System Requirements Specification

for

RTube Kaldi Research Team

Version 3.0

Prepared by

Tabitha O'Malley, Milan Haruyama, David Serfaty,

Tahmina Tisha, Adam Gallub

CS 490 RTube Kaldi Research Team Fall 2023

11/21/2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Tabitha, Milan, Tisha, David, Adam, Max	09/29/23	Starting the document	V1.0
Tabitha	10/25/23	Formatting Revision History Editing/Formatting Appendix Writing Section: 1.5	V2.1
Tabitha	10/26/23	Writing Sections: 1.2, 2.2, 2.5, 3.1	V2.2
Tabitha	10/27/23	Writing Requirements: 3.1	V2.3
Milan	10/27/23	Writing and Editing Requirements: 3.1	V2.4
David	10/27/23	Writing and Editing Requirements: 3.1	V2.5
Tabitha	10/29/23	Writing Section: 2.3, 2.4, 2.5	V2.6
David	10/29/23	Writing Section:2.3, 2.4, 2.5	V2.7
Tabitha	10/30/23	Writing Section: 4, 4.1, 4.2, 4.3, 4.4, 5.1, 5.2, 3	V2.8
Milan	10/30/23	Editing all Sections Writing Section: 5.1, 5.2, 2.1	V2.9
David	10/30/23	Writing Section: 2.1, 5.3	V2.10
Tabitha	10/31/23	Update Model Editing Sections	V2.11
Milan	10/31/23	Editing all sections	V2.12
David	10/31/23	Editing, 2.1, 3, Appendix A, 4.1, 4.2, 4.1.3, 4.2.3, 4.3, 4.3.3, 2.2 Writing Section: 5.2	V2.13
Adam	11/05/23	Writing Section: 2.4 and 2.6	V3.1
Milan	11/05/23	Editing all sections	V3.2
Tabitha	11/05/23	Rewrite: 2.3 Add to: 2.2	V3.3
David	11/05/23	Class Model: 2.3 Add to: 2.2	V3.4

Tisha	11/05/2023	Editing: 2.4, 2.5	V3.5
Tabitha	11/07/23	Adding/Editing: 3	V3.6
Milan	11/07/2023	Adding/Editing: 3	V3.7
David	11/07/2023	Adding/Edition: 3	V3.8
Adam	11/07/2023	Writing/Editing 2.6	V3.9
Tabitha	11/11/2023	Editing: 3	V3.10
Milan	11/11/2023	Editing: 3	V3.11
Adam	11/11/2023	Editing/Writing: 4.1.1	V3.12
Tabitha	11/14/2023	Update Requirements	V3.13
Adam	11/15/2023	Editing/Writing: 4	V3.14
Tisha	11/15/2023	Editing section: 5	V3.15
Tabitha	11/15/2023	Editing: 1.5, 2.3, 4.1, 5.1	V3.16
Tabitha	11/16/2023	Review/Edition/Commenting Section: 4 Update Class Diagram Update/Review Section: 2.3	V3.17
Tisha	11/18/2023	Rewriting section 5.2	V3.18
Tisha	11/18/2023	Edited section 5.2	V3.19
Milan	11/18/2023	Editing all sections	V3.20
David	11/19/2023	Adding/Editing/Rewriting: 4, 5.1, 5.2 Editing 2.2 Appendix B	V3.21
Tabitha	11/19/2023	Adding/Editing/Rewriting: 4, 5.1, 5.2 Appendix A, B	V3.22
Milan	11/19/2023	Editing all sections	V3.23
Milan	11/20/2023	Reviewing/editing all sections	V3.24

1. Introduction

1.1 Purpose

The RTube web application shall provide an interface for users to track the flight paths of aircrafts, and to listen to live Air Traffic Control (ATC) tower radio transmissions with the option to transcribe speech to text in real time. The live speech-to-text transcription is performed using an ASR model developed using the Kaldi ASR toolkit, which is the focus of this document.

1.2 Document Conventions

Primary Body	**Times New Roman, 11 point**
Section Headers	**Times New Roman, 18 point, bold**
Subsection Header	**Times New Roman, 14 point, bold**
In-Progress	**Cyan highlight**
Pending Review/Update	**Yellow highlight**
Complete Overhaul	**Red highlight**
Completed	**Green highlight**
Important Terms	**Bold text within document body**

1.3 Intended Audience and Reading Suggestions

The intended audience for the RTube web application is aircraft pilots and ATC operators. One of the more specific applications for RTube shall be for Embry-Riddle Aeronautical University (ERAU) instructors and student pilots to evaluate communications between the student and ATC operators.

In order to improve understanding of this document and the RTube project itself, it is highly recommended to read documentation related to phonetics, aviation phraseology, ASR, NLP, and the Kaldi ASR Toolkit.

1.4 Product Scope

As an aircraft pilot, learning to communicate with Air Traffic Control (ATC) is a daunting task. Despite being designed to mitigate miscommunication, aviation phraseology is highly intricate and requires hundreds of hours of training to learn its idiosyncrasies. While there exist a few resources (such as the website LiveATC) for student pilots to study aviation phraseology, these resources do little to accommodate the major learning curve present, especially since they do not provide live transcriptions of speech for student pilots to read.

As such, the RTube web application shall bridge the learning gap faced by many student pilots by providing the ability to transcribe live ATC transmissions into text in real-time, as well as providing a live interface for users to track the flight paths of aircraft. The live speech-to-text transcription is performed using an automatic speech recognition (ASR) model developed using the Kaldi ASR toolkit. With the ability to transcribe speech to text in real-time, student pilots can dramatically reduce the amount of training required to understand spoken aviation phraseology. In addition, the live interface helps students better understand different contexts in which specific phrases are used.

1.5 References

Fagan, Jonathan. "What Is a Good Accuracy Level for Transcription?" *University Transcription Services*, 3 June 2020, www.universitytranscriptions.co.uk/what-is-a-good-accuracy-level-for-transcription/. Accessed 26 Oct. 2023

Kaldi Research Team Product Vision Statement. Kaldi Research Team. 19 September 2023. Kaldi Research Team Drive.

Kaldi Research Team Software Design Document. Kaldi Research Team. 19 November 2023. Kaldi Research Team Drive.

Ravihara, Ransaka. "Gaussian Mixture Model Clearly Explained." *Medium*, Towards Data Science, 11 Jan. 2023, www.towardsdatascience.com/gaussian-mixture-model-clearly-explained-115010f7d4cf/.

"About Pandas." *Pandas*, www.pandas.pydata.org/about/. Accessed 26 Oct. 2023.

"LiveATC FAQ." *LiveATC.Net - Listen to Live Air Traffic Control over the Internet!*, www.liveatc.net/faq/. Accessed 26 Oct. 2023.

"A Python Library to Read/WRITE EXCEL 2010 Xlsx/XLSM Files¶." *Openpyxl*, www.openpyxl.readthedocs.io/en/stable/. Accessed 26 Oct. 2023.

"What Is Kaldi?" *Kaldi*, www.kaldi-asr.org/doc/about.html/. Accessed 26 Oct. 2023.

"What Is Speech Recognition?" *IBM*, www.ibm.com/topics/speech-recognition/. Accessed 26 Oct. 2023.

Chester F. Carlson Center for Imaging Science | College of Science | RIT, www.cis.rit.edu/class/simg716/Gauss_History_FFT.pdf/. Accessed 1 Nov. 2023.

"About FFmpeg" *FFmpeg,* www.ffmpeg.org/about.html/. Accessed 19 Nov. 2023.

