# Speech Corpus Generation For Low Resource Language (Pashto)

**By**

**Muhammad Shoaib**

**Supervisor: Dr. Shibli Nisar**

A thesis submitted to the faculty of the Computer Science Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Software Engineering

**December 2022**

# THESIS ACCEPTANCE CERTIFICATE

Certified that the final copy of the MS/MPhil thesis written by Mr. **Maj Muhammad Shoaib,** Registration No **NUST00000359455**, of **Military College of Signals,** has been vetted by the undersigned, found complete in all respect as per NUST Statutes/Regulations, is free of plagiarism, errors, and mistakes and is accepted as partial, fulfillment for the award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the student have been also incorporated in the said thesis.

Signature: ___________________________

Name of Supervisor: Asst Prof Shibli Nisar, PhD

Date: ________________________________

Signature (HoD): _______________________

Date: ________________________________

Signature (Dean/Principal): ______________

Date: ________________________________

# Abstract

Meta AI’s new unsupervised speech recognition framework (wav2vec variants) is the latest development of several years of work in speech recognition models, data sets, and training techniques. The wav2vec model has changed the way traditional ASR worked, now a few hours of spoken data is required to obtain transcribed speech. Despite this, over 6000 languages couldn't exploit the opportunity because they lack the required speech data corpus. The corpus should contain 4-5 hours of speech data on average, which is a challenge, especially for a low-resource language. To deal with the challenge the current approach is to manually record speech and then transcribe it. Such an approach is resource intensive and costly. On the internet, there is a wealth of speech data. To capitalize on such data, we will use an automated speech utilization process instead of manual recording. In our thesis, we have proposed a model that automatically fetches audio data from free video/audio sharing websites and segments them to produce desired length audio frames. The proposed model is generic and can be implemented for any low-resource language. Furthermore, using the proposed pipeline we generated speech data for the Pashto language.

# Declaration

I hereby declare that no portion of the work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere

# Dedication

“In the name of Allah, the most Beneficent, the most Merciful"

I dedicate this thesis to my family, teachers, and acquaintances who supported me

each step of the way

# Acknowledgments

All praises to Allah for the strengths and His blessing in completing this thesis.

I would like to convey my gratitude to my supervisor, Asst Prof. Shibli Nisar, Ph.D., for his supervision and constant support. His invaluable help of constructive comments and suggestions throughout the experimental and thesis work are major contributions to the success of this research. Also, I would thank my committee members; Asst Prof. Imran Qureshi, Ph.D., and Asst Prof. Yawar Abbas Bangash, Ph.D. for their support and knowledge extended during the formulation of this research study.

Last, but not least, I am highly thankful to my mother (late), and companion. They have always stood by my aspirations and have been a great source of motivation for me. I would like to thank them for all their care, love, and support through my times of stress and excitement.

# Table of Contents

# List of Figures

# List of Tables

# THESIS ACCEPTANCE CERTIFICATE

Certified that the final copy of the MS/MPhil thesis written by Mr. **Maj Muhammad Shoaib,** Registration No **NUST00000359455**, of **Military College of Signals,** has been vetted by the undersigned, found complete in all respect as per NUST Statutes/Regulations, is free of plagiarism, errors, and mistakes and is accepted as partial, fulfillment for the award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the student have been also incorporated in the said thesis.

Signature: ___________________________

Name of Supervisor: Asst Prof Shibli Nisar, PhD

Date: _______________________________

Signature (HoD): _______________________

Date: _______________________________

Signature (Dean/Principal): ______________

Date: _______________________________

# Abstract

Meta AI’s new unsupervised speech recognition framework (wav2vec variants) is the latest development of several years of work in speech recognition models, data sets, and training techniques. The wav2vec model has changed the way traditional ASR worked, now a few hours of spoken data is required to obtain transcribed speech. Despite this, over 6000 languages couldn't exploit the opportunity because they lack the required speech data corpus. The corpus should contain 4-5 hours of speech data on average, which is a challenge, especially for a low-resource language. To deal with the challenge the current approach is to manually record speech and then transcribe it. Such an approach is resource intensive and costly. On the internet, there is a wealth of speech data. To capitalize on such data, we will use an automated speech utilization process instead of manual recording. In our thesis, we have proposed a model that automatically fetches audio data from free video/audio sharing websites and segments them to produce desired length audio frames. The proposed model is generic and can be implemented for any low-resource language. Furthermore, using the proposed pipeline we generated speech data for the Pashto language.

# Declaration

I hereby declare that no portion of the work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere

# Dedication

“In the name of Allah, the most Beneficent, the most Merciful"

I dedicate this thesis to my family, teachers, and acquaintances who supported me

each step of the way

# Acknowledgments

All praises to Allah for the strengths and His blessing in completing this thesis.

I would like to convey my gratitude to my supervisor, Asst Prof. Shibli Nisar, Ph.D., for his supervision and constant support. His invaluable help of constructive comments and suggestions throughout the experimental and thesis work are major contributions to the success of this research. Also, I would thank my committee members; Asst Prof. Imran Qureshi, Ph.D., and Asst Prof. Yawar Abbas Bangash, Ph.D. for their support and knowledge extended during the formulation of this research study.

Last, but not least, I am highly thankful to my mother (late), and companion. They have always stood by my aspirations and have been a great source of motivation for me. I would like to thank them for all their care, love, and support through my times of stress and excitement.

# Table of Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

## 1.1 Overview

Data plays a critical role in the building of language technologies. Language models are frequently evaluated using static corpora. The characteristics of these datasets fundamentally influence the model's behavior. The model can be divided into two categories: Natural Language Processing (NLP) and Automatic Speech Recognition (ASR). The ASR field has become an important research area for human-to-machine communication. The core function of an ASR system is to convert spoken words into corresponding written text automatically [1]. Earlier ASR technology relied on manual feature extraction and traditional Machine Learning (ML) techniques like Gaussian Mixture Models (GMM), Dynamic Time Warping (DTW), and Hidden Markov Models (HMM). These Models were a statistical approach, but they still fall short in terms of performance accuracy [2,3]. In the 1990s, the research tendency began to shift toward the use of artificial neural networks (ANNs) in speech applications. The ability to detect and learn complex structures from a vast amount of data is what enabled an ANN-based technique to achieve high performance in ASR systems. However, ANN required a high amount of data to produce accurate results [4,5,6]. Since the year 2000, the growing popularity of Deep Learning (DL) has shifted the trend from traditional ASR to E-2-E (End to End) ASR technology [7,8,9,10]. E2E ASR systems decode input speech data into an output data sequence using a single network [11,12,13,14,15]. As shown in Figures 1.1 and 1.2, the traditional approach's pronunciation, acoustic, and language models are now trained in a single system.

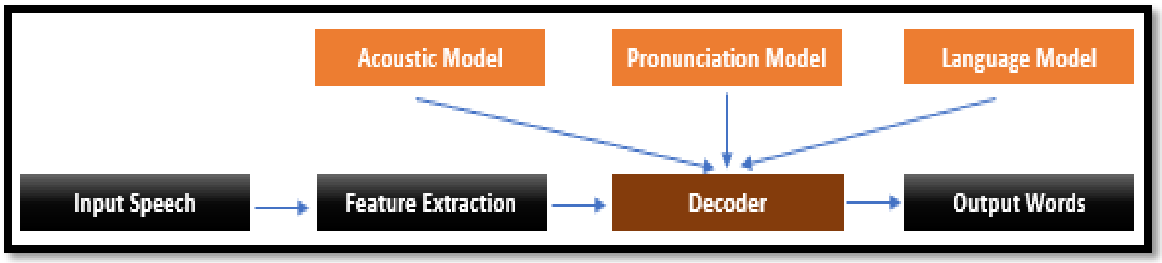


Figure 1.1: Traditional ASR Model

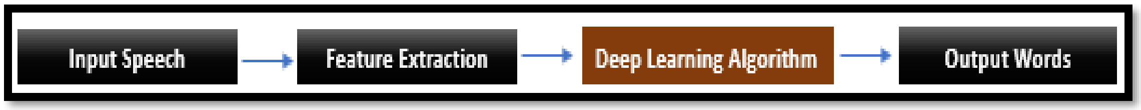


Figure 1.2: E2E ASR Model

Deep Learning models [16,17,18,19] have achieved great performance in speech recognition tasks and need to be trained on a massive amount of human-labeled voice data, which often comes with a high cost in terms of time and resources. The majority of data available for practical use is extremely unorganized, and labels are hard to obtain. Furthermore, such ASR systems are usually accessible in English and a few other resource-rich languages. Therefore, for a large majority of the 7,000 languages spoken worldwide, simply supervised teaching is impracticable [20]. Consequently, Artificial Intelligence (AI) researchers started exploring unsupervised learning approaches for developing less data-intensive and better results-giving ASR systems [21,22,23]. Self-training methods were explored [24,25,26] that showed promising results [17,27,28,29,30,31]. The idea is to apply a model's learned intelligence from a larger dataset to a target task with limited data. As shown in Figures 1.3 and 1.4, the model is trained using a self-supervision technique on some contextual data as a pre-target task, and then the learned intelligence is fine-tuned using supervised learning on a target task.

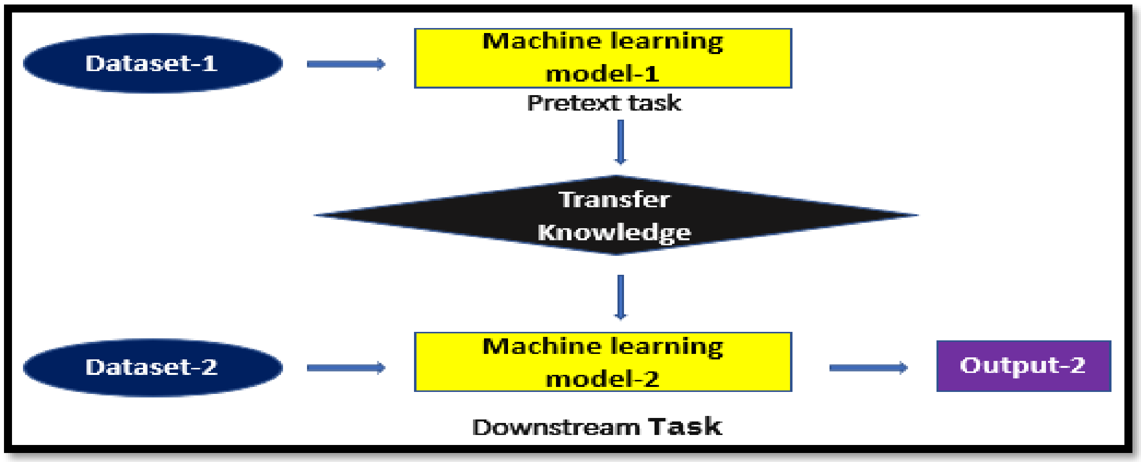


Figure 1.3: Traditional Machine learning

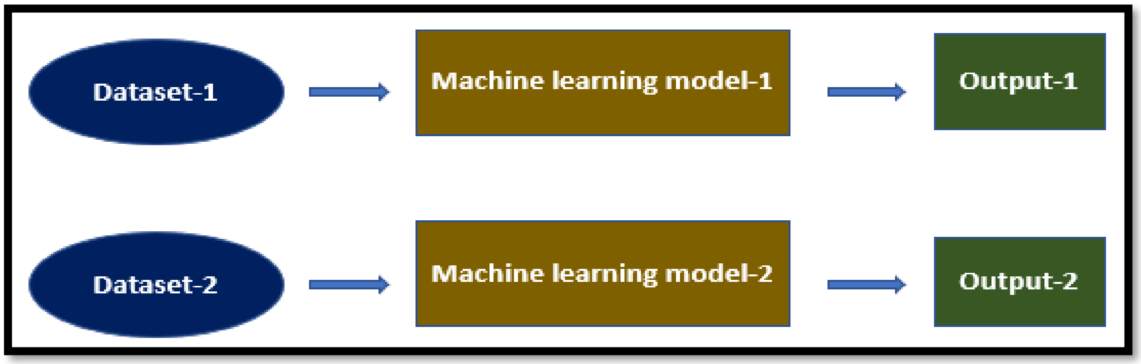


Figure 1.4: Transfer Learning

The researcher's interest in the field of self-supervised learning started to expand in 2016 onwards as shown in Figure 1.5. Facebook AI pundits came up with the idea of wav2vec and matured it into wav2vec2.0 with the addition of a transformer network to existing convolutional networks. The encoder accepts raw audio as input, processes it in layers, and outputs latent speech representations for a predetermined number of time steps. When they utilized the whole Libri Speech dataset (English) for fine-tuning, the test-clean subset had a Word Error Rate (WER) of 1.8%, whereas the test-other had a WER of 3.3%. When only 10 minutes of labeled training data were used, the results of Libri Speech were 4.8% on a test clean and 8.2% on test-other subsets [32].

