomly masking feature vectors and then passing them through a transformer network. This combination of advanced techniques and large-scale training data has made Wav2Vec2 and XLSR-Wav2Vec2 leading models in the field of ASR.XLSR-Wav2Vec2 is finetuned using Connectionist Temporal Classification (CTC), which is an algorithm that is used to train neural networks for sequence-to-sequence problems and mainly in Automatic Speech Recognition and handwriting recognition.

Overall, XSLR-Wav2Vec2 is a powerful tool for developing cross-lingual speech recognition systems, and its ability to transfer knowledge across languages makes it a valuable resource for developing speech recognition systems for a wide range of languages.

Chapter 2

Literature Review

Malayalam is a Dravidian language spoken primarily in the Indian state of Kerala. Automatic Speech Recognition (ASR) is a technology that allows machines to transcribe spoken language into written text. ASR has become an increasingly important research area due to its potential to improve human-computer interaction, transcription of speech in various domains, and language learning.

Research on Malayalam ASR has been ongoing for several years. This literature review aims to provide an overview of the key findings and trends in Malayalam ASR research.

Early work in Malayalam ASR focused on developing acoustic models and lexicons for the language. In a study by Sreelekha et al. (2013), researchers developed an acoustic model for Malayalam using Hidden Markov Models (HMMs) and a lexicon based on the Malayalam pronunciation dictionary. The study achieved a word recognition accuracy of 62%.

Subsequent studies focused on improving the accuracy of Malayalam ASR systems. In a study by Jayakumar et al. (2016), the researchers proposed a hybrid approach that combined Deep Neural Networks (DNNs) and HMMs to improve the performance of Malayalam ASR. The study achieved a word recognition accuracy of 70%, which was an improvement over previous results.

Recent research has focused on developing more robust and accurate Malayalam ASR systems. In a study by Sajeev et al. (2020), the researchers proposed a deep learning-based approach that utilized Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to improve the accuracy of Malayalam ASR. The study achieved a word recognition accuracy of 83%, which was a significant improvement over previous results.

Another area of research in Malayalam ASR has been the development of domain-specific ASR systems. In a study by Nair et al. (2019), the researchers developed a Malayalam ASR system for agricultural domain speech. The system achieved a word recognition accuracy of 80% and could be used to transcribe speech related to agriculture in Malayalam.

Overall, research on Malayalam ASR has been ongoing for several years, with researchers focusing on improving the accuracy and robustness of the systems. Recent studies have shown significant improvements in the performance of Malayalam ASR systems, which suggests that the technology is becoming more mature and could be used in various applications. However, there is still room for further research, particularly in developing domain-specific ASR systems for Malayalam.

2.1 Automatic Speech Recognition in Malayalam Using DNN-based Acoustic Modelling

In the paper[4], it describe Malayalam automatic speech recognition using DNN-based acoustic modelling. Mel-frequency cepstral coefficient features (MFCC) are extracted, and acoustic modelling is done using GMM-HMM, SGMM and DNN. The dependency of the performance on different factors like the type of modelling used and alignment algorithms employed are studied. Training and testing of the system were performed using the open-source Kaldi toolkit. We investigated the performance by varying number of hidden layers and hidden units. Testing of the models was carried out on very large vocabulary continuous Malayalam speech recognition task.

Database used consists of a set of Malayalam sentences spoken by 43 speaker (19 male and 24 female).Speaker utterances are generally Malayalam proverbs, newspaper reports, short stories and blogs collected from different resources. Entire dataset comprises of nearly 9 hours of data, of which is split as: 8 hours for training and 1 hour for testing. In the training set, audio files from 19 female and 15 male speakers are used and for test set, 5 female and 4 male speakers are used.

The overall model is speaker independent.The results improved with WER of 3.01% for hybrid DNN from WER of 4.04% of baseline monophone model. Even in the triphone model, the improvement in WER can be noticed from the table, for the techniques Incorporated in the modelling. It is worth noting that the absolute SER reduction of 5% is obtained for DNN modelling as compared to baseline system.

2.2 Continuous Speech Recognition System for Malayalam Language Using Kaldi

An automatic speech recognition system aims to enable communication between human and computer. In this paper[1], a continuous Malayalam speech recognition system is developed and realized using Kaldi toolkit. For building the proposed system, MFCC features and its transformations such as LDA and MLLT are extracted from Malayalam speech sentences. Then, the system has been trained to estimate the HMM parameters for speech sound units. Performance of both monophone and triphone models using bi-gram language model is compared in terms of word error rate(WER). A substantial drop in WER is obtained for triphone models.