2. Overall Description

2.1 Product Perspective

The product developed by the Kaldi Research Team shall be an ASR model created using the Kaldi ASR Toolkit. The toolkit was first produced by Johns Hopkins University in 2007 and comes packaged with a sample ASR model that serves as a baseline for this project. Though a self-contained product on its own, the ASR model developed by the Kaldi Research Team shall be combined in the future with the NeMo project in the form of the RTube web application. The RTube web application shall use the ASR model to transcribe live ATC transmissions in real-time. Since this project solely focuses on ASR research and development, the scope is significantly more limited than that of the NeMo project.

2.2 Product Functions

Func. 1	Receive WAV files for processing
Func. 2	Transcribe audio into text
Func. 3	Output a text file with the transcribed words from the audio file.

2.3 User Classes and Characteristics

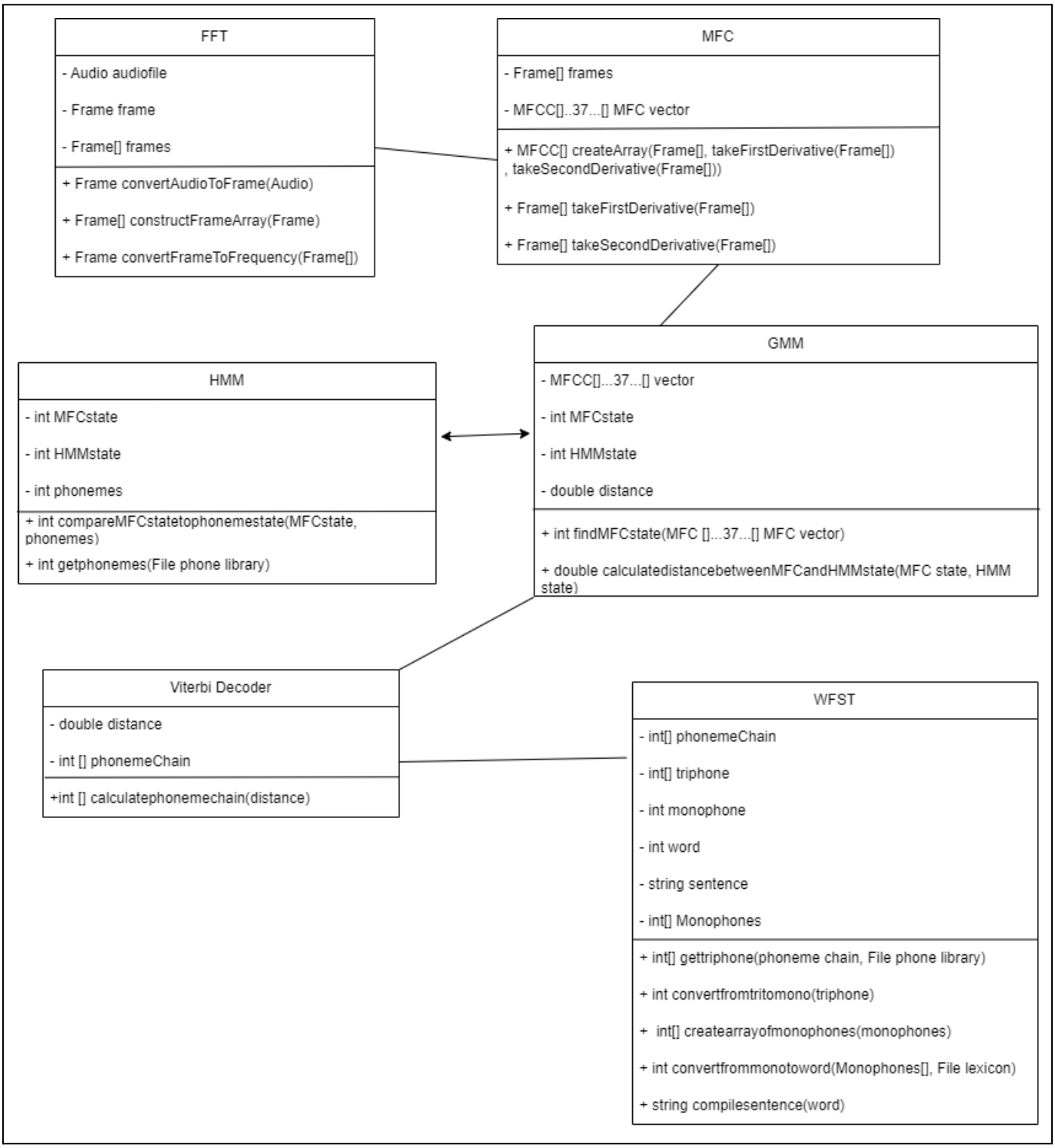

Figure 1: Kaldi Toolkit Sample ASR Model Class Diagram V3.2

Fast Fourier Transform (FFT): responsible for converting audio frames from the time-domain to the frequency-domain.

- Variables: Audio audioFile, Frame frame, Frame[] frames
- Methods: convertAudioToFrame(Audio), constructFrameArray(Frame), convertFrameToFrequency(Frame [])
- Inputs: WAV file
- Outputs: frequency-domain frames

Mel-Frequency Cepstrum (MFC): converts frequency-domain frames into a 39-dimensional MFCC feature vector represented as a 39-dimensional array.

- Variables: Frame[] frames, MFCC[]... 37…[]
- Methods: create array (Frame [], takeFirstDerivative(Frame []), takeSecondDervivative(Frame[])), takeFirstDerivative(Frame []), takeSecondDervivative(Frame[])
- Inputs: frequency-domain frames
- Outputs: 39-dimensional MFCC feature vector

Gaussian Mixture Model (GMM): finds the distance between the MFCC feature vector and the phoneme states.

- Variables: MFCC[]...37…[] MFC, int MFCstate, int HMMstate, double distance
- Methods: findMFCstate(MFCC[]...37…[] MFC), calculatedistancebetweenMFCandHMMstate(MFCstate, HMMstate)
- Inputs: MFC feature vector
- Outputs: Distance

Hidden Markov Model (HMM): finds the HMM state closest to the MFC state.

- Variables: double distance, int HMMstate, in phonemes
- Methods: compareMFCstatetophonemestate(MFCstate, phonemes), getphonemes(File phone library)
- Inputs: MFC state
- Outputs: HMM state

Viterbi Decoder: finds the phoneme chain based on the previously calculated distance.

- Variables: double distance, int HMMstate, in phonemes
- Methods: compareMFCstatetophonemestate(MFCstate, phonemes), getphonemes(File phone library)
- Inputs: MFC state
- Outputs: Phoneme chain

WFST (Weighted Finite State Transducer): responsible for converting the phoneme chain into sentences.

- Variables: int[] phonemeChain, int[] triphone, int monophone, int word, string sentence, int[] Monophones
- Methods: gettriphone(phonemeChain, File phone library), convertfromtritomono(triphone),createarrayof monophones(monophones), convertfrommonotoword(Monophones[], File lexicon), compilesentence(word)
- Inputs: Phoneme chain
- Outputs: Sentences

2.4 Operating Environment

The project currently utilizes the ASR model provided by the Kaldi ASR Toolkit. The ASR model runs on a command-line interface via the Windows Subsystem for Linux (WSL). The specific operating system needed to install and run the interface is Ubuntu, a free and open-source Linux distribution. The Kaldi ASR Toolkit also uses code-level integration with finite state transducers (FSTs), and compiles using the OpenFst Toolkit.