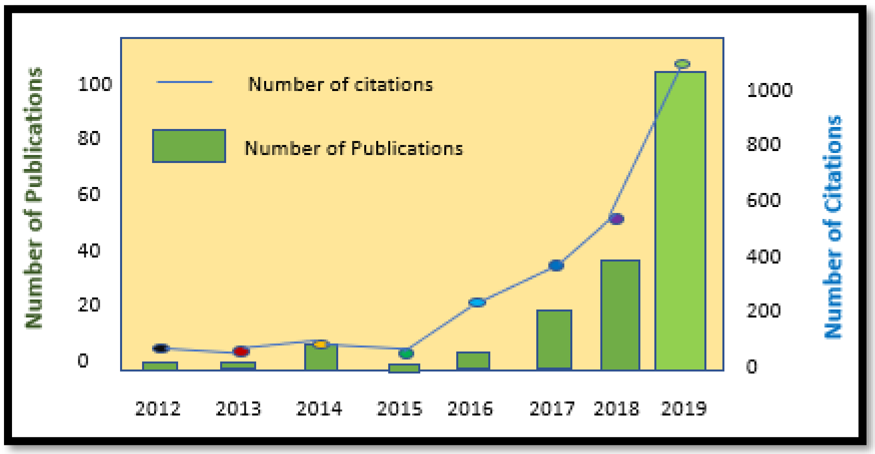


Figure 1.5: Statistics of “Self-Supervised Learning” Term Appearing in Papers

Wav2vec isn't the only self-supervised method available in this domain. TRILL (TRIpLet Loss network) is an unsupervised embedding for speech developed by Google AI using the massive AudioSet dataset containing over 2 million audio clips [33]. Soon researchers started testing and improvising ASR models with a self-supervised concept utilizing wav2vec and a related framework. A trend of developing a series of unlabeled speech repositories has popped up for different languages like Mozilla Common Voice [34], Deep Speech, LibriVox, VoxPopuli, Chinese [35,36], and many more. The wav2vec framework and other alike models require a few hours of a transcribed dataset (normally 4-5 hours) to test an ASR system but producing such data is one of the main challenges, especially in the case of low-resource languages. Speech data can be either recorded in an environment under certain standards or fetched from online resources. Recording speech is expensive in terms of time, cost, and other resources. Alternatively, there is an enormous spoken data available over the internet on different audio or video platforms like YouTube and Facebook, etc. Such data can be extracted with less or almost no cost and provide an opportunity for potential speech corpora. Now, the question is how to extract and test it against prevailing ASR technologies. Hence, compiling a few hours of transcribed speech data is not an easy task and we again face a challenge. This challenge presents a viable area of research: How to preserve online speech data and use it for state-of-the-art ASR applications? We will try to find our way out in the preceding chapters for the low-resource language “Pashto”.

## 1.2 Motivation

Ethnologue reports that people speak almost 7117 languages across the globe. Unfortunately, this number is decreasing, with 90% of these languages having fewer than 100,000 speakers. Over 2000 languages are spoken in Africa alone. Languages spoken in Central Asia, Northeast India, Afghanistan, and Iran region have limited resources. Low-resource languages spoken in Pakistan include Sindhi, Baluchi, Kashmiri, Gojri, Hindko, Siraiki, Dari, Gilgiti, Balti, Pashto, etc. Most of these languages lack lexicons, phonetics, and language models [37]. Producing Speech data is a major problem in such languages. Our motivation is twofold: firstly, there is a need for a low-cost, less resource-intensive, and easy-to-use speech data collection technique. Secondly, we are interested in generating speech data for the Pashto language to build ASR corpora and fine-tune cutting-edge models.

Pashto is classified as an Indo-European language by worldData.info, with 50.4 million speakers worldwide. It is primarily spoken in Afghanistan and northwest regions of Pakistan. It is recognized as one of Afghanistan's official national languages. There is no API support available in the Google Cloud Speech-to-Text domain [“Google Cloud Speech-to-Text” and “Google Cloud Translation AI”]. Mozilla Common voice has not published the Pashto language corpus yet [commonvoice.mozilla]. Pashto has a limited web presence, digital resources, and linguistic expertise available. Currently, there are no sufficient acoustic or standard text corpora, pronunciation lexica, and data repositories held.

## 1.3 Problem Statement

The existing state of ASR work is not satisfactory for low-resource languages including Pashto. As a result, a sizable proportion of the global population, including the Pashtun community, is unable to benefit from the boom in speech digitization. To preserve the language and capitalized on speech technology digitization it is a dire need to create data repositories and test these with the latest ASR trends. The traditional ASR approach is a human-labeled data-intensive, costly, and impractical approach for low-resource languages. Meta AI wav2vec [32] self-supervised ASR framework provide an opportunity; it can be used on a few hours of a transcribed dataset. But Building a few hours of speech corpus is a challenging task. As voice recording is costly, speech data available online in the public domain can be utilized but how to collect such information again is a problem. Now we need a data-gathering mechanism, that is free and fast enough to scrap speech online. Once such a mechanism is built then a speech corpus needs to be developed utilizing online voice data. YouTube is one such platform that provides a huge amount of speech data in the public domain. Thus, a handsome amount of research can activate native speakers to develop advanced ASR systems for low-resource languages.

## 1.4 Objectives

The following are the thesis's primary goals: -

- To develop an automated online speech collection technique for a low-resource language.
- To produce a Pashto speech dataset of 5 to 6 hours in line with the wav2vec framework for speech transcription.

## 1.5 Thesis Contribution

After conducting an extensive literature review, so far, no such work of speech data collection and ASR Corpus development can be found for a low-resource language generally and Pashto particularly. The mechanism proposed in this paper has not been used for any other project. An effort has been put forward to exploit the research gap identified. The following are the major contributions of this work: -

- This work can activate further research in the field of ASR for Pashto and other alike languages.
- The proposed pipeline is generic, any language can use it to generate speech data, especially a low-resource language.
- The proposed speech collection pipeline can be used for any kind of audio or video data available online in the public domain.

- The proposed data collection technique can be modified and improvised according to requirements.
- The speech data acquired for Pashto can be used to test the latest ASR models.
- The Pashto speech dataset can provide the basis for further larger repository building.
- The Pashto speech dataset can be used for further transcription and fine-tuning of advanced ASR models.
- Finally, such corpora have multipurpose implications in the areas of health, education, industries, and law enforcement sectors.

## 1.6 Thesis Organization

The remainder of the thesis is organized as follows: -

- Chapter 2 gives an extensive literature review describing the dynamics of Low resource Language (LRL) followed by existing research on the subject.
- Chapter 3 contains an analysis of Pashto speech generated with the help of the proposed model.
- Chapter 4 covers the generic concept of how video streaming works and how it affects downloading and extraction of audio data from video-sharing platforms. A detailed description of YouTube's mechanism of video streaming is given.
- Chapter 5 describes the methodology adopted to develop the model and its use to generate speech data for Pashto.
- Chapter 6 explains the model evaluation mechanism including results and discussions.
- Chapter 8 marks the end of the document presenting future work and a conclusion.

# Chapter 2

# Literature Review

The first part of this chapter introduces the concept of a low-resource language, its challenges, and the implications for a resource-scarce language in the field of ASR. The second part of the chapter provides an overview of previous research available. The chapter is summarized in the following sequence:

## 2.1 Dynamics of a Low Resource Language (LRL)

### 2.1.1 What is a Low resource language (LRL)?

The building of language technologies has a significant impact on the lives of billions of people worldwide. We use important techniques like computational linguistics, speech recognition, and artificial intelligence to exploit the voids of a language. Any language can be either rich in digital resources or low. The phrase "under-resourced languages" was first used in 2003 & 2004 [33,34]. The synonyms for the same concept can be found as low-density languages, resource-poor languages, low-data languages, less-resourced languages, and under-resourced languages. Richness in resources can be viewed in terms of the following aspects:

- Massive amounts of raw text from various subjects and resources.
- Resources for lexicons, phonemes, syntax, and semantics.
- Task-specific resources (for example, parts-of-speech tags, named entities, and so on).

### 2.1.2 ASR Challenges

The resources available to train speech recognizer tasks for a low-resource language are severely limited. There can be multiple reasons: -

- Firstly, a language with limited resources needs strategies that go well beyond simple model retraining. Resource scarcity necessitates novel data-gathering techniques such as crowdsourcing [38] or models that allow information sharing among different languages [39,40].

- Secondly, several cultural and social variables connected to the low-resource language environment pose lingering problems. Some of the important variables include numerous regional dialects, code-switching or codemixing occurrences, and the prevalence of many non-native speakers.
- Thirdly, the gap between language speakers and technology developers must be bridged. Finding native speakers with the technical skills required to train ASR systems in their language is nearly impossible.
- Finally, research describing under-resourced languages is limited and sometimes poorly addressed in linguistic literature. To put such systems in place, resources and knowledge from similar languages must be borrowed, necessitating the assistance of dialectologists, phoneticians, and other relevant experts. All these factors present a multidisciplinary challenge for researchers working on ASR technology for under-resourced languages.

### 2.1.3 Implications for a Low-Resource Language

Keeping ASR trends in view, the commercial market is driven by several high-resource languages, especially English, French, Spanish, Mandarin, Chinese and Japanese. As a result, speakers of low-resource languages run a risk of becoming socioeconomically isolated due to the unavailability of digital resources. As speech is the fastest mode of data processing, therefore, the rescue can be seen in exploiting the latest trends in speech technology. Low-resource languages may benefit from ASR tools that require less amount of speech for model training.

In the case of Pashto, despite having abundant speakers, it is one of the low-resource languages for multiple reasons. There is a lack of research work in language technology in the areas of Natural Language Processing and voice recognition. The language raw text, lexical, syntactic, semantic dictionaries, dependency tree corpora, and semantic databases are limited. One can hardly find any funded project to produce some sort of Pashto text and speech data. The current small amount of work has a high word error rate and lacks study to improve its accuracy. In such circumstances speech models like wav2vec selfsupervise framework present an opportunity. Such models need a few hours of data to finetune on transcribe dataset. This will help a low-resource language such as Pashto in developing a data preservation and repository system for speech recognition tasks.

## 2.2 Available Research

The generation of an ASR corpus has been essential to the advancement of speech technology. Several initiatives have been launched over the years. In 1993 TIMIT (Texas Instrument/Massachusetts Institute of Technology) database was launched [41]. Wall Street Journal [42] and Switchboard [43] databases have enabled the growth of large vocabulary continuous speech recognition systems. However, the high cost of collecting, maintaining, and disseminating such good-quality datasets stifles independent speech technology research. The realm of speech recognition has evolved from statistical to deep learning methods and self-supervised learning models. As a result, the cost of developing and maintaining data repositories has decreased, and researchers' interest in speech recognition has grown over time. The research already carried out in the ASR field can be described in the following subsections: -

### 2.2.1 Developed Languages

In the case of developed languages like English, Chinese, and Japanese, etc., speech recognition research has progressed significantly. These languages have established advanced repositories to test and train ASR's latest models. LibriSpeech [44] is a 1,000-hour English speech dataset collected from audiobooks. GigaSpeech [45] is another 10,000-hour English speech dataset. Mozilla Common Voice [46] is a 10,000-hour crowdsourced read speech corpus in a variety of languages. MLS [47] is a 50,000-hour speech corpus derived from audiobooks. Aishell-1 [48] is an open-source mandarin speech corpus and "Corpus of Spontaneous Japanese" (CSJ) is a Japanese database.

### 2.2.2 Emerging Languages

The speech recognition research is progressing for emerging languages like Turkish, Hindi, Urdu, and Persian.  The tendency of publishing speech datasets is on the rise. The 90 hours of Turkish Speech Corpus [49] extracted from Turkish movies were used in speech recognition activities. One of the initiating works for the Hindi language is an annotated and time-aligned speech database consisting of a total of 500 sentences uttered by 50 speakers [semanticscholar.org]. The most recent effort is 1111 hours of Hindi ASR Challenge 2022 named Gramvaani data [openslr.org]. It is based on sharing spontaneous telephone speech recordings in regional variations of Hindi from an enterprise Gram Vaani. A corpus for 0 to 9 Urdu digits and another dataset for 250 Urdu isolated words have been published [50]. There is an effort to use a 50,000 unique and phonetically rich Urdu text corpus for training a Large Vocabulary Continuous Speech Recognition System (LVCSR) [51]. There is research in the Persian language available on large vocabulary and Speech Emotion Recognition using deep learning [52, 53]. The ASR research for some of the languages is progressing like Marathi [54] and Myanmar language [55]. A Corpus containing 16 multiple rare languages spoken in the Eastern and Northeastern regions of India has been published [56]. One more work for Indo-Aryan languages: Awadhi, Bhojpuri, Braj, and Magahi is given in [57].

This para contains some of the most pertinent literature on the subject. The pansori-TEDxKR dataset [58] is based on open online video content like TED conference talks in the Korean language. Pansori retrieves two streams, audio, and subtitle data in SubRip format, from online video-sharing resources using a cloud-based speech API. Another work is JTubespeech [59]: a corpus of Japanese speech gathered from YouTube for speech recognition and speaker verification. JTubespeech produces search terms from HTML files of Wikipedia article words and then extracts videos with subtitles. The num2words Python library is used to replace numbers with their spoken equivalents. The average relative Levenshtein distance between subtitles is used to detect and filter out automated subtitles. A CTC segmentation is used as an alignment tool to calculate a score and re-align the subtitles to the audio. The BembaSpeech [60] corpus was recorded using the Lig-Aikuma app (Gauthier et al., 2016c). Speakers recorded audio from tokenized text scripts at the sentence level using the software's elicitation mode. Bengali Common Voice Speech Dataset [61] is based on a crowdsourcing technique. Multi-Domain Cantonese Corpus (MDCC) [62] is based on audiobooks downloaded from sources filtered manually by language experts. The original audio pieces are converted into shorter audio utterances using a voice activity detection (VAD) tool.