The experimental results were reported from the spoken corpus made up of 5114 Malayalam sentences spoken by 20 speakers. Context dependent triphone system and monophone system were developed. The MFCC features and their transformations were used for model generation. Kaldi uses the FST-based framework and the IRSTLM toolkit was used to build the LM model from raw text.Several kinds of experiments are conducted on database collected using Kaldi itself. Case 1: Trained the model using 2901 story sentences and tested with 831 story sentences. Case 2: Trained the model using 2901 story sentences and tested with 200 AIR Read speech. Case 3: Here, the system is trained using 2901 story sentences and also 1200 AIR read speech and tested with 831 story sentences and 200 AIR Read speech.

MFCC feature were chosen for building monophone models. MFCC transformations, such as LDA and MLLT are used for the development of triphone models and the recognition accuracy is compared. The triphone model giving the best accuracy for the out of vocabulary data. In future, use of deep neural network will be explored for improving the performance of ASR system.

2.3 Automatic speech recognition system for utterances in Malayalam language

The paper[3] is discussing the phoneme level interpretation of sound signal to form the words in Malayalam language. Natural language processing with advanced algorithms are adopted here to ensure the accuracy and optimization of the processing. Analog to digital conversion of voice signal takes numerous methodologies for perfect processing. In this paper, training a new system with these techniques to test whether the given commands belong to the dataset or not.

The Mel Frequency Cepstral Coefficient (MFCC) methodis studied here for extracting the features of speech signal.The pre-processing and feature extraction stages of a patternrecognition system serves as an interface between the realworld and a classifier operating on an idealised model of real-ity. Then HMM is used to train these features into the HMMparameters and used to find the log likelihood of entire speechsamples. HMM is a dominant approach in most state-of-the-art speaker-independent, continuous speech recognitionsystems.

In recognition this likelihood is used to recognize the spo-ken word. Speech recognition systems till date are not a centpercentage accurate even though lots of research works arebeing going on in the field of speech recognition. The systemsdeveloped so far have limitations: The current systems arelimited to number of vocabularies and future works are neces-sary towards expanding this vocabulary. ASR techniques canbe extended to speech recognition in Malayalam language.Since Malayalam language has several accents and a dialectthat varies from region to region it poses a great challenge inthis area of research.

Chapter 3

Methodology

The methodology for developing an ASR system using the wav2vec model would involve collecting a dataset of audio recordings in Malayalam, and transcribing them manually to create a dataset of paired audio and text data. This dataset would then be split into training, validation, and testing sets, and the wav2vec model would be fine-tuned on the training set.

Various techniques could be used to optimize the performance of the model, such as adjusting the learning rate, using data augmentation to generate additional training data, and applying language-specific features such as phoneme embeddings or a language model to the model.

The results of the ASR system could be evaluated by comparing the output of the model to the ground truth transcription on the test set. Metrics such as word error rate (WER) or character error rate (CER) could be used to assess the accuracy of the model.

Developing an Automatic Speech Recognition (ASR) system using the wav2vec model involves several key steps as below:

1. Data Collection: The first step is to collect a dataset of audio recordings in the Malayalam language. The dataset should be representative of the domain or application for which the ASR system is being developed. The dataset should include both clean and noisy audio recordings, covering a range of accents, speaking styles, and genders.

2. Data Preprocessing: The audio files in the dataset are preprocessed to convert them into a format that the model can process. This involves converting the audio files into a numerical representation, such as Mel Frequency Cepstral Coefficients (MFCCs).

3. Data Labeling: Once the audio files are preprocessed, the next step is to transcribe them into text. This involves manually labeling each audio

file with its corresponding text transcription. This labeled dataset will be used to train and evaluate the ASR system.

4. Data Splitting: The labeled dataset is divided into three subsets: training, validation, and testing. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, and the testing set is used to evaluate the final performance of the ASR system.

5. Model Selection: The wav2vec model is a pre-trained model, which can be fine-tuned on a specific language or domain. This model can be used as a starting point for training the ASR system. Alternatively, other models can be considered based on their performance on similar tasks.

6. Fine-tuning the Model: The next step is to fine-tune the model on the labeled dataset. This involves adjusting the pre-trained model's parameters to minimize the difference between the predicted transcriptions and the labeled transcriptions. This process may take several iterations, and various techniques can be used to optimize the performance of the model, such as adjusting the learning rate, using data augmentation to generate additional training data, and applying language-specific features such as phoneme embeddings or a language model.

7. Evaluation: The final step is to evaluate the performance of the ASR system on the testing dataset. This involves measuring the error rate of the system, typically using metrics such as Word Error Rate (WER) or Character Error Rate (CER). The results of the evaluation can be used to determine whether the system is suitable for the intended application, and whether further improvements are needed.

3.1 Dataset Used

We have used the openSLR dataset for the training purpose.The Open Speech and Language Resources (OpenSLR) is a platform that provides a collection of free, open-source speech and language datasets for research and development purposes. The datasets available on OpenSLR cover a wide range of languages, accents, and domains, including speech recognition, text-to-speech, speaker recognition, language identification, and more.