2.5 Design and Implementation Constraints

Design Constraints:

- Minimum of 12 GB of VRAM for ASR model to run within an hour.
- Minimum storage size of 12.5 GB (13,479,809,024 bytes).

Implementation Constraints:

- Input shall be a WAV file
- Audio shall be in General American English.
- No punctuation or grammar marks shall be transcribed

2.6 Assumptions and Dependencies

The development of the ASR model to be used in the RTube web application is constrained by several factors. These include but are not limited to: the assumption that all speech from aircraft pilots and ATC is both clear and direct (i.e., speakers are not soft-spoken, speakers do not mumble their words, et cetera); the assumption that all audio input shall be in General American English; the assumption that the audio to be transcribed contains low radio interference and background noise (e.g., interfering signals, loud engine noise, et cetera); the assumption that ATC transmissions are obtained from the Daytona Beach International Airport (DAB); the assumption that the RTube web application is constrained to the state of Florida; and the assumption that 30 hours of training data is sufficient to produce a transcription accuracy of at least 80%.

The development of the aforementioned ASR model is also dependent on the developers having computers with enough video memory and storage to both store and run the Kaldi ASR toolkit to develop the ASR model.

3. Software Interface Requirements

Data Preparation		
WAV file	Req. 1.1.1	The system shall only accept WAV files as input.
	Req 1.1.2	The system shall convert other file types into WAV files using the FFMPEG Linux utility.
Fast Fourier Transform (FFT)	Req. 2.1.1	The system shall partition the WAV file into overlapping 25-millisecond frames.
	Req. 2.2.1	The system shall obtain each frame in 10-millisecond sliding intervals starting from time equals 0.
	Req. 2.2.2	The system shall use the formula $[10n,\ 10n\ +\ 25]$ to obtain the closed time intervals of each frame (e.g., $[0,\ 25],\ [10,\ 35],\ [20,\ 45],\ ...$).
	Req. 2.2.3	The system shall define the value n as any non-negative integer (i.e., $\{n \in \mathbb{Z} : n \geq 0\}$).
	Req. 2.3.1	The system shall use a FFT to convert the audio data within the frames from the time-domain to the frequency domain.
Mel-Frequency Cepstrum (MFC) & Mel-Frequency Cepstral Coefficients (MFCCs)	Req. 3.1.1	The system shall convert the results of the FFT into MFCCs
	Req. 3.2.1	The system shall convert the frequency-domain data in each frame into a set of 13 MFCCs.
	Req. 3.2.2	The system shall derive the first-order derivative of the 13 MFCCs.
	Req. 3.2.3	The system shall derive the second derivative of the 13 MFCCs.
	Req. 3.2.4	The system shall produce a 39-dimensional MFC represented as an MFCC feature vector.
	Req. 3.2.5	The system shall use the first set of thirteen dimensions (i.e., [0, 12]) to represent the thirteen MFCCs.
	Req. 3.2.6	The system shall use the second set of thirteen dimensions (i.e., [13, 25]) to represent the first-order derivatives of the MFCCs.
	Req 3.2.7	The system shall use the third set of thirteen dimensions (i.e., [26, 38]) to represent the second-order derivatives of the MFCCs.
	Req. 3.3.1	The system shall use the formula $[10n,\ 10n\ +\ 25]$ to obtain the MFCC vectors for the corresponding time-domain segment (e.g., $[0,\ 25],\ [10,\ 35]$, ...).
	Req. 3.4.1	The system shall maintain the order of the frames.
Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM)	Req. 4.1.1	The system shall use a GMM to represent the MFCC vector of each frame.
	Req. 4.1.2	The system shall use a HMM to represent the relationship between the MFCC vector and the phonemes.
	Req. 4.1.3	The system shall use a HMM to represent the phonemes, which are unobservable and evolve in a chain order.
	Req. 4.1.4	The MFCC vector shall be emitted following the GMM probability model

		from the hidden phonemes.
	Req 4.1.5	The system shall use triphones instead of monophones to more accurately model the phonemes.
Decoding Operations		
Viterbi Decoder	Req. 5.1.1	The system shall use a Viterbi decoder to find a chain of phoneme states.
	Req. 5.1.2	The system shall input the distances between the MFCC vector and the GMM probability model into the Viterbi decoder.
Weighted Finite State Transducer (WFST)	Req. 6.1.1	The system shall use four WFST models for further processing.
Hidden Markov Model (HMM)	Req 7.1.1	The system shall input the output of the Viterbi decoder into a HMM to output triphones.
Context-Dependency Model	Req. 8.1.1	The system shall input triphones to convert the triphone states into monophones.
Lexicon	Req. 9.1.1	The system shall use the lexicon to convert monophones into words.
Language Model	Req. 10.1.1	The system shall use the words from the lexicon out to score the next predicted word.
WFST	Req. 11.1.1	The system shall combine the four aforementioned models into a single simplified model.
Text file	Req. 12.1.1	The system shall output a text file upon completion of all aforementioned operations.
	Req. 12.1.2	The system shall only write transcribed words into the aforementioned text file.
	Req. 12.2.1	The system shall not write punctuation or grammatical marks into the aforementioned text file.
	Req. 12.3.1	The system shall transcribe words found in the lexicon in uppercase.
	Req 12.3.2	The system shall transcribe words not found in the lexicon in lowercase.

4. System Features

The following section details the system features described in Section 2.2 of this document. The section begins with the reception of the audio file, followed by the processing of the file, the transcription of the audio into text, the output of the transcribed text, and the storing of unknown words into a non-lexicon library.