### 2.2.3 Pashto Language

Speech recognition research in the Pashto language is extremely limited. The research approach adopted is very basic and primitive. Research carried out so far can be summarized as:

- An effort has been made to generate a Pashto Medium Vocabulary Speech Corpus [63]. The corpus contains 161 isolated Pashto words, including the most frequently used Pashto words, names of the days of the week, and digits 0 to 25.
- In the Pashto script, there is an OCR-based approach for developing image datasets for optical character recognition systems [64].
- A Pashto Spoken Digits database for automatic speech recognition was developed [65].
- The LVCSR system was developed as part of the Pashto speech-translation system at the SCALE (Summer Camp for Applied Language Exploration) 2015 workshop on "Speech-to-text-translation for low-resource languages." [66].
- Google Translation API's recognition engine supports the Pashto (ps) language for the Neural Machine Translation (NMT) model but there is no API support available in the Google Cloud Speech-to-Text domain [cloud.google.com].

According to the literature review, there is a need to develop an automated mechanism for speech collection from online resources for a low-resource language such as Pashto. To extract speech, the mechanism should be independent of APIs (Application Programming Interface) or other platform-dependent tools, as most low-resource languages lack such support. Further, there is no speech data available for Pashto that can be used for the transcription of an ASR system.

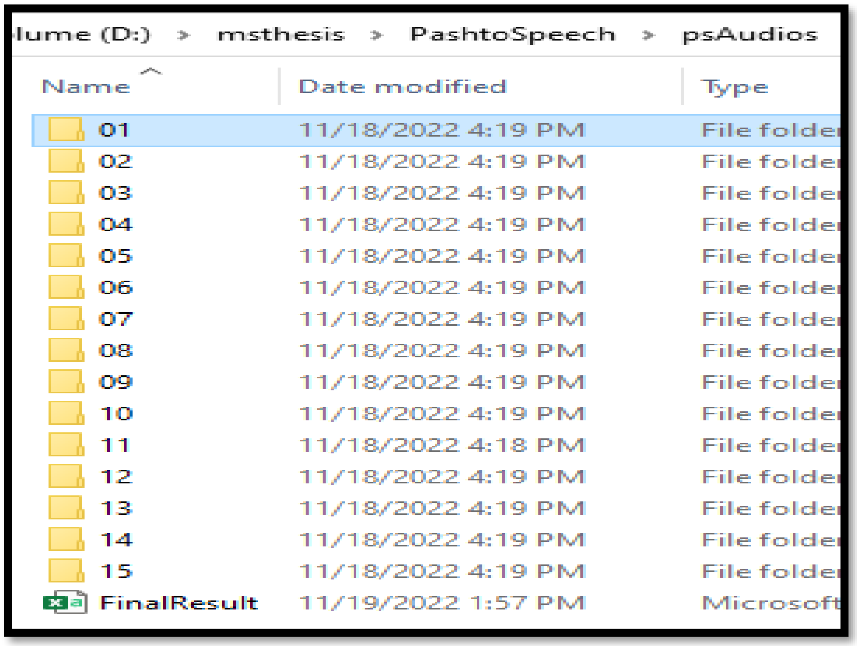
# Chapter 3

# Pashto Speech Corpus Analysis

The chapter explains the characteristics and statistics of Pashto Speech data in the following ways:

## 3.1 Description

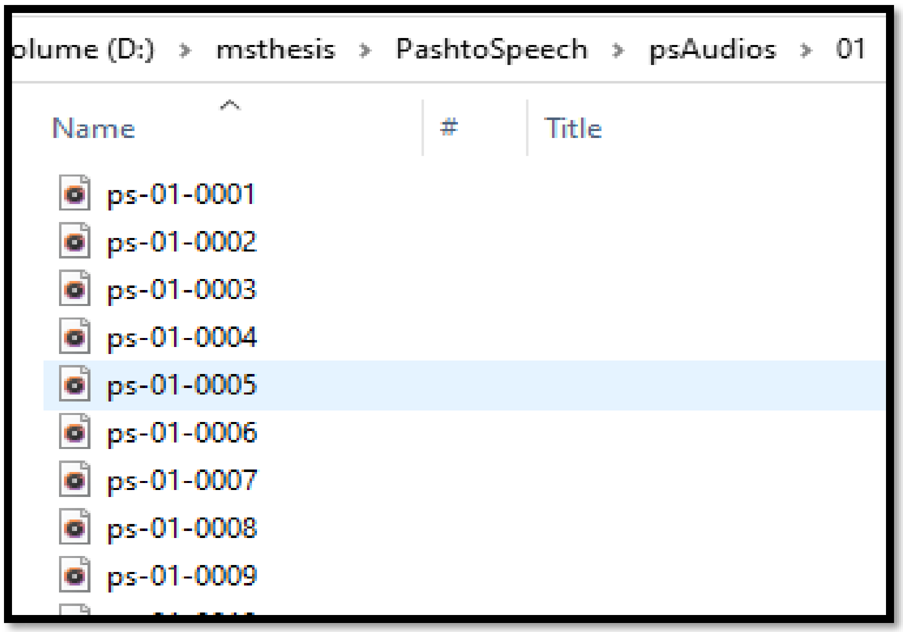
The corpus has a speech of size 725 Megabytes with a total duration of 5.9 hours of speech. The audio data has been split into chunks of one (1) to fifteen (15) seconds. Audio files of different segments are placed in different folders as shown in Figure 3.1. The naming convention of folders and audio files is done on the lines of Mozilla Common Voice Languages as shown in Figure 3.2. An excel file is generated that shows the statistics of clean audio files as given in Table 3.1.



lume (D:) > msthesis > PashtoSpeech > psAudios

| Name | Date modified | Type |
|---|---|---|
| 01 | 11/18/2022 4:19 PM | File folde |
| 02 | 11/18/2022 4:19 PM | File folde |
| 03 | 11/18/2022 4:19 PM | File folde |
| 04 | 11/18/2022 4:19 PM | File folde |
| 05 | 11/18/2022 4:19 PM | File folde |
| 06 | 11/18/2022 4:19 PM | File folde |
| 07 | 11/18/2022 4:19 PM | File folde |
| 08 | 11/18/2022 4:19 PM | File folde |
| 09 | 11/18/2022 4:19 PM | File folde |
| 10 | 11/18/2022 4:19 PM | File folde |
| 11 | 11/18/2022 4:18 PM | File folde |
| 12 | 11/18/2022 4:19 PM | File folde |
| 13 | 11/18/2022 4:19 PM | File folde |
| 14 | 11/18/2022 4:19 PM | File folde |
| 15 | 11/18/2022 4:19 PM | File folde |
| FinalResult | 11/19/2022 1:57 PM | Microsoft |

Figure 3.1: Audio File Organization


olume (D:) › msthesis › PashtoSpeech › psAudios › 01

| Name | # | Title |
|---|---|---|
| ps-01-0001 | | |
| ps-01-0002 | | |
| ps-01-0003 | | |
| ps-01-0004 | | |
| ps-01-0005 | | |
| ps-01-0006 | | |
| ps-01-0007 | | |
| ps-01-0008 | | |
| ps-01-0009 | | |

Figure 3.2: Naming Convention of Folders and Audio Files (Pashto Speech)

| **Audio File Type** | 1s | 2s | 3s | 4s | 5s | 6s | 7s | 8s | 9s | 10s | 11s | 12s | 13s | 14s | 15s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Audio file count** | 1375 | 1003 | 702 | 456 | 326 | 313 | 210 | 180 | 128 | 131 | 104 | 88 | 72 | 69 | 64 |
| **Total Time Length in seconds** | 1375 | 2006 | 2106 | 1824 | 1630 | 1878 | 1470 | 1440 | 1152 | 1310 | 1144 | 1056 | 936 | 966 | 960 |
| **Total Time Length in minutes** | 22.91 | 33.43 | 35.1 | 30.4 | 27.16 | 31.3 | 24.5 | 24 | 19.2 | 21.83 | 19.06 | 17.6 | 15.6 | 16.1 | 16 |
| **Total Time Length in hours** | 0.38 | 0.56 | 0.58 | 0.51 | 0.45 | 0.52 | 0.41 | 0.4 | 0.32 | 0.36 | 0.32 | 0.29 | 0.26 | 0.27 | 0.27 |
| **Collective Time Length in Hours** = 0.38+0.56+0.58+0.51+0.45+0.52+0.41+0.4+0.32+0.36+0.32+0.29+0.26+0.27+0.27 = **5.9 Hours** | | | | | | | | | | | | | | | |

Table 3.1: Clean Speech Files

## 3.2 Speakers & Accent characteristics

The corpus consists of 5221 audio files spoken by 137 speakers. Speakers fall into male and female categories according to gender. Three age groups were chosen: speakers aged year 18-40, 40-60, and over 60. Video URLs were selected based on the speaker's fluency to speak and read Pashto mainly in Afghani and Yousufzai accents. All the speakers are identified as Pashtuns from Pakistan and Afghanistan. Speakers have recorded videos under their choice of environment & uploaded the same at their convenience. All the utterances are expected to have background noise. The noise has been removed with the audio source separation mechanism using the underlining concept of the short-time energy calculation for an audio signal.

## 3.3 Audio Format and Encoding

The video data on YouTube is available in a variety of audio formats and encoding schemes. We used the ".WAV" (Waveform Audio File Format) for speech-to-text experiments with UTF-8 (“Unicode Transformation Format - 8 bits”). UTF-8 is a Unicode encoding system that assigns each character a unique code called a code point. It can convert any Unicode character to a unique binary string and the binary string back to a Unicode character. Unicode is now the global standard for encoding all human languages, including emojis.

## 3.4 Audio Parameters

As the video data on YouTube is sampled with different frequencies. The most common sampling frequency used to produce most digital audio is 44.1kHz. A human can hear all the audio frequencies below 22.05kHz. Since ASR experiments do not require such a high sampling rate, we down sampled the audio to a sampling frequency of 16kHz, which has become the de facto standard for speech recognition in both production and research. The bit rate is set to 256 and the sample width is kept at 2 with a single track (mono).

## 3.5 Availability

The speech data of the corpus can be used for research purposes on a request basis.

Chapter 4

# Video Streaming Technology

Video technology has advanced significantly since the invention of the first moving picture in 1888. Today, you can share a video with millions of people around the world with a few clicks or taps on your desktop and smartphone. On average, each person watches about 16 hours of video per week. Video streaming protocols are required for your device to be able to send and play video. When a video is streamed, it is possible to view it before the entire file is downloaded to the client. [Blog: blog.vidizmo.com]. A video can be streamed directly from streaming servers to various end devices, such as personal computers or smartphones, using video-sharing platforms like YouTube. As discussed earlier the source of speech acquisition in our model is YouTube. Therefore, a quick overview of how streaming works in general, how it affects video surfing or downloading, and how YouTube utilizes it is given in this chapter.

## 4.1 What is Video Streaming?

Streaming is a mechanism that allows you to watch audio or video without downloading it. Initially, websites were made up of text pages with a few images. These days anyone with a high-speed Internet connection can watch high-quality media online utilizing this facility. The continuous transmission of audio or video files from a server to a client is known as streaming. The client device stores the media file remotely and then transmits it to the server over the Internet instantly. [Blog: cloudflare.com]. The following diagram (Figure 4.1) summarizes the entire process [blog:vidizmo.com]:-

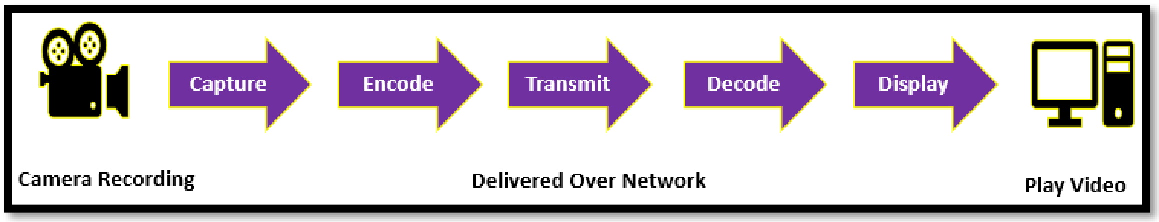


Figure 4.1: Video Streaming Generic Process

## 4.2 Streaming and Downloading?

Streaming is more efficient than media downloading. When a media file is downloaded, the device stores the copy of the complete file and then plays it when it finishes downloading. On the other hand, when the video is streamed, it is neither saved nor copied and the browser plays it instantly. The entire file is not loaded at once rather loading happens in segments without saving information locally in the browser [Blog: cloudflare.com].

## 4.3 Technical Approaches

Generally, streaming occurs in three ways, as shown below [Blog: ns1.com].

### 4.3.1 Progressive Download

The delivery of streaming video or audio files over HTTP is referred to as progressive download. When the file has been downloaded sufficiently to play it, playback begins, and the remaining portion of the file is downloaded in the background. There is normally a delay when playing the video, and it is not possible to view the next parts of the video if they have not yet been downloaded. Videos are streamed at one type of bit rate, and all users receive the same video file regardless of connection speed.

### 4.3.2 Single-Bitrate Streaming Protocols

Real-Time Streaming Protocol (RTSP) protocols necessitate the use of a streaming server and a dedicated client player. The client displays the contents received in segments from the RTSP protocol before recycling it. Unlike in a progressive download, users can skip parts of a video by sending a request to the streaming server to load a new segment of the video. RTSP is UDP oriented and provides better performance with the same connection speed.

### 4.3.3 Adaptive Bitrate Streaming

An adaptive bitrate is an advanced form of streaming. it uses UDP to process variable bit rates when transmitting media. The bitrate of streamed content is tuned according to the user's connection speed, network conditions, and device resources. The video creator must upload the same audio or video file encoded with different bitrates for adaptive bitrate streaming to work. On the client side, the video player monitors device-end processing. The player provides the highest possible bit rate at which the user can watch media without interruption. This situation is evaluated every 5-10 seconds, and if necessary, the player requests different bit rates according to user requirements.