OpenSLR datasets are typically distributed as compressed archives containing audio files, text transcriptions, and metadata files. The metadata files contain information such as the file ID, speaker identity, and transcription text, which are necessary for training and evaluating speech and language models.

To use an OpenSLR dataset, researchers and developers can download the dataset from the OpenSLR website, extract the archive, and process the audio files and metadata files as needed for their specific task. The metadata files are typically in a standard format, such as TSV or JSON, making it easy to load them into a Python program for further processing.

OpenSLR datasets have been used in many speech and language research projects, including the development of automatic speech recognition systems, speaker recognition systems, and language identification systems. Using OpenSLR datasets can save researchers and developers time and resources by providing a pre-annotated dataset with high-quality audio files, and it can also help to promote the reproducibility of research by providing a common benchmark dataset.

3.2 Trying with LSTM as First Approach to Buid ASR

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that is commonly used in natural language processing (NLP) tasks such as speech recognition, language translation, and sentiment analysis. However, LSTM alone may not be the best choice for automatic speech recognition (ASR) tasks.

ASR involves converting spoken language into written text, and it requires modeling the temporal dynamics of speech signals. While LSTMs are able to model sequential data over long time periods, they are not optimized for processing the kind of acoustic signals that are used in speech recognition. This is because speech signals are highly complex and require a more specialized approach for accurate recognition.

Instead, ASR systems typically use specialized architectures such as deep neural networks (DNNs) or convolutional neural networks (CNNs) that are optimized for acoustic modeling. These models are often used in combination with LSTMs or other RNNs to integrate language models and context information into the ASR system.

Furthermore, ASR systems require a large amount of training data to accurately recognize speech, and LSTMs may not be able to efficiently process such large datasets due to their sequential nature. Therefore, ASR systems typically use more computationally efficient models that are better suited for processing large datasets.

The LSTM model which we built are not good enough to predict the outliers.It works like the speech recognition model rather than automatic speech recognition model.

In summary, while LSTMs are a powerful tool in natural language processing, they are not optimized for the specialized requirements of ASR, and other models are typically used for this task.

3.3 Why Transformers for ASR?

Transformers are an important architecture in automatic speech recognition (ASR) for several reasons:

1. Ability to model long-term dependencies: ASR models need to be able to capture long-term dependencies in speech signals to accurately transcribe speech. Traditional recurrent neural networks (RNNs) can struggle with this, especially when processing long sequences of speech data. Transformers, on the other hand, are specifically designed to model long-term dependencies and can be more effective in capturing the context of speech signals.

2. Attention mechanism: Transformers employ a self-attention mechanism, which allows the model to focus on relevant parts of the input at each time step. This can be particularly useful for ASR, as it allows the model to attend to the most important parts of the speech signal for transcription.

3. Effective for sequence-to-sequence tasks: ASR is a sequence-to-sequence task, where the input is a sequence of audio signals and the output is a sequence of words. Transformers have been shown to be highly effective for sequence-to-sequence tasks in natural language processing (NLP) and have also been successfully applied to ASR tasks.

4. Multilingual support: Transformers have been used in multilingual ASR to handle multiple languages within a single model. By leveraging the shared architecture of the transformer, a single model can be used to transcribe speech in multiple languages, which can be more efficient and cost-effective than building separate models for each language.

Overall, transformers are an important architecture in ASR due to their ability to model long-term dependencies, employ attention mechanisms, and handle sequence-to-sequence tasks effectively. These qualities make transformers an effective choice for ASR, and their ability to support multilingual models further enhances their value for real-world applications.Hence for the development of Malayalam ASR system we choose transformers for better results.

3.4 Creating JSON Files And Dataset Cleaning

As the first step we will be saving all the dataset as JSON files in seperate output folders.For each sample in the dataset, we will creates a dictionary with two keys: "path" and "sentence", which respectively contain the path to the sample and the sample text.Then saves the sample dictionary as a JSON file with a filename that includes the sample number using the json.dump() method.

One of the reasons to save a dataset as JSON files for Automatic Speech Recognition (ASR) is that JSON is a lightweight, widely used data format that can easily store structured data. JSON files can store the audio signal's metadata, such as speaker identity, transcription, and other relevant information, which is essential for ASR tasks.