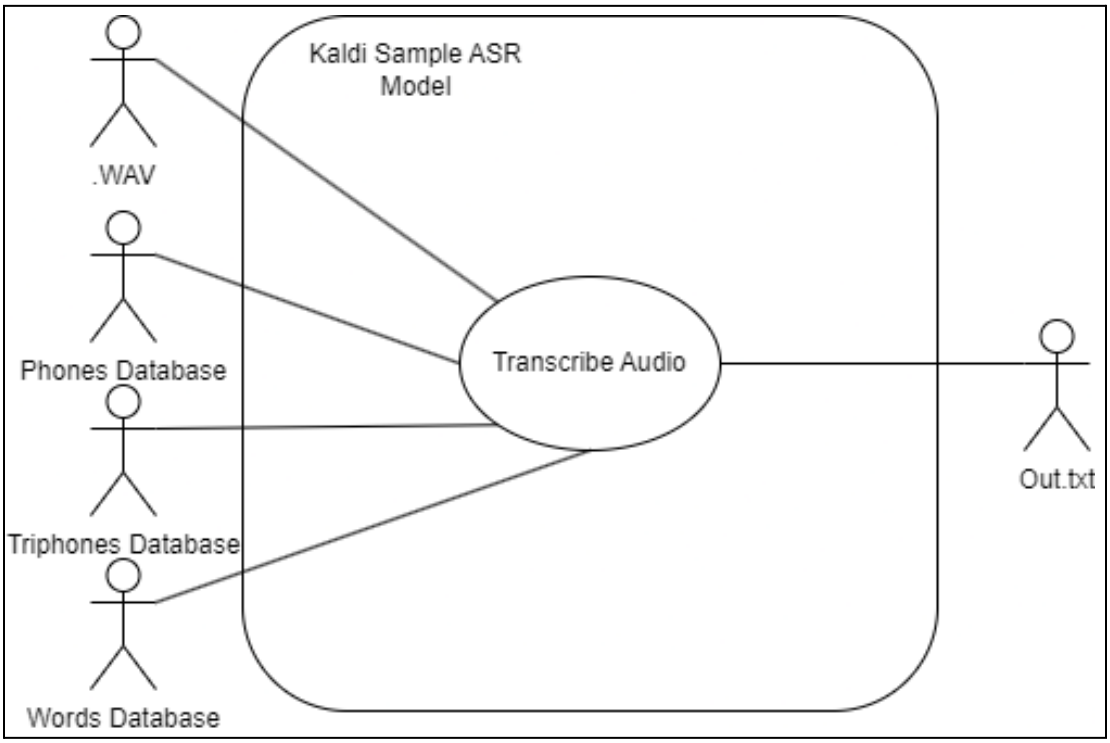

Figure 2: Kaldi ASR Toolkit Sample ASR Model Use Case Diagram V2.1.1

4.1 Reception of the audio file for processing

4.1.1 Description and Priority

Reception of the audio file for processing from the user via a Python3 script is of low priority. Input shall be given as a terminal command in the format, "python3 main.py filename.wav" (*Figure 3*). The argument, "python3", calls the execution of a Python3 file. The file in question, "main.py, is a script that initiates the execution of the sample ASR model. Lastly, the argument, "filename.wav", calls the WAV file that shall be transcribed by the sample ASR model. The file shall be located in the same directory as the script. In essence, the script shall only execute if it is called alongside a single WAV file in the Ubuntu terminal (*Figure 3*). Any input that does not adhere to this format shall cause the system to throw an exception and print a message in the terminal (*Figures 4, 5, 6*), explained in detail in the proceeding subsection.

4.1.2 Stimulus/Response Sequences

Calling any file type other than WAV shall cause the script to throw an exception and print, "Provided filename does not end in '.wav'" (*Figure 4*). Calling multiple files of any type shall cause the script to throw an exception and print, "Too many arguments provided. Aborting" (*Figure 5*). In the absence of a WAV file, the system shall attempt a traceback to the most recently called WAV file; if unsuccessful, the system shall throw a "ValueError" exception and print, "No .wav file in the root directory" (*Figure 6*).

It is highly advised that the user converts other audio file types (e.g., FLAC) into WAV beforehand using the FFmpeg Linux utility. The format for the terminal command is "ffmpeg filename.filetype filename.wav" (*Figure 7*). The argument, "ffmpeg", calls the execution of the FFmpeg utility. The "-i" flag denotes that the next argument, "filename.filetype", is the input audio file. Lastly, the result of the conversion is specified by the argument, "filename.wav", which shall ideally have the same filename as the input file. Once executed, the utility converts the audio file into a WAV file, and the user shall be able to execute the aforementioned Python3 script without issue (*Figure 3*).

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py gettysburg.wav
tree-info exp/chain_cleaned/tdnn_1d_sp/tree
tree-info exp/chain_cleaned/tdnn_1d_sp/tree
```

Figure 3: Sample ASR Model Successful Input (with code execution snippet)

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py gettysburg.flac
Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 16, in <module>
    raise ValueError("Provided filename does not end in '.wav'")
ValueError: Provided filename does not end in '.wav'
```

Figure 4: Sample ASR Model Unsuccessful Input; non-WAV input exception

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py gettysburg.wav gettysburg.wav
Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 18, in <module>
    raise ValueError("Too many arguments provided. Aborting")
ValueError: Too many arguments provided. Aborting
```

Figure 5: Sample ASR Model Unsuccessful Input; too many arguments exception

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py
Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 10, in <module>
    FILE_NAME_WAV = glob.glob("*.wav")[0]
IndexError: list index out of range

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 12, in <module>
    raise ValueError("No .wav file in the root directory")
ValueError: No .wav file in the root directory
```