## 4.4 How YouTube Streaming Works?

After Google, YouTube has the second-most views and, after Netflix, it is the second-most favorite online video content website. YouTube has more than 2 billion users logged in monthly across the globe. [Blog: influencermarketinghub.com]. YouTube supports both DASH (Dynamic Adaptive Streaming over HTTP) and legacy streaming techniques. DASH provides both audio and video streams to be downloaded. The media streams can be merged using media editing tools like FFmpeg. Legacy streaming is also known as progressive download. It provides both audio and video streams in the same file. The resolutions of 720p and lower are supported [88]. Streaming techniques enable us to view audio or video content on YouTube at numerous bit rates and resolutions.

# Chapter 5

# Methodology

The chapter explains how data is collected and preprocessed for model processing of speech data. A detailed overview of the model's structure and functions is then provided. Finally, the output data is post-processed to achieve the desired results. The working of the chapter is described in multiple sections; -

## 5.1 Data Collection Process

In this section, we will analyze the existing methods of how voice data is collected and used for ASR tasks. Following that, a brief description of how we collected spoken data for our language of interest (Pashto) is presented.

### 5.1.1 Speech Data Collection in general

Generally, gathering data is an essential component of developing ASR systems for under-resourced languages. Speech data can be generated in two ways: those that make use of already-existing audio or video resources and those that record speech as part of the data collection procedure. The major portion of already recorded speech data is available on the internet, and some of the data can be accessed offline and stored in stand-alone repositories. Already recorded speech categories include TV or radio broadcast recordings, spoken lectures, parliamentary speeches, audio or video data on social media platforms, offline speech collections, etc. Although the prerecorded voice data serve as the starting point for corpus generation, the main challenge here is to edit and transcribe the recordings.

The second method of recording speech as a part of data collection faces certain complexities: firstly, low-resource languages have normally poorly standardized writing systems, secondly, manual transcription is time taking and a lack of qualified language professionals makes manual audio transcription more difficult. Crowdsourcing [67] may provide some relief in this regard, but the contributor may not record their voices accurately and there are not enough workers available in case of a low-resource language [64]. Thirdly, the available sources of speakers are not often diverse enough to be useful for ASR. The recording process can be done using some sort of menu-driven telephone service, but it could be time taking and prone to repetition and mistakes. As an alternative, recordings can be obtained during in-person recording sessions, but logistical difficulties could arise as more recording devices may be required. keeping its widespread availability developing smartphone apps can accelerate recording speed, but its development, maintenance, and operation cost is a major concern. However, spontaneous speech corpora with limited resources are normally less helpful as a starting point for ASR development. Keeping in view the cost, efforts, and difficulties inherent in resource use, transcribing spontaneous speech in resource-scarce languages favors the existing recorded speech data approach.

### 5.1.2 Speech Data collection for Pashto

This is an important step in the corpus development study. We have used the existing recorded speech data approach to gather data. A thorough literature search was conducted to find Pashto literature and its corresponding video data on YouTube. Video information was searched on YouTube for different categories of Pashto literature. There are no criteria adopted to select specific literature. Approximately 10-15 genre categories were identified including culture, Drama, Fairy Tales, Health, interview, Literature, lifestyle, news, Novel, Politics, Short stories, Sports, and TV shows. Random keywords were searched to find speech URLs on YouTube.

The study focused on URLs that were posted between 2018 and 2022 on multi-YouTube channels. Depending on the availability of videos on YouTube, five categories of Pashto literature including novels, dramas, short stories, fairy tales, and interviews were selected.

A total of 130 video URLs are selected. Novels are audiobooks on different topics uploaded on YouTube. Dramas include web series, TV serials, and telefilms on daily life happenings. Short stories normally consist of personal biography, ideas, or past happenings of speakers. Fairy tales are cartoon stories either translated or dubbed into the Pashto language by different speakers. Interviews of different speakers in daily routines are also part of the literature. A URL excel sheet was maintained with some additional information like video time length, the number of speakers for different ages, channels, and the date of uploading. Watch time is used to calculate the total video hours of URL excluding ads. Details of the video data selected are given in Table 5.1.

| Genre Category | Links Selected | Speakers | | Video length (Hours) | | Uploading Duration |
|---|---|---|---|---|---|---|
| | | Male | Female | Male speakers | Female Speakers | |
| Novel | 40 | 25 | 06 | 20.01 | 09.04 | |
| Interviews | 16 | 08 | 23 | 04.03 | 10.03 | |
| Short Stories | 13 | 06 | 09 | 02.09 | 03.02 | |
| Fairy Tales | 11 | 08 | 05 | 02.04 | 04.03 | |
| Culture | 12 | 05 | 04 | 02.02 | 03.05 | |
| Politics | 10 | 02 | 02 | 03.07 | 02.08 | |
| News | 15 | 03 | 04 | 02.06 | 03.03 | |
| Drama | 13 | 15 | 12 | 04.04 | 08.09 | |
| | | 72 | 65 | 39.36 | 42.37 | |
| Total:08 | 130 | Total: 137 | | Total: 81.73 | | 2018-2022 |

Table 5.1: Video URLs Selected

## 5.2 Data Preprocessing

This is the initiating process of the project, and its purpose is to produce a refined URL list as input to the pipeline model. As described earlier, the URLs are selected for male and female speakers of different age groups according to genre categories. The process is divided into two steps:

- Firstly, a fair review is carried out to find the target video URLs. Videos for all the selected URLs were played on YouTube to check for any inconsistency or missing segments. URLs with muted sections and abnormal sounds were dropped. URLs containing a major portion of its video as music were discarded. The filtered URLs were placed in an excel file with a record of the URL's title, date of uploading, channel name, speaker age, and gender. Final URLs were copied to the text file.

- Secondly, the URLs were tested against the flow of independent video download and audio extraction modules. There was a problem with downloading some videos due to their copyrights or being unavailable as removed afterward by the channel. All such problematic URLs are identified and replaced with similar valid URLs; thus, a refined Text file is produced.

## 5.3 Model Development

The section illustrates in detail how the proposed model for Pashto Speech corpus development works. To extract audio chunks an automated pipeline is proposed. The pipeline extracts audio files from YouTube URLs and split them into required time-length chunks. Multiple modules are used to process the audio segments, resulting in clean audio chunks for our project as shown in Figure 5.1. For clarity, the speech generation pipeline is further divided into five subcomponents: raw audio extraction, audio cleaning, audio parameter adjustment, audio segmentation, and postprocessing clean audio chunks. Details are described as under.

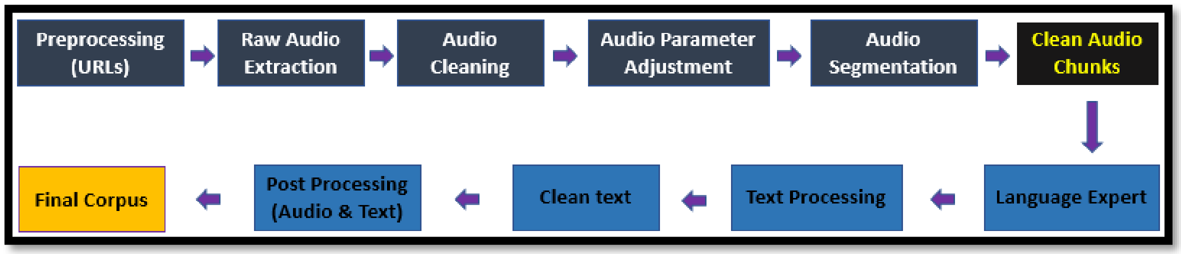


Figure 5.1: Pashto Speech Corpus Model Block Diagram

### 5.3.1 Raw Audio Download

This module deals with downloading raw audio files from YouTube videos. A raw audio file consists of vocals and noise. In this process, a URL text file is provided as input to the pipeline. The procedure can be explained in the following steps:

#### 5.3.1.1 Create URL List

The pipeline reads the URL text file line by line and compiles it in the form of a list. Python has a set of built-in methods that can be used for string manipulation. We tested the following three well-known methods to perform this task.

- The readlines method returns a list of lines from the stream. A new line character \n is also added in the string and it can be removed with str.rstrip('\n').
- We can iterate over the file to read it line by line. The iteration method only takes one line of the file content to the memory and processes it.
- The file.read(size=-1, /) method reads from the file until EOF (End of File) if the size is not set. The splitlines function can be used with it to split the lines at their boundaries to compile a list.

Memory usage and other processing statistics can be observed using python inbuild modules like tracemalloc or debugging code chunks. We tested and compared the efficiency performance of the above three methods used for URL list creation. We increase the number of lines in the tested file incrementally to compare the performance difference. Generally, the file iteration method is much better than the readlines method from the perspective of memory usage when the file size is super large. The readlines method holds all the lines of the file in the memory, but the iteration method only takes one line of the file content to the memory and processes it to avoid memory errors [90]. In our case, file.read().splitlines() is the most efficient method as compared to the other two methods because it takes less time to compile the URL list and also does remove newlines.

#### 5.3.1.2 Fetch URLs Title

The purpose of this module is to retrieve the title information of the videos to be downloaded. We want to save the extracted audio files with the name of the title; therefore, audios need to be fetched with their meaningful title as given on YouTube. Video title information can provide help and clarity when keeping track of audio chunks in the directory and processing text for speech transcription afterward. Python provides several inbuilt functions to download videos and their information from YouTube.

youtube-dl is one such library that is available in the public domain and can be used according to one's needs.

YoutubeDL objects have a method that allows InfoExtractors to work in some defined order. Information extractors extract all the needed information to download the actual video file and write it to a disk. When a URL is passed as an argument, the YoutubeDL object forwards it to the first InfoExtractor being able to handle the dictionary and stores the video information retrieved by The InfoExtractor. The information is then sent to YoutubeDL, which downloads the video into the directory. We have implemented InfoExtractors for retrieving Youtube video title information.

#### 5.3.1.3 Extract Raw Audios

This is an important step of the audio download module as it supplies raw audio speech for the further working of the model. Python provides important tools like youtube-dl, pytube, Moviepy, etc for audio or video exploitation. youtube-dl (or YoutubeDL) is a platform-independent and command-line library to download videos from Youtube as described earlier. The youtube-dl project is currently being developed on GitHub. pytube has no decencies and it provides a command line facility to manipulate YouTube Videos owned by tfdahlin. MoviePy is another python module used for video processing, editing, or creating advanced effects.

As discussed earlier, we need to save the audio files in the directory with the name of the video title information. We tested youtube-dl for audio extraction from Youtube videos. The youtube-dl library was creating problems in saving audio files with the name of the title in the directory. The cause traced during debugging was with special characters in the title string. Errors were occurring while saving audio files with the title name. Despite we used the string characters replacement mechanism, it didn’t work. Renaming the audio file other than the title worked but it was not desired. MoviePy was also tried but it needed other dependencies and packages. Then pytube being a lightweight and dependency-free library for audio download was tried and it solved our purpose.

Each file system has its own set of rules for forming the various components of a directory or file path. Some of the special characters cannot be used when creating or renaming folders and files in the directory. These characters include: \, /,:, *, ?, ", <, >, |, ~, #, %, &, +,{, }, - note that comma (,) is not included. Therefore, the title information fetched in the first step was passed through the string character replacement mechanism to remove forbidden characters when saving audio files in the directory folders. We have implemented the concept in Python.

As previously stated in the video streaming chapter, YouTube supports both legacy streaming (progressive) and DASH (Dynamic Adaptive Streaming over HTTP or MPEG-DASH) techniques. The legacy streams contain the audio and video in a single file whereas adaptive streams split the video and audio tracks. Pytube supports both Progressive & DASH streams and makes it easy to filter media based on the choice of the stream according to user preferences. The refined title information is supplied to the YouTube object (yt) for accessing the media streams. We chose progressive streaming using the filter function of The Pytube API. The top streams are selected using the first() method and likewise downloaded at the desired path with the download method() of pytube library.

### 5.3.2 Audio Cleaning

The output of the audio extraction module is raw audio i.e, it is composed of noise and vocals. Raw audio is provided as input to the audio cleaning module. The cleaning module process raw audio and removes noise to get clean audio. Speech source separation is not a simple task. The underlying concept that works for speech source separation is time-frequency (TF) masking. The process of awarding values to the slots of a time-frequency interpretation to improve, reduce, or separate portions of an audio segment is termed Time-frequency masking [Blog: mathworks.com]. There are varying frequency corresponding to various types of sounds. For example, the lead vocals would occupy different frequency bands than the drums.TF masking filters the spectrum of frequencies that comprise a piece of music, allowing us to choose which frequencies to preserve.TF masking has been implemented using Spleeter, an AI (artificial intelligence) tool that breaks down music into its instrumental pieces and is owned by Deezer an online music streaming provider. Spleeter divides raw speech into multiple sound sources using pre-trained models. These models include two, four, and five stems. The 2-stem model is the simplest and divides input speech into vocals and accompaniments). Similarly, 4-stems produce output sounds such as vocals, drums, bass, and others, while 5-stems produce output speech such as vocals, drums, bass, piano, and others [68].

Our audio cleaning module is using Spleeter's default 2-stems model for audio source separation. The 2-stems model splits the audio into two very basic layers i.e., vocals track and accompaniment (karaoke or other track). Spleeter has two dependencies: ffmpeg and libsndfile (). ffmpeg library is dependant on ffprobe. ffmpeg is a multimedia stream manipulating tool whereas ffprobe is the sub-part of ffmpeg. libsndfile is a C language library used for reading or writing audio files in a wide variety of audio file formats. These dependencies can be installed according to Spleeter version, and the programming environment used for development.