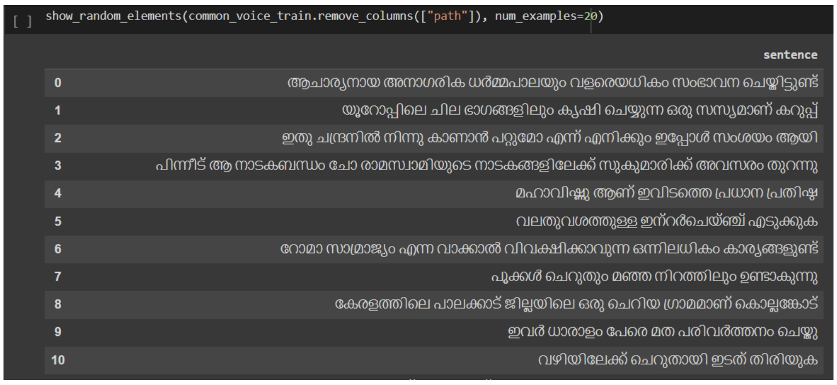
In ASR, JSON files can be used to store the audio signal's features such as spectrogram, Mel-frequency cepstral coefficients (MFCC), or filter banks. These features can be preprocessed and extracted from the audio signal and saved as JSON files, which can then be used as input to train an ASR model.

By storing the dataset as JSON files, it becomes easier to read, manipulate, and preprocess the data, which is essential for ASR pipelines. The JSON file format is also language-independent, meaning it can be used across different programming languages and platforms, making it a flexible option for storing and exchanging data between different ASR systems.

Dataset of males and females voices are seperatly created and stored in different folders.Later on these dataset of JSON files are loaded and splitted and combined to make a dataset of both male and female voices.The transcriptions are cleaned by removing special characters.

3.5 Extraction Of Unique Characters

Once the dataset are loaded and generated the JSON files we need to extract the unique characters from from the transcriptions and create a dictionary that maps each character to a unique integer index.Batch of examples from the datasets are taken and concatenates all the sentences into one long string.It then extracts a list of unique characters present in long string and

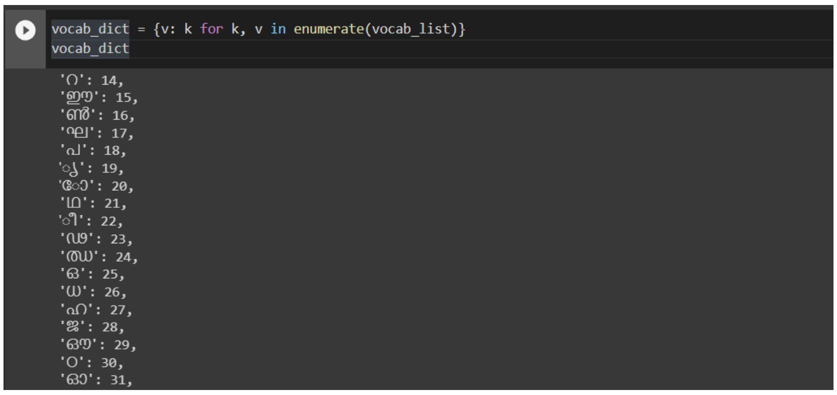


```
[ ] show_random_elements(common_voice_train.remove_columns(["path"]), num_examples=20)
```

	sentence
0	ആചാര്യനായ അനാഗരിക ധർമ്മപാലയും വളരെയധികം സംഭാവന ചെയ്തിട്ടുണ്ട്
1	യൂറോപ്പിലെ ചില ഭാഗങ്ങളിലും കൃഷി ചെയ്യുന്ന ഒരു സസ്യമാണ് കറുപ്പ്
2	ഇതു ചന്ദ്രനിൽ നിന്നു കാണാൻ പറ്റുമോ എന്ന് എനിക്കും ഇപ്പോൾ സംശയം ആയി
3	പിന്നീട് ആ നാടകബന്ധം ചോ രാമസ്വാമിയുടെ നാടകങ്ങളിലേക്ക് സുകുമാരിക്ക് അവസരം തുറന്നു
4	മഹാവിഷ്ണു ആണ് ഇവിടത്തെ പ്രധാന പ്രതിഷ്ഠ
5	വലതുവശത്തുള്ള ഇന്റർചെയ്ഞ്ച് എടുക്കുക
6	റോമാ സാമ്രാജ്യം എന്ന വാക്കാൽ വിവക്ഷിക്കാവുന്ന ഒന്നിലധികം കാര്യങ്ങളുണ്ട്
7	പൂക്കൾ ചെറുതും മഞ്ഞ നിറത്തിലും ഉണ്ടാകുന്നു
8	കേരളത്തിലെ പാലക്കാട് ജില്ലയിലെ ഒരു ചെറിയ ഗ്രാമമാണ് കൊല്ലങ്കോട്
9	ഇവർ ധാരാളം പേരെ മത പരിവർത്തനം ചെയ്തു
10	വഴിയിലേക്ക് ചെറുതായി ഇടത് തിരിയുക