Figure 6: Sample ASR Model Unsuccessful Input; no file in directory exception

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ ffmpeg -i gettysburg.flac gettysburg.wav
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enable-gpl
--disable-stripping --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --enable-lib
dav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-
libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librabbitmq --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --
enable-libsrt --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-l
ibxvid --enable-libzimg --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-pocketsphinx --enable-librsvg -
-enable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, flac, from 'gettysburg.flac':
  Metadata:
    encoder         : Lavf58.76.100
  Duration: 00:00:10.00, start: 0.000000, bitrate: 178 kb/s
  Stream #0:0: Audio: flac, 22050 Hz, mono, s16
Stream mapping:
  Stream #0:0 -> #0:0 (flac (native) -> pcm_s16le (native))
Press [q] to stop, [?] for help
Output #0, wav, to 'gettysburg.wav':
  Metadata:
    ISFT            : Lavf58.76.100
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 22050 Hz, mono, s16, 352 kb/s
    Metadata:
      encoder         : Lavc58.134.100 pcm_s16le
size=     431kB time=00:00:09.92 bitrate= 355.6kbits/s speed= 769x
video:0kB audio:431kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.017682%
```

Figure 7: Execution of FFmpeg utility; conversion of FLAC to WAV

4.1.3 Functional Requirements

Req. 4.1.3.1	The script shall throw an exception if any file type other than WAV is called.
Req. 4.1.3.2	The script shall throw an exception if multiple files are called.
Req. 4.1.3.3	The script shall attempt a traceback of the most recently transcribed WAV file if a WAV file is not called.
Req. 4.1.3.4	The script shall throw an exception upon the failure of the traceback.
Req. 4.1.3.5	The system shall initiate the transcription process upon the successful execution of the script
Req 4.1.3.6	The system shall only accept input from a command-line interface (CLI).

4.2 Transcription of audio into text

4.2.1 Description and Priority