The general expression to accomplish the 2-stems source separation is given as: "**spleeter separate -p spleeter:model type -o output/ directory.audio format**". The -p is used for providing the model settings (2stems) "-o" stands for the output file name in a particular audio format. Using the source separation concept discussed above, we have used the 2-stem Spleeter model to extract vocals from raw audio and store them in the desired directory (dataset folder).

### 5.3.3 Audio Parameters Setting

Audio parameter adjustment is the third module of our speech pipeline, and its function is to reset the audio sampling rate and other necessary parameters. An audio file has several characteristics associated with them. The most common parameters include channels, frame rate (sample rate), and sample width. To understand its importance a brief overview is necessary [Blogs: vocitec.com, github.com, headphonesty.com]:

- A communication channel that transports a speech signal from its source to the listener is known as an audio channel. An audio channel can be monophonic, stereophonic, or multiphonic. Mono process one sound, stereo support two channels, and multi-channel sound has more channels.
- The sample rate is defined as the number of samples of a signal collected per second and is measured in Hertz (Hz) or kHz, as shown in Figure 5.2. These samples are

evenly spaced in time. For example, a sampling rate of 16000 Hz means that 16000 samples are captured per second, with every single sample being precisely 1/16000 of a second equally spaced on the time scale.

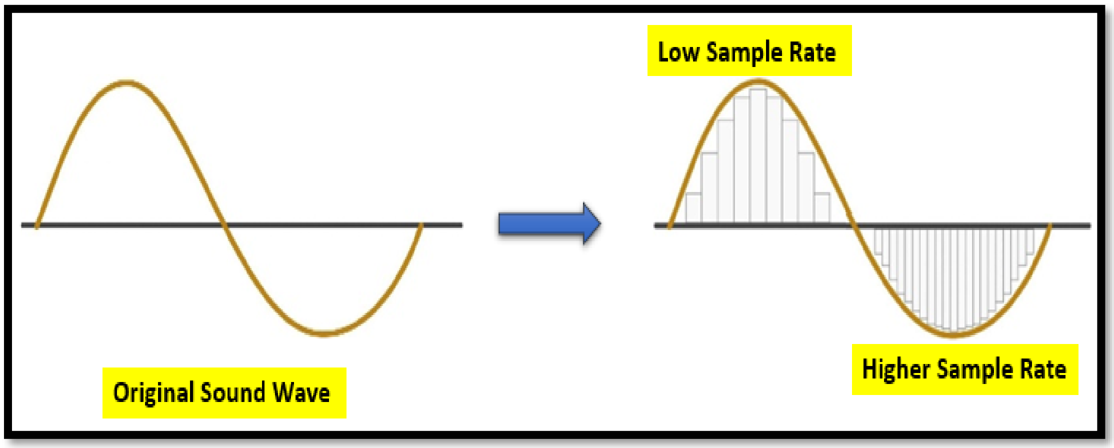


Figure 5.2: Different Sampling Rates

- The **frame rate** measurement for a video is analogous to the sampling rate. A video is a collection of images, that are displayed back-to-back rapidly to create the appearance of continuous, uninterrupted motion or movement. These collections are referred to as "frames" in this context. Some common sample rates are 48 kHz (most common for audio tracks in movies), 44.1 kHz (most frequently used for music CDs), and 8 kHz (most frequently used for telephone conversations).

- The amount of information bits in each sample is known as the **bit depth** and is expressed in bits per sample. A sample depth of 16 bits means that each audio sample can represent 2^16 = 65,536 distinct amplitudes or levels of sounds.

- Yet there is another important property of audio called a **bit rate**, which is calculated as **bit rate** = **sample rate** x **bit depth**.

Several APIs recommend using a minimum sampling rate of 16,000 Hz because sampling below this rate can cause a loss of information to the signal. Wav2Vec framework has been trained on speech sampled at 16kHz. Python provides Pydub that can be used to play, split, merge, and edit only the .wav audio files**.** The instance of AudioSegment on creation saves audio parameters including sampling rate from extracted speech. In our project, we have kept speech to be sampled at a rate of 16 kHz. Channel (1) is used to avoid unnecessary heavy loading of audio data i.e any audio recorded with stereo or multi-channel is converted to mono channel as shown in Figure 5.3.

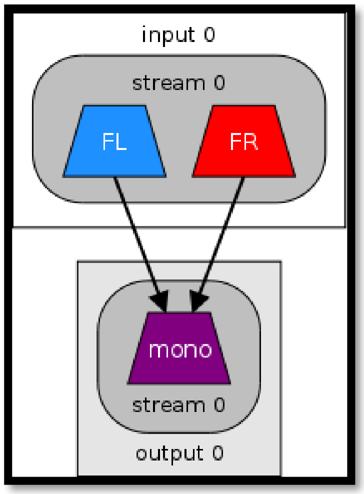


Figure 5.3: Conversion of Stereo to a Mono Channel

### 5.3.4 Audio Segmentation

Breaking audio files based on quiet parts or muteness is a common use case in audio processing and has various applications. Multiple Machine learning tools allow us to carry out this function. Python provides packages like libros for audio signal extraction and visualization and pydub for audio file manipulation. Librosa is generally geared more toward music, but it can also be utilized for audio-slicing purposes. PyDub takes very less post-processing tasks utilizing the 'split_on_silence' function. When the audio is large or the silence time for splitting exceeds 5 seconds, the processing time increases. It worked fine in our case because the silence threshold used is less than five seconds.

There are two options available to slice audio into chunks i.e., standard splitting and split on silence. Standard splitting causes audio to be sliced into equal-sized chunks in time. The issue with this kind of segmentation is that we get half-uttered syllables at the start and end of the audio chunk. There is also a chance that the chunks may also contain undesirable silences between the start and end of the audio file. Half-sliced chunks need time padding. One solution is that we can use time padding of standard time length, but it again causes half-sliced words as different words have different time lengths. Finding the variable time length padding for each chunk sliced differently is time taking and difficult.

Even though undesirable silence within the chunk needs to be removed, that's why we opted for the split-on-silence method. To split audio, several parameters can be passed as an argument to the 'split_on_silence' module. Usually, to get desired chunks length following parameters are used [Blog: github.com/jiaaro]:

- **sound** - It is the input speech segment utilized for silence detection.
- **min_silence_len** - It is the portion of the audio segment that must be silent for a certain time in milliseconds, and it is selected as the threshold duration for the segmentation of an audio file. By default, it is 1000 milliseconds (ms). The optimized value for our requirement is **700 ms**.
- **silence_thresh** – It is the higher limit for audio to be silent in dFBS (**decibels relative to full scale**). By default, it is -16 and below this value, audio is considered silent. The best value suited to our requirement is **-40 dB**.
- **seek_step** - It is the step (window/frame) size used for iterating over the segment in milliseconds. A sliding window of a certain time size (in ms) is used for the detection of silences. seek step sets how much to move the sliding window over each time. We are using the default value of **one ms**.
- **keep_silence** - It is the amount of silence added at the beginning and end of the chunks. It is used to avoid abruptly cutting off an audio segment. The "True" value denotes that all the silence is retained, and the "False" value denotes that none is kept. It is measured in milliseconds and the default value is 100ms whereas in our case it is **100 ms**.

There are two important points in the audio segmentation process that make sense i.e how silence threshold works? and how audio slicing is performed (seek step). Analog and digital audio levels are measured in decibels (dB) and dBFS respectively. Decibels (dB) is generally a measure of the ratio between two values on a logarithmic scale. The values can be a power or any field quantity. When the term is applied to audio signals the values of amplitudes between two audio signals are compared given as **dB = 20log10(A1/A2),** where **1 dB = 1/10th** of a bel and A1 and A2 are the amplitudes of the signals of interest [69]. **dBFS (dB relative to full scale)** also measures the ratio between two signal levels and hence, both dB and dBFS are dimensionless metrics. Mathematically dBFS is expressed as **dBFS = 20log10(A/Max),** where A is the amplitude of the measured signal, and Max is the maximum (peak) possible amplitude for that signal. For instance, if the signal's amplitude is 1, and the maximum amplitude that's possible for the signal before clipping (distortion) is 10, then the dBFS value for the signal is calculated as **dBFS = 20log10(1/10) = -20 dBFS,** it implies that the signal in our example is at a level that's 20 dB below its clipping point (maximum). When a signal is at its maximum amplitude, its dBFS is 0. All subsequent dBFS readings for the signal will be negative since the signal level can only fall from its peak amplitude level.

The second point of interest is the windowing or framing of a digital signal. It is based on short-term energy calculation for a signal. The concept is used to detect silence at a certain threshold energy level. The lowest energy level becomes the threshold value, any silence equal to or less than this value causes the audio to split into chunks. Since for most of the practical cases the unvoiced part has low energy content and thus silence (background noise) and unvoiced part is classified together as silence/unvoiced and is distinguished from the voiced part. Short-time energy calculation can be used to find voiced and silent segment parts of an audio signal. The energy of a signal x(m) is given as [72]:

$$\mathrm{E} = \sum_{m=-\infty}^{\infty} x^2(m) \qquad \text{Equation 2}$$

$$En = a_0 + \sum_{m=n-N+1}^{n} x^2(\mathrm{n} - \mathrm{N} + 1) + \ldots + x^2(n) \qquad \text{Equation 3}$$

Generally, for the time-varying signals, the speech signal is divided into frames at its utterance level, therefore applying the nth window frame to calculate the short-term energy of the speech signal. Thus, for frame 1 and n-th frame equations becomes [73]:

$$\mathrm{E}(1) = \sum_{m=0}^{N-1} [x(m)]^2 \qquad \text{Equation 4}$$

$$\mathrm{E}(\mathrm{n}) = \sum_{m=-\infty}^{\infty} [x(m)\,.\,w(n-m)]^2, \qquad \text{Equation 5}$$

where, w = window function,

and n = Shift in samples/Frames

A window of the rectangular pulse does not change the amplitude of the speech sample. It is given as:

$$w(n) = \begin{Bmatrix} 1; & 0 \leq n \leq N \\ 0; & else \end{Bmatrix} \qquad \text{Equation 6}$$

The high and low energy would denote spoken and silent parts respectively. Under certain threshold energy, the unvoiced shall be considered as silence. Using the above concept an optimized approach was used to get the desired clean audio file chunks.

### 5.3.5 Clean Speech

This marks the final process of audio chunk formation. The procedure is divided into two steps. In step 1, we intend to divide the speech from 1 second into 15 seconds chunk files and save it in separate folders. Normally speech segments of less than 30 seconds give good accuracy [Sewade Ogun] when used for ASR testing, therefore we selected chunks of the time length of 1-15 seconds. A Jason file is created to keep a record of counts of audio files for each chunk type.

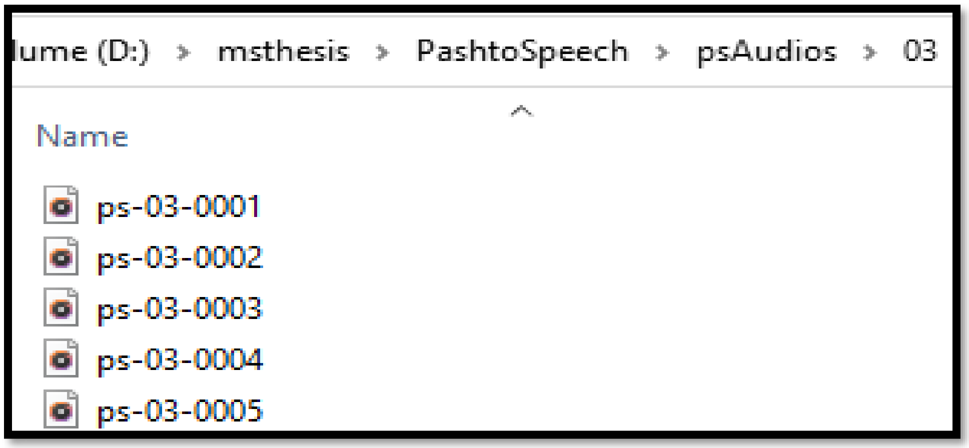
In the second step, the speech collection is passed through the filter to remove unnecessary data. We have used the round function to avoid float values as pydub gives output in integers.

## 5.4 Text Processing

This section of the model is concerned with text formulation for the final clean audio files generated by the speech processing pipeline. To avoid abrupt cutting, each audio file is padded with one millisecond at the beginning and end. This facilitates comprehension of spoken words. Furthermore, we saved the raw audio with the title information, which aids in the understanding of the individual speech segment in case of any ambiguity. Pashto script writing is not an easy task, it needs a language expert to understand the speech and produce a valid text for it. The language expert will produce the transcribed text for corresponding speech segments.

## 5.5 Data Postprocessing

This section deals with the activities required to produce transcribed speech. Language experts produce text for corresponding audio files. Each audio file is given an ID value and a similar ID for a corresponding audio file is also given as shown in Figures 5.4 & 5.5.