Figure 3.1: Random Elements After Removing Special Characters

returns them as a dictionary with keys "vocab" and "long string".This process is applied to both traiing and testing dataset and the entire dataset processed in one batch. The resulting vocabulary lists are combined into a set and then back into a list, "vocablist", to ensure uniqueness. The "vocabdict" dictionary is then created by iterating over "vocablist" and assigning a unique integer index to each character.The resulting "vocabdict" can be used to convert the text data to a sequence of integers, which can then be fed into a neural network or other machine learning model for training or inference.Our vocabulary consists of 39 tokens, which means that the linear layer that we will add on top of the pretrained XLSR-Wav2Vec2 checkpoint will have an output dimension of 39.



```
vocab_dict = {v: k for k, v in enumerate(vocab_list)}
vocab_dict
```

```
'റ': 14,
'ഈ': 15,
'ൺ': 16,
'ഘ': 17,
'ഫ': 18,
'്യ': 19,
'ോ': 20,
'ഥ': 21,
'ീ': 22,
'ഢ': 23,
'ഝ': 24,
'ഒ': 25,
'ധ': 26,
'ഹ': 27,
'ജ': 28,
'ഔ': 29,
'ഠ': 30,
'ഓ': 31,
```

Figure 3.2: Unique Integer Index to Each Character

In Automatic Speech Recognition (ASR), text data is often represented as a sequence of characters, which can be used to transcribe an audio signal into text. Before feeding the text data into an ASR model, it is often necessary to encode the characters as integers, using a process called character-level tokenization. This is because neural networks and other machine learning models can only process numerical data, not raw text.

In order to encode the characters as integers, each character in the text data is mapped to a unique integer index, which can then be used to represent the character in a numerical format. This requires a mapping between the characters and their integer indices, which can be stored as a dictionary.However, before creating the character-to-index mapping, we need to determine the set of unique characters that appear in the text data. This is because the size of the character-to-index mapping will depend on the number of unique characters in the text data. By extracting all unique characters from the dataset, we ensure that the size of the character-to-index mapping is as small as possible, which in turn reduces the number of parameters in the ASR model and can improve its performance. Additionally, encoding the characters as integers using a small set of indices, rather than a large number of indices, can also reduce the amount of memory required to store the text data during training and inference.

3.6 Tokenizer and Feature Extractor

ASR models transcribe speech to text, which means that we both need a feature extractor that processes the speech signal to the model's input format, e.g. a feature vector, and a tokenizer that processes the model's output format to text.

In Transformers, the XLSR-Wav2Vec2 model is thus accompanied by both a tokenizer, called Wav2Vec2CTCTokenizer, and a feature extractor, called Wav2Vec2FeatureExtractor.

3.6.1 Create Wav2Vec2CTCTokenizer

The pretrained Wav2Vec2 checkpoint maps the speech signal to a sequence of context representations as illustrated in the figure above. A fine-tuned XLSR-Wav2Vec2 checkpoint needs to map this sequence of context representations to its corresponding transcription so that a linear layer has to be added on top of the transformer block (shown in yellow). This linear layer is used to classifies each context representation to a token class analogous how, e.g., after pretraining a linear layer is added on top of BERT's embeddings for further classification.The output size of this layer corresponds to the number of tokens in the vocabulary, which does not depend on XLSR-Wav2Vec2's pretraining task, but only on the labeled dataset used for fine-tuning.

In Connectionist Temporal Classification(CTC), it is common to classify speech chunks into letters, so we have done the same here. We have extracted all distinct letters of the training and test data and built our vocabulary from this set of letters.

3.6.2 Create XLSR-Wav2Vec2 Feature Extractor

Speech is a continuous signal and to be treated by computers, it first has to be discretized, which is usually called sampling. The sampling rate hereby plays an important role in that it defines how many data points of the speech signal are measured per second. Therefore, sampling with a higher sampling rate results in a better approximation of the real speech signal but also necessitates more values per second. A pretrained checkpoint expects its input data to have been sampled more or less from the same distribution as the data it was trained on. The same speech signals sampled at two different rates have a very different distribution, e.g., doubling the sampling rate results in data points being twice as long. Thus, before fine-tuning a pretrained checkpoint of an ASR model, it is crucial to verify that the sampling rate of the data that was used to pretrain the model matches the sampling rate of the dataset used to fine-tune the model. XLSR-Wav2Vec2 was pretrained on the audio data of Babel, Multilingual LibriSpeech (MLS), and Common Voice. Most of those datasets were sampled at 16kHz, so that Common Voice, sampled at 48kHz, has to be downsampled to 16kHz for training. Therefore, we will have to downsample our fine-tuning data to 16kHz in the following. A XLSR-Wav2Vec2 feature extractor object requires the following parameters to be instantiated:

1. feature_size: Speech models take a sequence of feature vectors as an input. While the length of this sequence obviously varies, the feature size should not. In the case of Wav2Vec2, the feature size is 1 because the model was trained on the raw speech signal.
2. sampling_rate: The sampling rate at which the model is trained on.
3. padding_value: For batched inference, shorter inputs need to be padded with a specific value
4. do_normalize: Whether the input should be zero-mean-unit-variance normalized or not. Usually, speech models perform better when normalizing the input
5. return_attention_mask: Whether the model should make use of an attention_mask for batched inference. In general, XLSRWav2Vec2 models should always make use of the attention_mask.