The ASR model transcribes audio into text by utilizing a database of phones, triphones, and words, as well as a series of models for each database (*Figure 8*). For this project, transcription is a high-priority feature. The system shall prepare the data for transcription by partitioning the audio into 25-millisecond frames in 10-millisecond sliding intervals that shall be converted from the time-domain to the frequency-domain. The system then performs calculations to convert each frequency-domain frame into a MFCC feature vector, which is a representation of the distance between the MFC state and the HMM phoneme state. The system then uses the information from the MFCC feature vectors to compile a chain of phoneme states. The system then utilizes the databases to complete the final steps between the chain of phoneme states and the final sentences for the output.

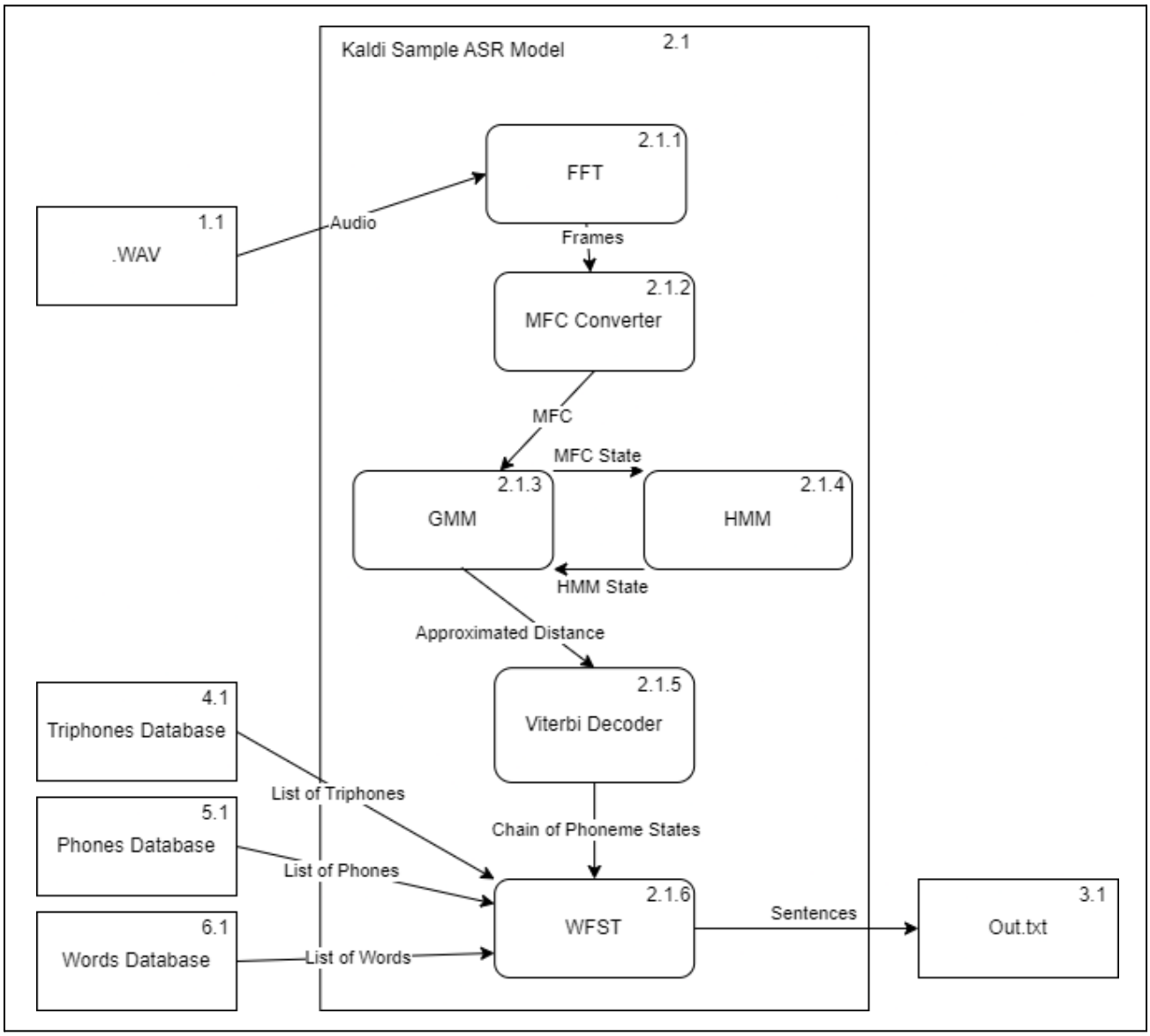

Figure 8: Kaldi ASR Toolkit Sample ASR Model Level 1 Data Flow Diagram V3.1.3

4.2.2 Stimulus/Response Sequences

	Step	System Reaction
UC1	2	Partitions audio data into 25-millisecond frames in 10-millisecond sliding intervals, allowing for three frames to overlap
	3	Converts frames from the time-domain to the frequency-domain
	4	Generates a 39-dimensional MFCC feature vector
	5	Populates the MFCC feature vector with the initial frequency-domain data, the first-order derivative of the data, and the second-order derivative of the data to show changes in the frequency-domain
	6	Compares MFCC feature vector data with HMM state
	7	Calculates the distance to the most probable phoneme based on the MFCC feature vector and the data sent by the HMM
	8	Constructs a chain of phoneme states based on the distance calculated by the GMM
	10	Builds triphones using the phoneme chains from the Viterbi Decoder
	12	Converts triphones to monophones based on context-dependency
	14	Builds words using the monophones
	15	Uses the previous two words to predict the following word
	16	Uses the language model to assemble the words into sentences

4.2.3 Functional Requirements

Req. 4.2.3.1	The system shall partition the audio data into 25-millisecond frames in 10-millisecond sliding intervals starting from time equals 0 milliseconds
Req. 4.2.3.2	The system shall convert the audio data in the frames from the time-domain to the frequency-domain.
Req. 4.2.3.3	The system shall produce a 39-dimensional MFC represented as an MFCC feature vector.
Req. 4.2.3.4	The system shall use the first set of thirteen dimensions (i.e., [0, 12]) to represent the thirteen MFCCs.
Req. 4.2.3.5	The system shall use the second set of thirteen dimensions (i.e., [13, 25]) to represent the first-order derivatives of the thirteen MFCCs.
Req. 4.2.3.6	The system shall use the third set of thirteen dimensions (i.e., [26, 38]) to represent the second-order derivatives of the thirteen MFCCs.