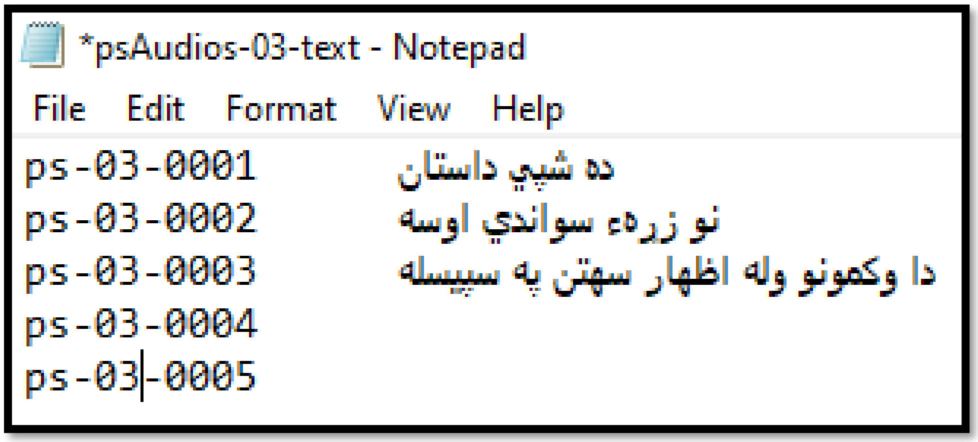


lume (D:) › msthesis › PashtoSpeech › psAudios › 03

Name

ps-03-0001
ps-03-0002
ps-03-0003
ps-03-0004
ps-03-0005

Figure 5.4: Audio Files IDs in Pashto Speech



*psAudios-03-text - Notepad

File Edit Format View Help

```
ps-03-0001        ده شپې داستان
ps-03-0002        نو زره‌ء سواندي اوسه
ps-03-0003        دا وکمونو وله اظهار سهتن په سپیسله
ps-03-0004
ps-03-0005
```

Figure 5.5: Transcribe Pashto Text in Pashto Speech

# Chapter 6

# Model Evaluation

Our research involves the development of an automated speech corpus generation pipeline and its application to produce speech data in Pashto. We explore that there are no specific evaluation metrics that can be used to check the pipeline's operation; however, we can test the model's output accuracy and quality using parameters such as audio extraction, source separation, audio splitting, URL input, etc. In this chapter, we will present some metrics for evaluating our model and discuss the results while analyzing the potential parameters.

## 6.1 Evaluation Metrics

Our model is made up of interconnected modules. One module's output serves as the next module's input. Using the input and output results, along with module parameters, certain evaluation metrics can be defined to track the model's progress. These metrics are as follows:

- **Audio Files**

In our project, the output speech is divided into multiple audio chunks known as audio files. The length of these audio or speech files ranges from one (1) to fifteen (15) seconds.

- **Silent Audio File**

As shown in Figure 6.1, these are the audio files that do not contain any human vocal data.

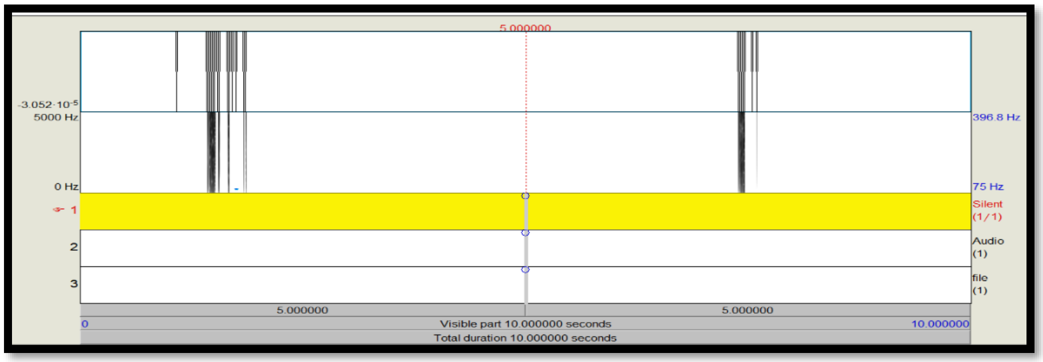


Figure 6.1: Silent Audio File (PRAAT image)

- **Partial Silent Audio Files**

These are the audio files that contain both human vocal data and silence. As illustrated in Figure 6.2, the words in partially spoken segments are clear and complete.

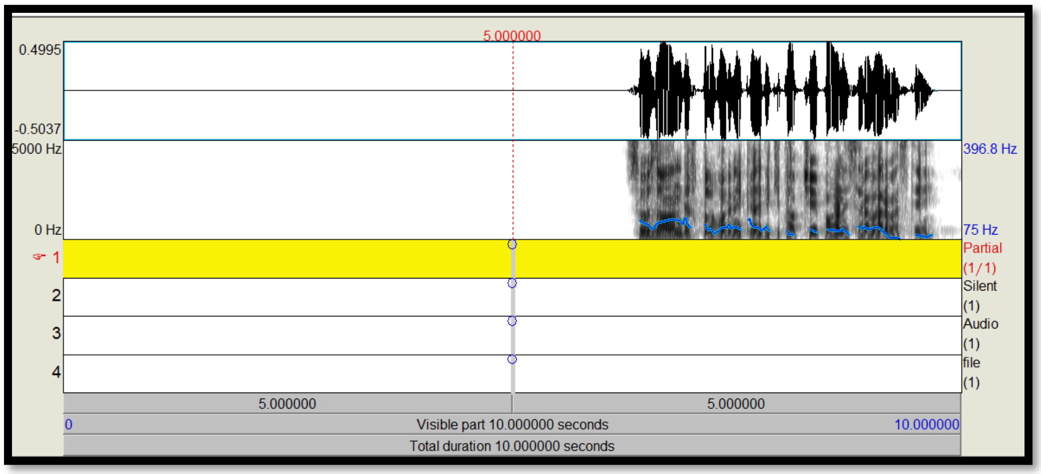


Figure 6.2: Partly Silent Audio File (PRAAT image)

- **Sliced Only Audio File**

These are the audio files that contain human vocal data, but the words spoken words at the beginning and end of the audio file are half-sliced. There are no silent segments that are longer than the threshold value (Loudness less than -40dB & Silence Length of 700 milliseconds) as shown in Figure 6.3.

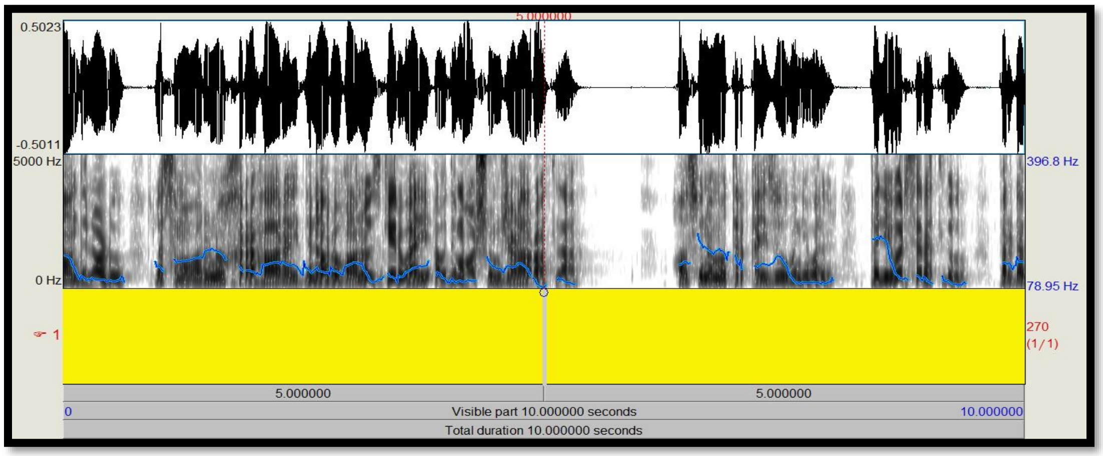


Figure 6.3: Sliced Only Audio File (PRAAT image)

- **Sliced & Silent Audio Files**

These are audio files that include both half-uttered vocal and silent segments. The half-spoken words occur at the start and end of an audio file as shown in Figure 6.4.

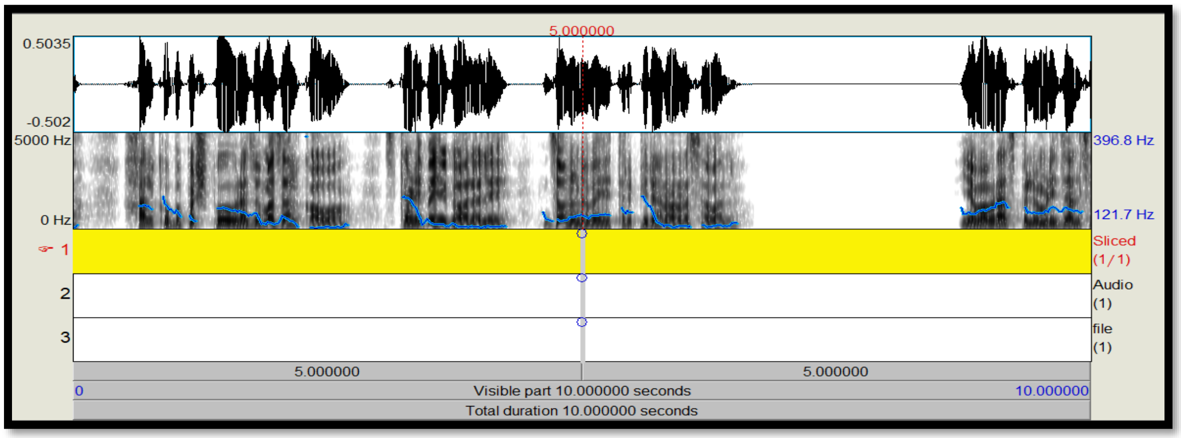


Figure 6.4: Sliced and Silent Audio Files (PRAAT image)

- **Noisy Speech Audio Files**

These are audio files with background noise, echoes, multi-speaker vocals, or music. Such speech contains vocals that are either inaudible or clear enough for a normal human to understand.

- **Clean Speech Audio Files**

These are the model's final audio files, and they contain clear vocals with no or negligible background noise. A normal person can hear and comprehend its words easily.

- **Valid Speech Data**

These are clean speech audio files that contain clear vocals with no background noise or music source. A normal person can hear and comprehend its words. Such data do not contain any faulty audio segments.

- **Invalid Speech Data**

These are clean speech audio files that do not contain any of the following unwanted segments.

  - Silent.
  - Partial Silent.
  - Sliced only.
  - Sliced & Silent
  - Noisy Segments

- **Quality of Speech Data**

The quality parameters of output speech are given as follows: -

  - Channel = Mono
  - Sampling Width = 16 bit
  - Sampling Rate = 16 kHz
  - Audio Format = wav

- **Accuracy of Speech Data**

It is the percentage of valid speech data in the dataset under consideration.

## 6.2 Experiments

Keeping in mind the data flow and functionality of our model, we can experiment as follows: -

### 6.2.1 Audio Extractor results

- **Results**

Raw audio extraction is the first intermediate output in our project. YouTube supports streaming techniques that deliver audio and video in separate streams. The Pytube audio extractor API was used in conjunction with the progressive streaming technique to extract audio data. The results for different URLs (40, 90, and 135) are shown in Table 6.1.

| URLs Data | Failed Extraction | Successful Extraction | Time Taken | Total Final | Accuracy (%) |
|---|---|---|---|---|---|
| 40 (31.6 Hours) | 03 (7.5%) | 37 | 19 Minutes | 37 | 92.5 |
| 90 (59.8 Hours) | 02 (2.2%) | 88 | 35 Minutes | 88 | 97.77 |
| 135 (82.73 Hours) | 05 (3.7%) | 130 | 73 Minutes | 130 | 96.29 |

Table 6.1: Audio Extraction for URL Input Data

- **Discussion**

Results of Table 6.1 show that more than 90% of URLs selected as input gave valid audio extraction. The extraction failure rate is less than 8%. When the corresponding URLs were examined, it was discovered that either the URL had been removed by the uploader or it did not allow the data to be downloaded. Such issues pertain to the source of video data but not the extractor.

### 6.2.2 Speech Source Separation

- **Results**

The speech source separation module segregates raw audio into vocals (clean speech) and accompaniment (noise). Table 6.2 gives the time taken by the source separation mechanism to split input audio into noise and clean vocals from different speech inputs whereas Table 6.3 shows the overall result for 81.73 hours (130 URLs) of video data input. The ratio of clean to noisy speech audio files along with the accuracy of clean speech is provided.

| URLs | URL Video Length | Audio Extracted | Noise Removal Time |
|---|---|---|---|
| 37 | 31 Hours | 21 Hours | 23 Minutes |
| 88 | 59 Hours | 40 Hours | 35 Minutes |
| 130 | 81.73 Hours | 54 Hours | 83 Minutes |

Table 6.2: Noise Removal Profile

| Audio File Type | Audio Files (in Seconds) | | Accuracy (%) |
|---|---|---|---|
| | Clean | Noisy | |
| 1s | 1375(22.9Minutes) | 3(0.05Minutes) | 99.78 |
| 2s | 1003 | 2 | 99.80 |
| 3s | 702 | 3 | 99.57 |
| 4s | 456 | 4 | 99.12 |
| 5s | 326 (27.1 Minutes) | 5 (0.4 Minutes) | 98.46 |
| 6s | 313 | 7 | 97.76 |
| 7s | 210 | 6 | 97.14 |
| 8s | 180 | 8 | 95.55 |
| 9s | 128 | 9 | 92.18 |
| 10s | 131(21.8Minutes) | 10(1.6Minutes) | 92.36 |
| 11s | 104 | 10 | 90.38 |
| 12s | 88 | 11 | 87.5 |
| 13s | 72 | 10 | 86.11 |
| 14s | 69 | 11 | 84.05 |
| 15s | 64(16 Minutes) | 12(3 Minutes) | 81.25 |

Table 6.3: Audio Source Separation Profile

- **Discussion**

The results of Table 6.2 shows that splitting raw audio of 21 hours into vocal and accompaniment takes 23 minutes. Similarly, 40 and 54 hours of raw audio data take 35 and 83 minutes respectively to remove noise. As a result, the time required to remove noise from audio data increases as the amount of input speech increases. Table 6.3 gives results of clean and noisy speech for the whole input URL data. For instance, out of 1375 audio files of type one-second, only 03 audio files are noisy. Similarly, 326 audio files of type five-second contain 05 noisy audio files, 131 audio files of type ten-second contain 10 noisy audio files, and 64 audio files of type fifteen-second contain 12 noisy audio files. The accuracy of source separation is good with smaller audio files and decreases as the file size increases. For example, it is 99% in the case of a one-second audio file and 81% in the case of a fifteen-second audio file. Audio cleaning entails separating the media into vocal and accompaniment tracks. The vocal is clean audio, whereas the accompaniment is noise.