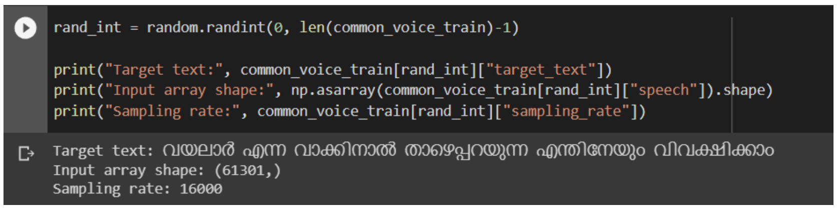
To make the usage of XLSR-Wav2Vec2 as user-friendly as possible, the feature extractor and tokenizer are wrapped into a single Wav2Vec2Processor class so that one only needs a model and processor object.

3.7 Preprocess Data

The preprocessing steps are critical for ensuring the quality and reliability of the dataset used to train an ASR model, which can ultimately have a significant impact on the accuracy of the ASR system.

XLSR-Wav2Vec2 expects the audio file in the format of a 1-dimensional array, so in the first step, let's load all audio files into the dataset object.The audio file are saved in the .mp3 format. The .mp3 format is usually not the easiest format to deal with. We found that the torchaudio library works best for reading in .mp3 data.An audio file usually stores both its values and the sampling rate with which the speech signal was digitalized.Both are to be stored in dataset.

Librosa library is used to downsample the audio files to 16KHz.Downsampled audiofiles are then reviewed for to ensure all the audio files are correctly loaded.The speakers change along with their speaking rate, accent, and background environment, etc. Overall, the recordings sound acceptably clear though, which is to be expected from a crowd-sourced read speech corpus.



```
rand_int = random.randint(0, len(common_voice_train)-1)

print("Target text:", common_voice_train[rand_int]["target_text"])
print("Input array shape:", np.asarray(common_voice_train[rand_int]["speech"]).shape)
print("Sampling rate:", common_voice_train[rand_int]["sampling_rate"])
```

```
Target text: വയലാർ എന്ന വാക്കിനാൽ താഴെപ്പറയുന്ന എന്തിനേയും വിവക്ഷിക്കാം
Input array shape: (61301,)
Sampling rate: 16000
```

Figure 3.3: Shape of Speech Input

The data is a 1-dimensional array, the sampling rate always corresponds to 16kHz, and the target text is normalized.Hence, we can process the dataset to the format expected by the model for training.First, we check that the data samples have the same sampling rate of 16kHz. Second, we extract the input_values from the loaded audio file. In our case, this includes only normalization, but for other speech models, this step could correspond to extracting, e.g. Log-Mel features. Third, we encode the transcriptions to label ids.

3.8 Training

The data is processed so that we are ready to start setting up the training pipeline.In addition to loading the pre-trained model, we also specifies several hyperparameters that are used during training. These hyperparameters include:

1. attention_dropout: The probability of dropping out attention weights during training.
2. hidden_dropout: The probability of dropping out hidden states during training.
3. feat_proj_dropout: The probability of dropping out feature projection weights during training.
4. mask_time_prob: The probability of masking a timestep in the input sequence during training.
5. layerdrop: The probability of dropping out entire layers during training.
6. gradient_checkpointing: Whether to use gradient checkpointing to save memory during training.
7. ctc_loss_reduction: The reduction method to use when computing the CTC loss (in this case, taking the mean).
8. pad_token_id: The token ID used to pad sequences to a fixed length during training.
9. vocab_size: The size of the vocabulary used by the model's tokenizer.

We will make use of hugging face's Trainer for which we essentially need to do the following:

- Define a data collator: In contrast to most NLP models, XLSR-Wav2Vec2 has a much larger input length than output length. E.g., a sample of input length 50000 has an output length of no more than 100. Given the large input sizes, it is much more efficient to pad the training batches dynamically meaning that all training samples should only be padded to the longest sample in their batch and not the overall longest sample. Therefore, fine-tuning XLSR-Wav2Vec2 requires a special padding data collator.

- Evaluation metric: During training, the model should be evaluated on the word error rate. We should define a function to compute metrics accordingly.
- Load a pretrained checkpoint: We need to load a pretrained checkpoint and configure it correctly for training.
- Define the training configuration.