Req. 4.2.3.7	The system shall then compare the MFCC feature vector data with the current state of the HMM.
Req. 4.2.3.8	The system shall use the MFCC feature vector to calculate the distance from the HMM state.
Req. 4.2.3.9	The system shall construct a chain of phonemes using the distance.
Req. 4.2.3.10	The system shall find the phoneme states using the distance.
Req. 4.2.3.11	The system shall use the chain of phoneme states to construct triphones using the phone database.
Req. 4.2.3.12	The system shall convert the triphones to monophones using context-dependency.
Req. 4.2.3.13	The system shall construct words using the monophones.
Req. 4.2.3.14	The system shall use the two previous words to predict the next word.
Req. 4.2.3.15	The system shall use the language model to assemble the words into sentences.

4.3 Output a text file with the words in the audio file

4.3.1 Description and Priority

Outputting a file “out.txt” containing the words transcribed from the audio file is of high priority. The sample ASR model shall output a file “out.txt” and overwrite any previously existing instance of the file in the same directory.

4.3.2 Stimulus/Response Sequences

UC1	Step	System Reaction
	17	Transfers sentences to a text file

4.3.3 Functional Requirements

Req 4.3.3.1	The system shall write the sentence output from the Language Model into a text file called “out.txt”.
Req 4.3.3.2	The system shall store “out.txt” in the folder: /model_name/s5.
Req 4.3.3.3	The system shall transcribe words stored in the lexicon in uppercase.
Req 4.3.3.4	The system shall transcribe words not stored in the lexicon in lowercase
Req 4.3.3.5	The system shall spell all words not in the lexicon phonetically.
Req 4.3.3.6	The system shall not transcribe punctuation or grammatical markings.
Req 4.3.3.7	The system shall overwrite any previously existing instance of “out.txt” upon every transcription.
Req 4.3.3.8	The system shall append transcribed words not stored in the lexicon into a separate library.

Appendix A: Glossary

Term	Definition
ATC	***Air Traffic Control;*** the service that elicits communications between pilots and helps to prevent air traffic accidents.
ASR	***Automatic Speech Recognition;*** the ability for computers to recognize and translate spoken speech.
CLI	***Command-Line Interface;*** text-based interface that allows interaction from the user to the computer program.