### 6.2.3 Speech Segmentation

- **Results**

The final output in our project is clean audio file generation ranging from one second (1s) to fifteen seconds (15s). We conducted two types of segmentation methods i.e standard splitting and split on silence. Results of audio Segmentation for 10 minutes, one hour, and 5 hours of input URL data using both techniques are shown in Table 6.4 and Table 6.5. Table 6.6 shows the results of the time taken when segmenting the raw audio. Results include data for 10 minutes, one hour, and 5 hours of input URLs. Only the audio files of one-second (1s), five-second (5s), ten-second (10s), and fifteen-second (15s) duration are included in the results with calculated accuracy.

| Input Data | Total Files | Silent Files | Partial Silent Files | Sliced Only Files | Sliced & Silent Files | Invalid Speech | Valid Speech | Accuracy (%) |
|---|---|---|---|---|---|---|---|---|
| 10 mins | 60 | 4 | 3 | 33 | 12 | 52 | 8 | 13.33 |
| 1 Hour | 11 | 360 | 21 | 220 | 51 | 303 | 57 | 15.83 |
| 5 Hours | 59 | 1800 | 109 | 1009 | 239 | 1416 | 384 | 21.33 |

Table 6.4: Audio Standard Splitting (10-Second Audio Files)

| Input Data | Total Files | Silent Files | Partial Silent files | Sliced Only Files | Sliced & Silent Files | Invalid Speech | Valid Speech | Accuracy (%) |
|---|---|---|---|---|---|---|---|---|
| 10 Mins | 60 | 1 | 0 | 1 | 0 | 2 | 58 | 96.6 |
| 1 Hour | 693 | 3 | 0 | 5 | 0 | 8 | 685 | 98.8 |
| 5 Hours | 1753 | 13 | 0 | 21 | 0 | 34 | 1719 | 98.06 |

Table 6.5: Audio Split on Silences (10-Second Audio Files)

| URL Data | Audio Extracted | Type of Audio File | | | | Total Time | Segmentation Time |
|---|---|---|---|---|---|---|---|
| | | 1s | 5s | 10s | 15s | | |
| 37 (31Hours) | 21 Hours | 391 | 92 | 37 | 18 | 24.85 Minutes | 5 Minutes |
| 88 (59Hours) | 40 Hours | 930 | 220 | 88 | 43 | 59.25 Minutes | 9 Minutes |
| 130 (81Hours) | 54 Hours | 1375 | 326 | 131 | 64 | 1.46 Hours | 23 Minutes |

Table 6.6: Time Required to Segment Input Audio

- **Discussion**

The standard splitting technique produced unsatisfactory results. According to Table 6.4, the accuracy with 10 minutes, 1 hour, and 5 hours of input data is 13%, 15%, and 21%, respectively. Although increasing the amount of input data improves accuracy, the ratio of output valid speech is still too low (less than 22%). The split-on-silence method produces significantly better results. The split-on-silence technique has an overall accuracy of more than 95%, according to Table 6.5. As a result, we only used this information in our final speech data.

Table 6.2 shows that the final output speech consists of audio files ranging from one second (1s) to fifteen seconds (15s). For each audio file, the percentage of valid speech is given. For instance, it is 99% for one-second, 98% for five-second, 92% for ten-second, and 81% for fifteen-second audio files. The data shows that the length of an audio file influences the segmentation process, i.e. when the length of an audio file increases, the accuracy of valid chunks decreases.

Table 6.6 shows the segmentation time for four different types of audio files: 1s, 5s, 10s, and 15s. It takes 5 minutes to split the speech into vocals and accompaniment tracks from 21 hours of extracted audio. Similarly, 40 hours of audio takes 9 minutes to produce the required audio file chunks, while 54 hours of audio takes 23 minutes.

### 6.2.4 Speech Output for Multiple languages

As previously stated, the model is generic and can be applied to any language. In addition to Pashto, we tested the pipeline for data in Sindhi and Balochi to see how the results did respond. The experiments carried out can be summarized as follows: -

#### 6.2.4.1 Output Speech

- **Results**

We chose Pashto (Ps), Baluchi (Bl), and Sindhi (Sn) to test the pipeline accuracy in terms of output speech for multiple low-resource languages. The input data of 2.30 hours (07 URLs) for Pashto, 2.28 hours (09 URLs) for Baluchi, and 2.26 hours (11 URLs) for Sindhi is provided to the model. Table 6.7 displays the obtained results.

| Input Data | Type | Output Speech (Minutes) | | |
|---|---|---|---|---|
| | | Ps | Bl | Sn |
| Pashto: 07 URLs (02 Hours and 30 Minutes) Baluchi: 09 URLs (02 Hours and 28 Minutes) Sindhi: 11 URLs (02 Hours and 26 Minutes) | 1s | 73 | 61 | 12 |
| | 2s | 94 | 107 | 25 |
| | 3s | 51 | 45 | 12 |
| | 4s | 43 | 52 | 20 |
| | 5s | 55 | 59 | 28 |
| | 6s | 19 | 33 | 19 |
| | 7s | 17 | 23 | 12 |
| | 8s | 27 | 32 | 21 |
| | 9s | 21 | 16 | 08 |
| | 10s | 15 | 19 | 12 |
| | 11s | 9 | 06 | 04 |
| | 12s | 13 | 12 | 20 |
| | 13s | 11 | 13 | 10 |
| | 14s | 7 | 05 | 04 |
| | 15s | 12 | 08 | 06 |
| | Total | 34.58 Minutes | 40.51 Minutes | 23.93 Minutes |

Table 6.7: Output Speech for Pashto, Baluchi, and Sindhi

- **Discussion**

Table 6.7 shows nearly identical speech output data for all three languages. The output for Pashto is 34 minutes when 2.5 hours of URL data is provided as input. Similarly, 2.4 hours of Baluchi input data produces 40 minutes of output speech. For the Sindhi language, the output speech is 23 minutes when 2.3 hours of input data is provided. The results show that there is no significant difference in output speech regardless of the language data supplied to the model as input.

#### 6.2.4.2 Noisy Data

- **Results**

The results for noisy and valid speech data are shown in Table 6.8. Accuracy is calculated in terms of valid speech data for all three languages.

| URLs | Type | Noisy Speech | | | Valid Speech | | | Accuracy (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Ps | Bl | Sn | Ps | Bl | Sn | Ps | Bl | Sn |
| | 1s | | | | 73 | 61 | 12 | 100 | 100 | 100 |
| Pashto: 07 Urls (02 Hours and 30 Minutes) | 2s | | | | 94 | 107 | 25 | 100 | 100 | 100 |
| | 3s | | | | 51 | 45 | 12 | 100 | 100 | 100 |
| | 4s | | | | 43 | 52 | 20 | 100 | 100 | 100 |
| | 5s | | 1 | 1 | 55 | 58 | 27 | 100 | 98.3 | 96.4 |
| | 6s | | | | 19 | 33 | 19 | 100 | 100 | 100 |
| | 7s | 1 | | | 18 | 23 | 12 | 94.7 | 100 | 100 |
| Baluchi: 09 Urls (02 Hours and 28 Minutes) | 8s | | | | 27 | 32 | 21 | 100 | 100 | 100 |
| | 9s | 1 | | | 20 | 16 | 8 | 95.2 | 100 | 100 |
| | 10s | | 1 | 2 | 15 | 18 | 10 | 100 | 94.7 | 83.3 |
| | 11s | | 1 | 1 | 9 | 7 | 3 | 100 | 87.5 | 75 |
| | 12s | 1 | | | 12 | 12 | 20 | 92.3 | 100 | 100 |
| | 13s | 2 | 1 | 1 | 9 | 12 | 9 | 81.8 | 92.3 | 90 |
| Sindhi: 11 Urls (02 Hours and 26 Minutes) | 14s | 1 | 1 | 1 | 6 | 3 | 3 | 85.7 | 75 | 75 |
| | 15s | 2 | 1 | 1 | 10 | 5 | 4 | 83.3 | 83.3 | 80 |
| | Total | **98s** | **68s** | **78s** | | | | | | |

Table 6.8: Comparison of Noisy Speech for Pashto, Baluchi, and Sindhi

- **Discussion**

Table 6.8 shows noisy speech details for all three languages. The collective noisy audio files for Pashto, Baluchi, and Sindhi are 98s, 68s, and 78 seconds respectively. Results show that speech containing noise is independent of language. The ratio of noisy audio files is almost identical for all languages.

### 6.2.4.3 Comparison in Time Consumption

- **Results**

When URL input is provided to the model to generate the final output speech, the time taken by the model to process it is referred to as Time Consumption. Time consumption includes the time required to extract audio from video URLs, the time required to remove noise from raw audio, and the time required for speech segmentation to produce the final output speech, as shown in Table 6.9.

| Language | Input Data | Audio Extraction (Minutes) | Audio Source Separation (Minutes) | Audio Splitting (Minutes) |
|---|---|---|---|---|
| Pashto | 2.30 Hours | 9 | 12 | 1-2 |
| Baluchi | 2.28 Hours | 8 | 10 | 1-2 |
| Sindhi | 2.26 Hours | 7 | 9 | 1-2 |

Table 6.9: Time Consumption for Pashto, Baluchi, and Sindhi

- **Discussion**

Table 10 shows nearly identical results for all three languages. It takes less than ten minutes to extract audio from 2-3 hours of YouTube URL data. Similarly, the time required for noise removal is less than 15 minutes, and audio segmentation takes less than 2 minutes to produce the final output speech. This shows that the model is independent of language video data.

### 6.2.5 Comparison with other Datasets

Table 6.10 shows a quick comparison of our speech data with Urdu and Punjabi datasets from Mozilla Common Voice.

| Details | Urdu | Punjabi | Pashto Speech |
|---|---|---|---|
| Volume | 270 MB | 90.15 MB | 725 MB |
| Size (Hours) | 13 | 4 | 5.9 |
| Voices | 108 | 58 | 137 |
| Audio Format | MP3 | MP3 | Wav |
| Split | Age:<br>19-29 = 71%<br>40-49 = 7%<br>Gender:<br>Male = 61%<br>Female = 21% |  | Age:<br>18-40 = 40%<br>40-60 =35%<br>60+ = 25%<br>Gender:<br>Male = 52%<br>Female = 48% |
| Speech mode | Manual Recorded | Manual Recorded | Online (Youtube) |

Table 6.10: Speech Data Consumption for Pashto, Baluchi, and Sindhi

Chapter 7

# Future Work & Conclusion

## 7.1 Future perspective

Our research establishes a solid foundation for creating speech corpora and provides directions for future work in terms of improving model functionality and output speech quality. First and foremost, future work should address the challenges identified in our research by incorporating techniques to address the issues identified. Second, the model can be improvised to acquire the necessary speech data.

The URL selection process is manual; however, it can be automated by using standard keywords from Wikipedia or other textual sources to search for relevant data on YouTube. The job of source separation is to remove noise from raw audio extracted, but some audio files still contain noise. There is currently no automated mechanism in place to check speech and differentiate between valid and noisy audio files. As a result, a mechanism can be developed to identify problematic audio files and isolate them from valid speech. The time for audio extraction, noise removal, and speech segmentation is manually recorded for each video; this aspect can be automated by incorporating some technique that counts time for these activities. The current Pashto speech corpus is based on two main dialects as well as the written text. More speech data for multiple dialects mapped to multiple textual formats can be created. There is currently no predefined condition on the model that generates a specific amount of output speech for a specified number of audio chunks. The above scenario can be implemented to produce speech for predefined audio hours and speech segment file requirements.

## 7.2 Conclusion

Automatic speech recognition (ASR) for low-resource languages improves linguistic minorities' access to technological advantages. The first step in developing an ASR system is to generate a transcribed speech corpus. Traditional ASR models require massive amounts of speech data, whereas the Meta AI self-supervised concept allows us to train an ASR system with only a few hours of speech data. However, generating such transcribed speech is a widespread issue for low-resource languages. The primary problem is the collection of speech data. Recording and crowdsourcing are popular methods, but they are expensive for a low-resource language. As a result, this research contributes to the development of a model for automated online speech data retrieval. The model can generate voice segments suitable for speech recognition tasks in a low-resource language by utilizing video or audio-sharing platforms such as YouTube. A review of existing work in the field of speech recognition for the Pashto language has also been conducted. An ASR dataset is created using YouTube resources to address the issue of speech data scarcity in the Pashto language.

# Notations

# References

[1] Polat, Huseyin, and Saadin Oyucu. "Building a speech and text corpus of Turkish: Large corpus collection with initial speech recognition results." Symmetry 12.2 (2020): 290.