After having fine-tuned the model, we will correctly evaluate it on the test data and verify that it has indeed learned to correctly transcribe speech.Data collator treats the input_values and labels differently and thus applies to separate padding functions on them (again making use of XLSRWav2Vec2's context manager). This is necessary because in speech input and output are of different modalities meaning that they should not be treated by the same padding function. Analogous to the common data collators, the padding tokens in the labels with -100 so that those tokens are not taken into account when computing the loss.To allow models to become independent of the speaker rate, in CTC, consecutive tokens that are identical are simply grouped as a single token. However, the encoded labels should not be grouped when decoding since they don't correspond to the predicted tokens of the model.

Chapter 4

Results and Evaluation

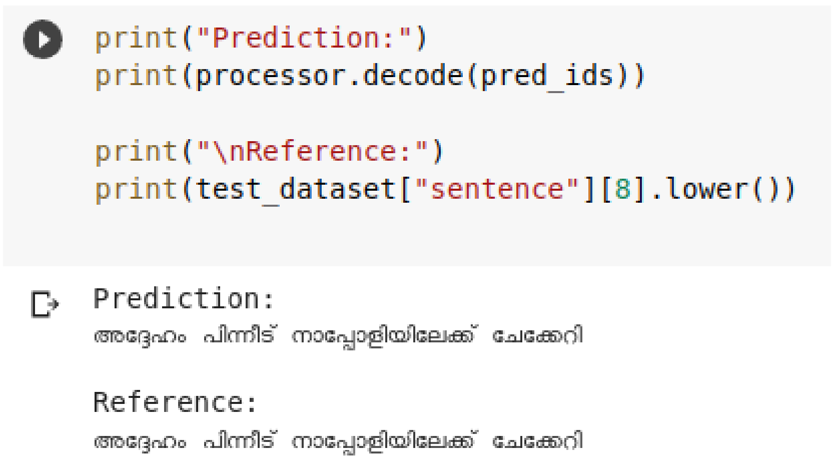
4.1 Results

The performance of an Automatic Speech Recognition (ASR) system for Malayalam depends on several factors, including the quality and quantity of training data, the complexity of the language's phonetic system, and the design and implementation of the ASR system.The recorded audio files are converted into numerical array and fed into model for prediction.We loads each audio file using torchaudio, resamples it if necessary, and converts it to a NumPy array.

It also removes certain characters from the transcription and returns a dictionary containing the audio data, the transcription, and the sampling rate.Then the batch of preprocessed audio data and transcriptions and passes them through the Wav2Vec2 model for prediction.As a result a dictionary containing the predictions and the original transcriptions are returned.

The predictions and transcriptions are then passed to the "wer" metric using the "compute" method to compute the Word Error Rate (WER) for the ASR system on the test dataset. The model loaded using "Wav2Vec2ForCTC.from_pretrained" method is then used to predict the transcription of the audio file, and the predicted transcription is returned as a NumPy array of integer ids.

We got fairly good predictions of transcriptions of audio files as given below:
Prediction: The predicted sentence by our model.
Reference: The actual transcription of the audio file.



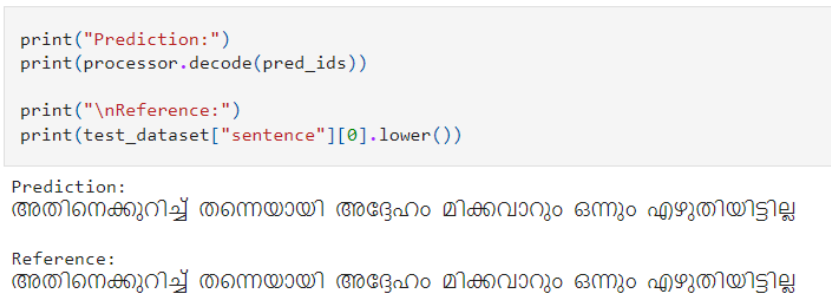
```
print("Prediction:")
print(processor.decode(pred_ids))

print("\nReference:")
print(test_dataset["sentence"][8].lower())
```

```
Prediction:
അദ്ദേഹം പിന്നീട് നാപ്പോളിയിലേക്ക് ചേക്കേറി

Reference:
അദ്ദേഹം പിന്നീട് നാപ്പോളിയിലേക്ക് ചേക്കേറി
```

Figure 4.1: Example 1:prediction and actual transcriptions



```
print("Prediction:")
print(processor.decode(pred_ids))

print("\nReference:")
print(test_dataset["sentence"][0].lower())
```

```
Prediction:
അതിനെക്കുറിച്ച് തന്നെയായി അദ്ദേഹം മിക്കവാറും ഒന്നും എഴുതിയിട്ടില്ല

Reference:
അതിനെക്കുറിച്ച് തന്നെയായി അദ്ദേഹം മിക്കവാറും ഒന്നും എഴുതിയിട്ടില്ല
```