DNN	***Deep Neural Network;*** a machine learning technique that represents learning and processing data in artificial neural networks.
ERAU	***Embry Riddle Aeronautical University;*** an aviation-centered university located in Daytona Beach, Florida, United States.
FFmpeg	***Linux utility;*** a portable open-source utility that allows users to decode, encode, transcode, multiplex, demultiplex, stream, filter, and play most human- or machine-made multimedia.
FFT	***Fast Fourier Transform;*** algorithm used to obtain the spectrum or . frequency content of a signal.
General American English	The most spoken variety of the English language in the United States.
GMM	***Gaussian Mixture Model;*** used to calculate the distance between the MFC feature vector and the HMM state.
HMM	***Hidden Markov Model;*** used to find the state locations of the phonemes..
IPA	***International Phonetic Alphabet:*** an alphabetic system of phonetic notation developed by the International Phonetic Association; used to represent speech sounds in a standardized format.
Lexicon	*pertaining to speech;* a library of words that is understood by the language model.
MFC	***Mel-Frequency Cepstrum;*** a representation of the short-term power spectrum of a sound
MFCCs	***Mel-Frequency Cepstral Coefficients;*** the coefficients that a MFC is comprised of
NLP	***Natural Language Processing;*** the culmination of computer science, linguistics, and machine learning.
Phone	*pertaining to speech;* a distinct speech sound or gesture;
Phoneme	*pertaining to speech;* a set of phones that can distinguish one word from another
Triphone	*pertaining to speech;* a sequence of three consecutive phonemes
WER	***Word Error Rate;*** the rate at which error in words occurs

Appendix B: Analysis Models

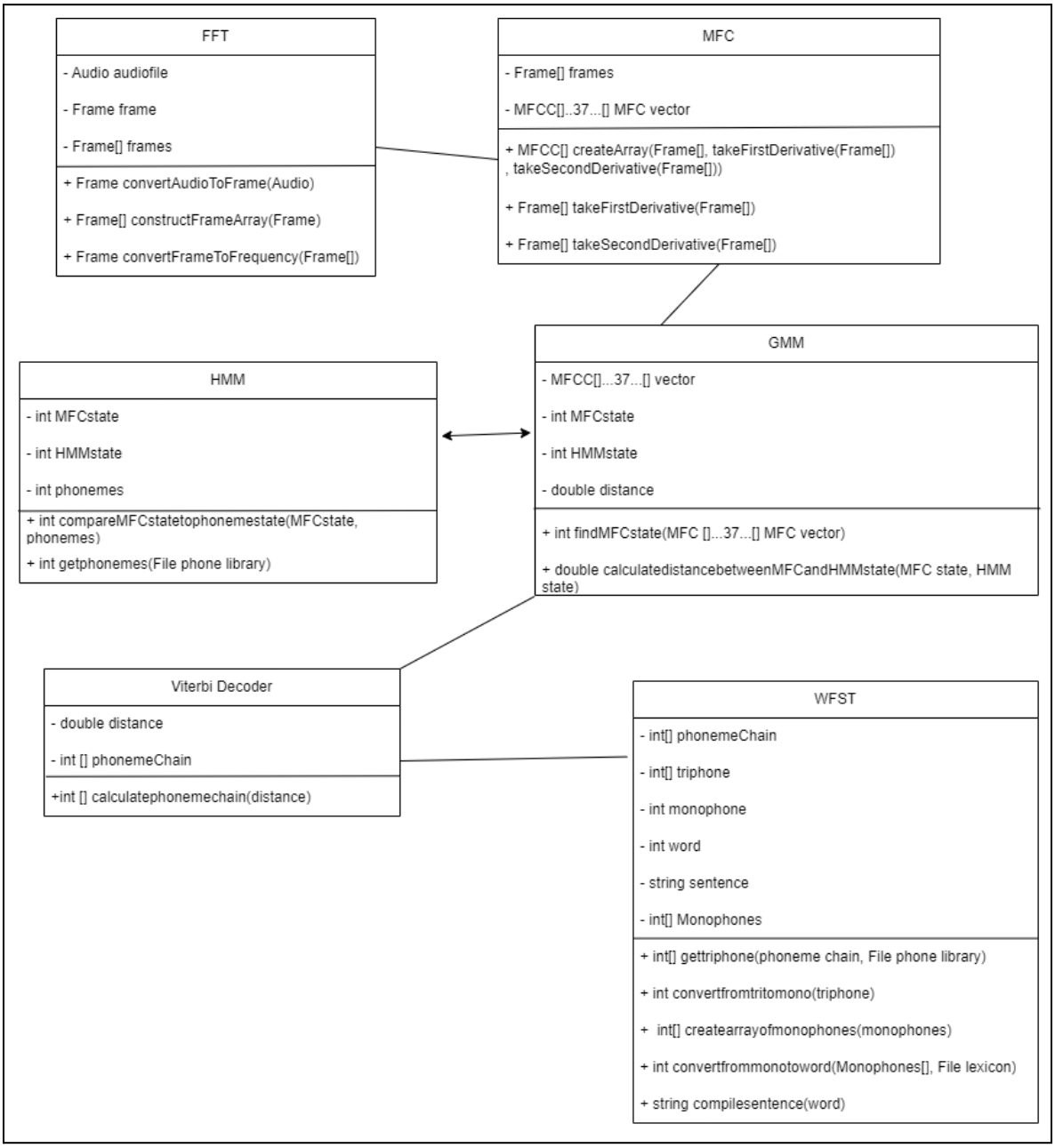

Figure 9: Kaldi Sample ASR Class Diagram V3.3

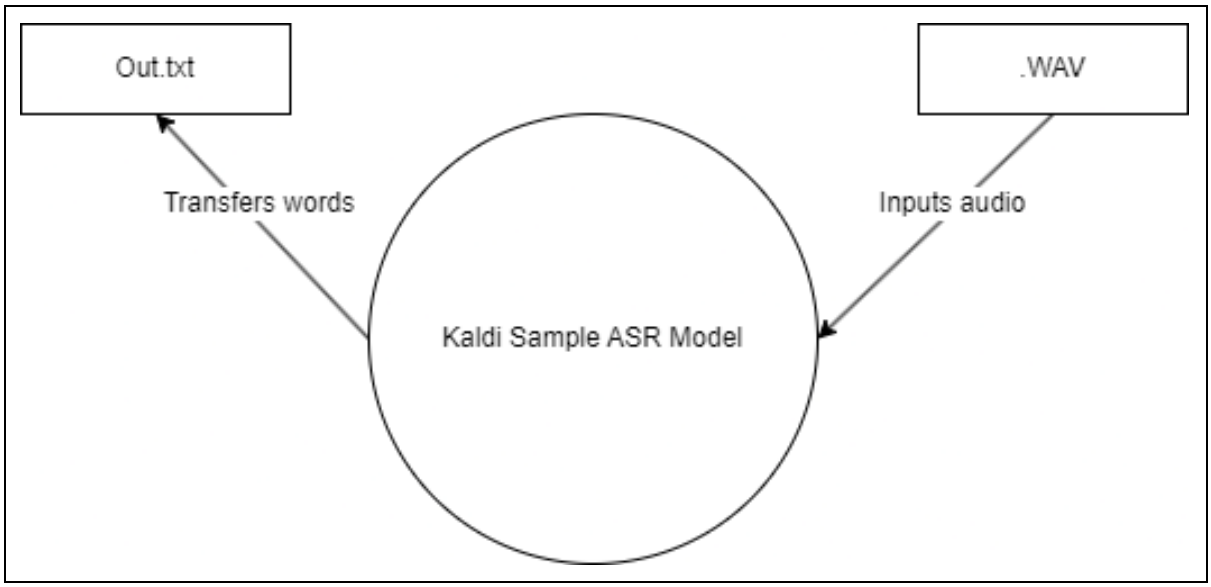

Figure 10: Kaldi Sample ASR Context Diagram V3

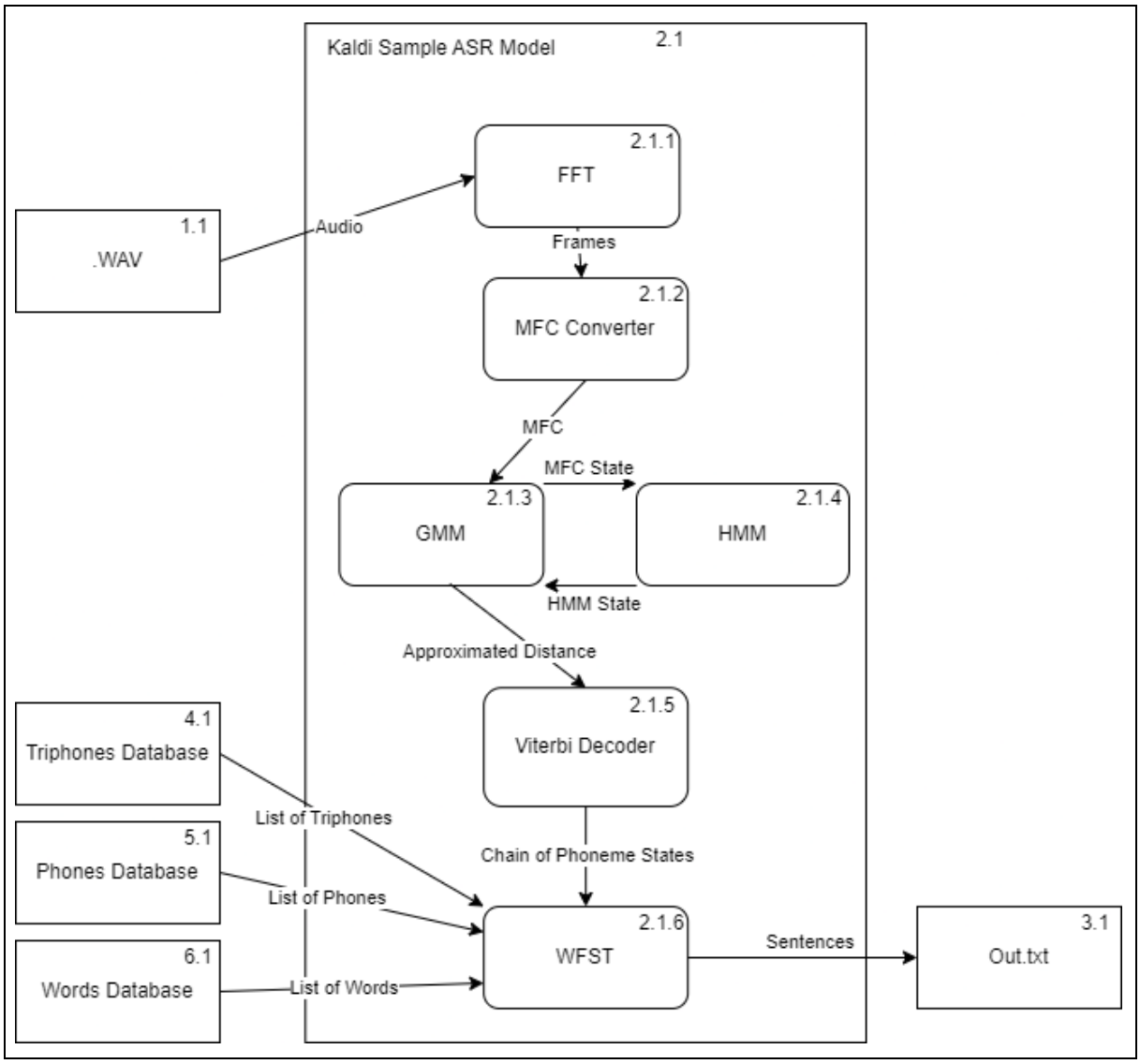

Figure 11: Kaldi ASR Toolkit Sample ASR Model Level 1 Data Flow Diagram V3.1.3

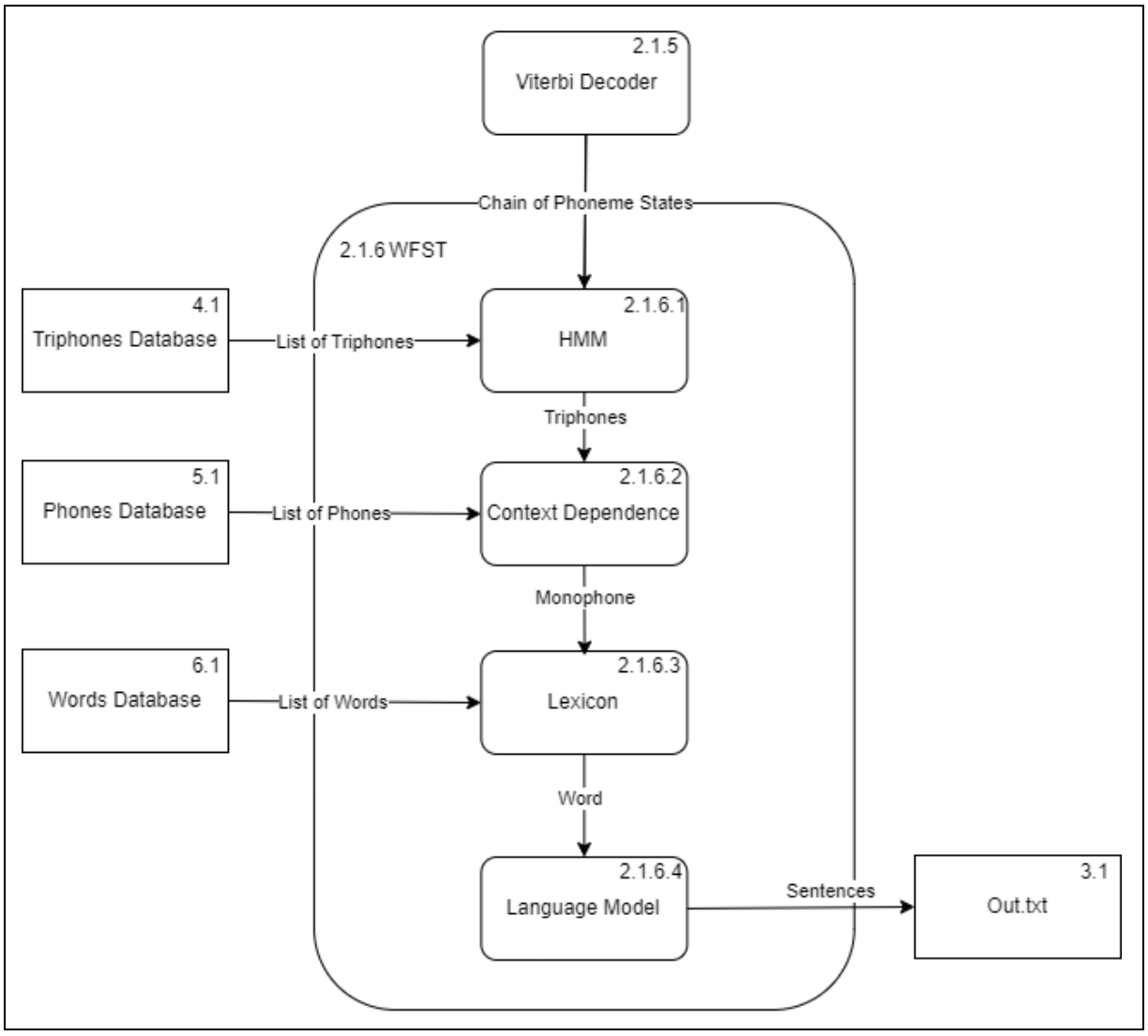

Figure 12: Kaldi ASR Toolkit Sample ASR Model Level 2 Data Flow Diagram V3.2.2

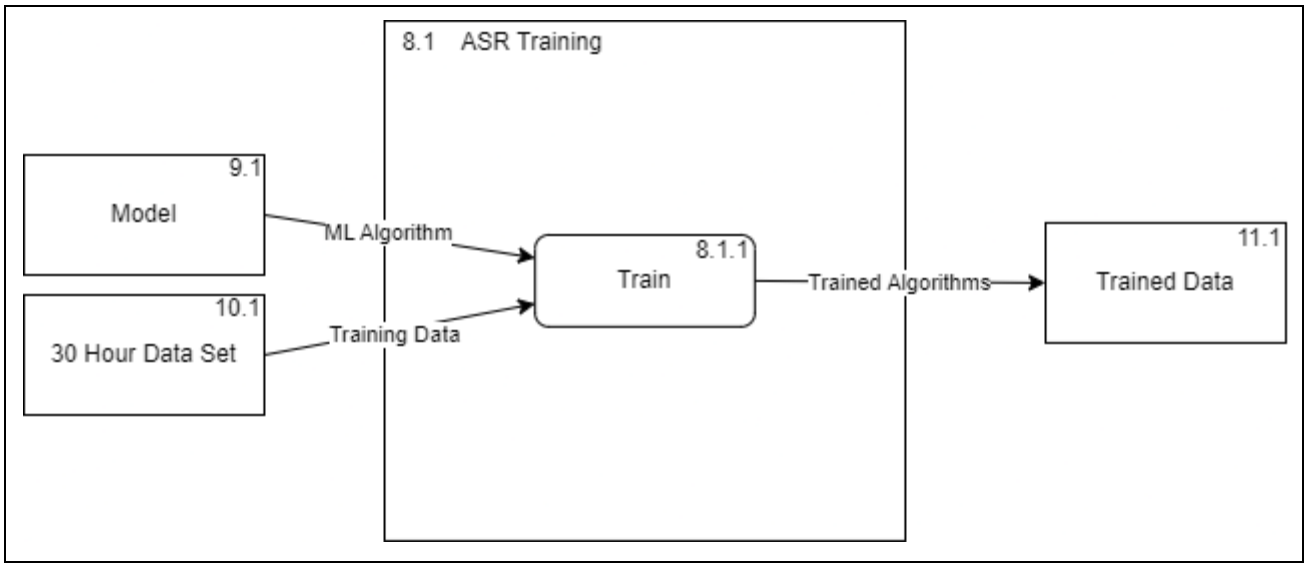

Figure 13: Kaldi ASR Training Model Level 1 Data Flow Diagram V3.3.1

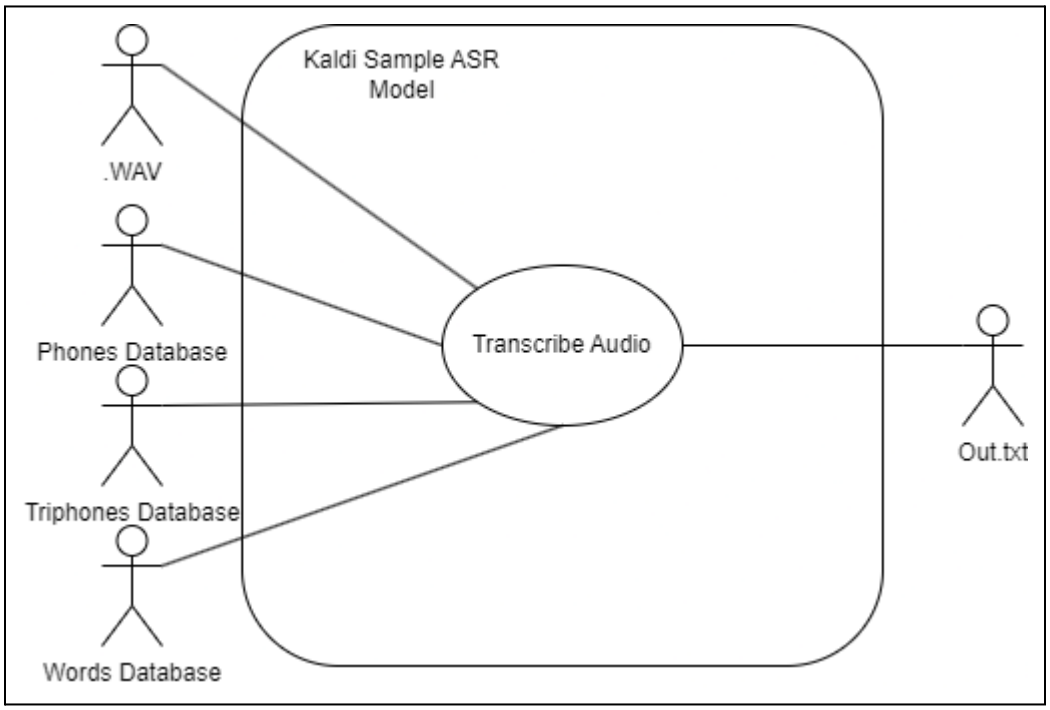

Figure 14: Kaldi ASR Toolkit Sample ASR Model Use Case Diagram V2.1.1

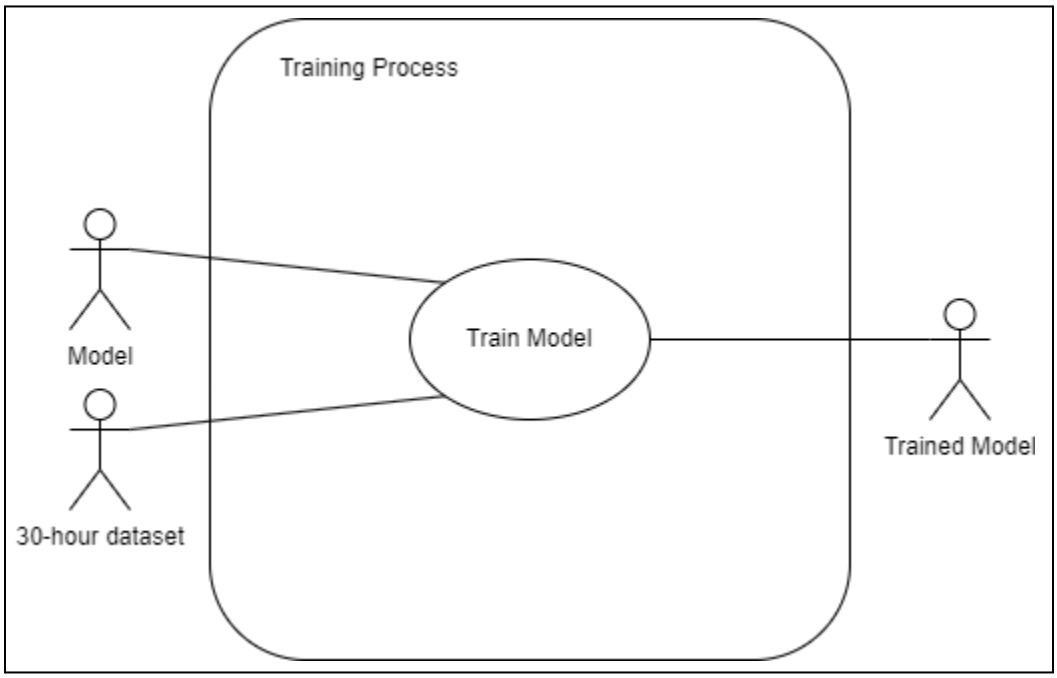

Figure 15: Kaldi ASR Toolkit Sample ASR Model Use Case Diagram V2.2.1