[2] Papastratis, Ilias "Speech Recognition: a review of the different deep learning approaches", theaisummer.com, Year "2021",

[3] Paramonov, P.; Sutula, N. “Simplified scoring methods for HMM-based speech recognition” Soft Comput. 2016,20, 3455–3460. [CrossRef]

[4] Mahanty, R.; Mahanti, P.K. Unleashing artificial intelligence onto big data: A review. In Research on Computational Intelligence Applications in Bioinformatics, 1st ed.; Dash, S., Subudhi, B., Eds.; IGI Global: Hershey, PA, USA, 2016; pp. 1–16.

[5] Aggarwal, R.; Dave, M. “Acoustic modeling problem for automatic speech recognition system: Advances and refinements (Part II)”. Int. J. Speech Technol. 2011, 14, 1572–8110. [CrossRef]

[6] [2201.02419v2] Automatic Speech Recognition Datasets in Cantonese: A Survey and New Dataset (arxiv.org)

[7] “What is Automatic Speech Recognition?” blog by voximplant dated: 2020-10-01. Blog Link: https://voximplant.com/blog/what-is-automatic-speech-recognition

[8] Zhang, Biao, Barry Haddow, and Rico Sennrich. "Revisiting End-to-End Speech-to-Text Translation From Scratch." arXiv preprint arXiv:2206.04571 (2022).

[9] Li, Jinyu. "Recent advances in end-to-end automatic speech recognition." APSIPA Transactions on Signal and Information Processing 11.1 (2022).

[10] Perero-Codosero, Juan M., Fernando M. Espinoza-Cuadros, and Luis A. Hernández-Gómez. "A Comparison of Hybrid and End-to-End ASR Systems for the IberSpeech-RTVE 2020 Speech-to-Text Transcription Challenge." Applied Sciences 12.2 (2022): 903.

[11] Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.R.; Jaitly, N.; Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Mag. 2012, 29, 82–97. [CrossRef]

[12] Prabhavalkar, R.; Rao, K.; Sainath, T.N.; Li, B.; Johnson, L.; Jaitly, N. A Comparison of Sequence-to-Sequence Models for Speech Recognition. Interspeech 2017, 2017, 939–943. [CrossRef]

[13] Rao, Kanishka, Haşim Sak, and Rohit Prabhavalkar. "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer." 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). IEEE, 2017.

[14] He, Y.; Sainath, T.N.; Prabhavalkar, R.; McGraw, I.; Alvarez, R.; Zhao, D.; Gruenstein, A. Streaming end-to-end speech recognition for mobile devices. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 6381–6385.

[15] Li, J.; Zhao, R.; Meng, Z.; Liu, Y.; Wei, W.; Parthasarathy, S.; Gong, Y. Developing RNN-T models surpassing high-performance hybrid models with customization capability. arXiv 2020, arXiv:2007.15188.

[16] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. In Proc. of Interspeech, 2019.

[17] G. Synnaeve et al. End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures. arXiv, abs/1911.08460, 2019.

[18] W. Han et al. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context. arXiv, 2020.

[19] Gulati, Anmol, et al. "Conformer: Convolution-augmented transformer for speech recognition." arXiv preprint arXiv:2005.08100 (2020).

[20] Simons, Gary F., and Charles D. Fennig. "Ethnologue: Languages of the World, Dallas, Texas: SIL International." Online version: http://www. ethnologue. com (2017).

[21] A. H. Liu, H.-Y. Lee, and L.-S. Lee. Adversarial training of end-to-end speech recognition using a criticizing language model. arXiv, 2018.

[22] Baskar, Murali Karthick, et al. "Semi-supervised sequence-to-sequence ASR using unpaired speech and text." arXiv preprint arXiv:1905.01152 (2019).

[23] W.-N. Hsu, A. Lee, G. Synnaeve, and A. Hannun. Semi-supervised speech recognition via local prior matching. arXiv, 2020.

[24] H. Scudder. Probability of error of some adaptive pattern-recognition machines. IEEE Trans. on Inform. Theory, 1965.

[25] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In Proc. of ACL, 1995.

[26] E. Riloff. Automatically generating extraction patterns from untagged text. In Proc. of AAAI, 1996.

[27] Mathur, Chetan N., and K. P. Subbalakshmi. "Security issues in cognitive radio networks." *Cognitive Networks: Towards Self-Aware Networks* (2007): 271-291.

[28] E. Riloff. Automatically generating extraction patterns from untagged text. In Proc. of AAAI, 1996.

[29] S. H. K. Parthasarathi and N. Strom. Lessons from building acoustic models with a million hours of speech. arXiv, 2019.

[30] J. Kahn, A. Lee, and A. Hannun. Self-training for end-to-end speech recognition. In Proc. of ICASSP, 2020.

[31] Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, and R. Collobert. Iterative pseudo-labeling for speech recognition. arXiv, 2020.

[32] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, and et al. Improved noisy student training for automatic speech recognition. arXiv, 2020.

[33] Alexei Baevski Henry Zhou Abdelrahman Mohamed Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. arxiv.org 2020.

[34] Krauwer, Steven. "The basic language resource kit (BLARK) as the first milestone for the language resources roadmap." *Proceedings of SPECOM*. Vol. 2003. No. 8. 2003.

[35] Berment, Vincent. *Méthodes pour informatiser les langues et les groupes de langues «peu dotées»*. Diss. Université Joseph-Fourier-Grenoble I, 2004.

[36] Young, Steve J., and Lin Lawrence Chase. "Speech recognition evaluation: a review of the US CSR and LVCSR programmes." *Computer Speech & Language* 12.4 (1998): 263-279.

[37] Kincaid, Jason, "A Brief History of ASR: Automatic Speech Recognition" July 2018

[38] Conneau, Alexis, et al. "Unsupervised cross-lingual representation learning for speech recognition." *arXiv preprint arXiv:2006.13979* (2020).

[39] Gelas, Hadrien, et al. "Quality assessment of crowdsourcing transcriptions for African languages." *Twelfth Annual Conference of the International Speech Communication Association*. 2011.

[40] Schultz, Tanja, and Alex Waibel. "Language-independent and language-adaptive acoustic modeling for speech recognition." Speech Communication 35.1-2 (2001): 31-51.

[41] Le, Viet-Bac, and Laurent Besacier. "Automatic speech recognition for under-resourced languages: application to Vietnamese language." IEEE Transactions on Audio, Speech, and Language Processing 17.8 (2009): 1471-1482.

[42] Lopes, Carla, and Fernando Perdigao. "Phone recognition on the TIMIT database." Speech Technologies/Book 1 (2011): 285-302.

[43] Paul, Douglas B., and Janet Baker. "The design for the Wall Street Journal-based CSR corpus." Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992. 1992.

[44] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: telephone speech corpus for research and development," in Proc. of IEEE ICASSP, 1992.

[45] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An asr corpus based on public domain audio books. In 2015 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 5206–5210, 2015. doi: 10.1109/ICASSP.2015.7178964.

[46] G. Chen, S. Chai, G. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang, M. Jin, S. Kudanpur, S. Watanabe, S. Zhao, W. Zou, X. Li,

X. Yao, Y. Wang, Z. You, and Z. Yan. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. Interspeech 2021, 2021.

[47] Ardila, Rosana, et al. "Common voice: A massively-multilingual speech corpus." arXiv preprint arXiv:1912.06670 (2019).

[48] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert. Mls: A large-scale multilingual dataset for speech research. Interspeech 2020, Oct 2020. doi: 10.21437/interspeech.2020-2826. URL http: //dx.doi.org/10.21437/Interspeech.2020-2826.

[49] Bu, H., Du, J., Na, X., Wu, B., and Zheng, H. (2017). Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, pages 1–5. IEEE.

[50] Ali, Hazrat, Nasir Ahmad, and Abdul Hafeez. "Urdu speech corpus and preliminary results on speech recognition." *International conference on engineering applications of neural networks*. Springer, Cham, 2016.

[51] Raza, Agha Ali; Hussain, Sarmad; Sarfraz, Huda; Ullah, Inam; Sarfraz, Zahid (2009). *[IEEE 2009 Oriental COCOSDA International Conference on Speech Database and Assessments - Urumqi, China (2009.08.10-2009.08.12)] 2009 Oriental COCOSDA International Conference on Speech Database and Assessments - Design and development of phonetically rich Urdu speech corpus., (), 38–43.* doi:10.1109/ICSDA.2009.5278380

[52] Sameti, Hossein, et al. "A large vocabulary continuous speech recognition system for Persian language." *EURASIP Journal on Audio, Speech, and Music Processing* 2011.1 (2011): 1-12.

[53] Yazdani, Ali, Hossein Simchi, and Yasser Shekofteh. "Emotion Recognition In Persian Speech Using Deep Neural Networks." *2021 11th International Conference on Computer Engineering and Knowledge (ICCKE)*. IEEE, 2021.

[54] Tiwari, Sonal A., Rajashri G. Kanke, and A. Maheshwari. "Marathi Speech Database Standardization: A Review and Work." *International Journal of Computer Science and Information Security (IJCSIS)* 19.7 (2021).

[55] Mon, Aye Nyein, Win Pa Pa, and Kyaw Thu Ye. "UCSY-SC1: A Myanmar speech corpus for automatic speech recognition." *International Journal of Electrical and Computer Engineering* 9.4 (2019): 3194.

[56] Basu, Joyanta, et al. "Multilingual speech corpus in low-resource eastern and northeastern Indian languages for speaker and language identification." *Circuits, Systems, and Signal Processing* 40.10 (2021): 4986-5013.

[57] Kumar, Ritesh, et al. "Annotated Speech Corpus for Low Resource Indian Languages: Awadhi, Bhojpuri, Braj and Magahi." *arXiv preprint arXiv:2206.12931* (2022).

[58] Choi, Yoona, and Bowon Lee. "Pansori: ASR corpus generation from open online video contents." *arXiv preprint arXiv:1812.09798* (2018).

[59] Takamichi, Shinnosuke, et al. "JTubeSpeech: corpus of Japanese speech collected from YouTube for speech recognition and speaker verification." *arXiv preprint arXiv:2112.09323* (2021).

[60] Sikasote, Claytone, and Antonios Anastasopoulos. "BembaSpeech: A Speech Recognition Corpus for the Bemba Language." *arXiv preprint arXiv:2102.04889* (2021).

[61] Alam, Samiul, et al. "Bengali common voice speech dataset for automatic speech recognition." *arXiv preprint arXiv:2206.14053* (2022).

[62] Yu, Tiezheng, et al. "Automatic Speech Recognition Datasets in Cantonese: A Survey and New Dataset." *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. 2022.

[63] Ahmed, Irfan, et al. "The development of isolated words corpus of pashto for the automatic speech recognition research." *2012 International Conference of Robotics and Artificial Intelligence*. IEEE, 2012.

[64] Wahab, Mehreen, Hassan Amin, and Farooq Ahmed. "Shape analysis of pashto script and creation of image database for OCR." *2009 International Conference on Emerging Technologies*. IEEE, 2009.

[65] Abbas, Arbab Waseem, Nasir Ahmad, and Hazrat Ali. "Pashto Spoken Digits database for the automatic speech recognition research." *18th International Conference on Automation and Computing (ICAC)*. IEEE, 2012.

[66] Trmal, Jan, et al. "Using of heterogeneous corpora for training of an ASR system." *arXiv preprint arXiv:1706.00321* (2017).

[67] Parent, Gabriel, and Maxine Eskenazi. "Toward better crowdsourced transcription: Transcription of a year of the let's go bus information system data." *2010 IEEE Spoken Language Technology Workshop*. IEEE, 2010.

[68] Hennequin, Romain, et al. "Spleeter: a fast and efficient music source separation tool with pre-trained models." *Journal of Open Source Software* 5.50 (2020): 2154.

[69] Tom Bäckström and Okko Räsänen and Abraham Zewoudie and Pablo Pérez Zarazaga and Liisa Koivusalo and Sneha Das and Esteban Gómez Mellado and Marieum Bouafif Mansali and Daniel Ramos, "Introduction to Speech Processing" edition 2, 2022.

[70] Ishizaka, Kenzo, and James L. Flanagan. "Synthesis of voiced sounds from a two-mass model of the vocal cords." Bell system technical journal 51.6 (1972): 1233-1268.

[71] Atal, B., and L. Rabiner. "A pattern recognition approach to voiced-unvoiced-silence classification with applications to speech recognition." IEEE Transactions on Acoustics, Speech, and Signal Processing 24.3 (1976): 201-212.

[72] Jalil, Madiha, Faran Awais Butt, and Ahmed Malik. "Short-time energy, magnitude, zero crossing rate and autocorrelation measurement for discriminating voiced and unvoiced segments of speech signals." 2013 The international conference on technological advances in electrical, electronics, and computer engineering (TAEECE). IEEE, 2013.

[73] Wang, Zhe. "Audio Signal Acquisition and Processing System Based on Model DSP Rapid Design." Security and Communication Networks 2022 (2022).

[74] Aajiz, N.M., and Pashto Academy Publications, “Bilingual primer Pashto - English”

[75] Rahman, Tariq. "Language policy, multilingualism and language vitality in Pakistan." Trends in linguistics studies and monographs 175 (2006): 73.

[76] Lorigo, Liana, and Venu Govindaraju. "Segmentation and pre-recognition of Arabic handwriting." Eighth International Conference on Document Analysis and Recognition (ICDAR'05). IEEE, 2005.

[77] Ardila, Rosana, et al. "Common voice: A massively-multilingual speech corpus." arXiv preprint arXiv:1912.06670 (2019).