Figure 4.2: Example 2:prediction and actual transcriptions

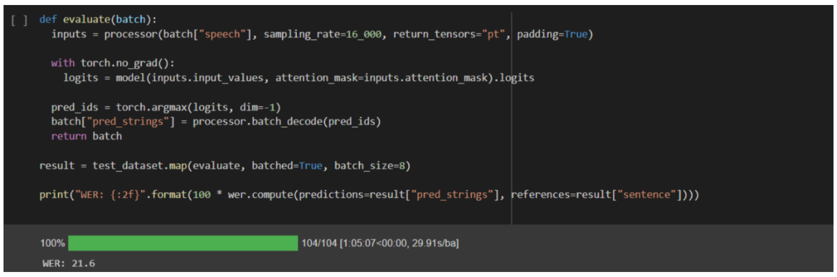
4.2 Evaluation Metric

Word error rate (WER) is a common metric of the performance of an automatic speech recognition system. The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one). The WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level. The WER is a valuable tool for comparing different systems as well as for evaluating improvements within one system. This kind of measurement, however, provides no details on the nature of translation errors and further work is therefore required to identify the main source(s) of error and to focus any research effort.

This problem is solved by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment.

Examination of this issue is seen through a theory called the power law that states the correlation between perplexity and word error rate.

We got a Word Error Rate(WER) of 21.6.



```
[ ] def evaluate(batch):
      inputs = processor(batch["speech"], sampling_rate=16_000, return_tensors="pt", padding=True)

      with torch.no_grad():
        logits = model(inputs.input_values, attention_mask=inputs.attention_mask).logits

      pred_ids = torch.argmax(logits, dim=-1)
      batch["pred_strings"] = processor.batch_decode(pred_ids)
      return batch

    result = test_dataset.map(evaluate, batched=True, batch_size=8)

    print("WER: {:2f}".format(100 * wer.compute(predictions=result["pred_strings"], references=result["sentence"])))

    100% 104/104 [1:05:07<00:00, 29.91s/ba]
    WER: 21.6
```

Figure 4.3: Word Error Rate Evaluation

Word error rate can then be computed as:

$$WER = (S + D + I)/N = (S + D + I)/(S + D + C) \tag{4.1}$$

where,S is the number of substitutions, D is the number of deletions, I is the number of insertions, C is the number of correct words, N is the number of words in the reference (N=S+D+C).

Some of the combined words are split and predicted as separate words.Even though such words does not change the meaning of the words but it counted in WER, hence increasing the error rate.Which means if s single word is split into two, then it will be considered as insertion of two new words and deletion of existing word from original transcription thus increasing the error rate.

Chapter 5

Conclusion

Development of a tokenizer and feature extractor, resulted in improved accuracy and efficiency in Malayalam speech recognition. This study demonstrates the potential of the XSLR-Wav2Vec2 model for multilingual speech recognition and opens up new avenues for further research and development in the field of ASR. The results suggest that the XSLR-Wav2Vec2 model has the potential to be applied to other low-resource languages for speech recognition, providing a scalable and flexible solution for the global ASR community.

In conclusion, the development of Automatic Speech Recognition (ASR) systems for Malayalam has been an active area of research for many years. Recent advancements in deep learning and natural language processing techniques have led to significant improvements in the accuracy and robustness of Malayalam ASR systems.

One promising approach for Malayalam ASR is the use of pretrained models, such as the wav2vec xsrl model, which have been trained on large amounts of speech data from multiple languages. This approach has been shown to be effective in other languages, and recent studies have demonstrated the potential of this approach for Malayalam ASR as well.

By leveraging the wav2vec xsrl pretrained model, researchers can significantly reduce the amount of labeled speech data required to train an accurate ASR model for Malayalam. This can be particularly beneficial in low-resource settings where labeled data is scarce.

However, further research is still needed to optimize the performance of the wav2vec xsrl model for Malayalam ASR and to explore its potential in different domains and applications. With ongoing research, it is likely that we will see further improvements in the accuracy and robustness of Malayalam ASR systems in the future.

REFERENCES

[1] Babu, L. B., George, A., Sreelakshmi, K. R., & Mary, L. (2018). Continuous speech recognition system for malayalam language using kaldi. In *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*, (pp. 1–4).

[2] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, *33*, 12449–12460.

[3] Kallooravi Thandil, R., K.P., M., & Kollathodi, R. (2019). Automatic speech recognition system for utterances in malayalam language. *Malaya Journal of Matematik*, *S*, 560–565.

[4] Moncy, A. M., M., A., Jasmin, H., & Rajan, R. (2020). Automatic speech recognition in malayalam using dnn-based acoustic modelling. In *2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, (pp. 170–174).