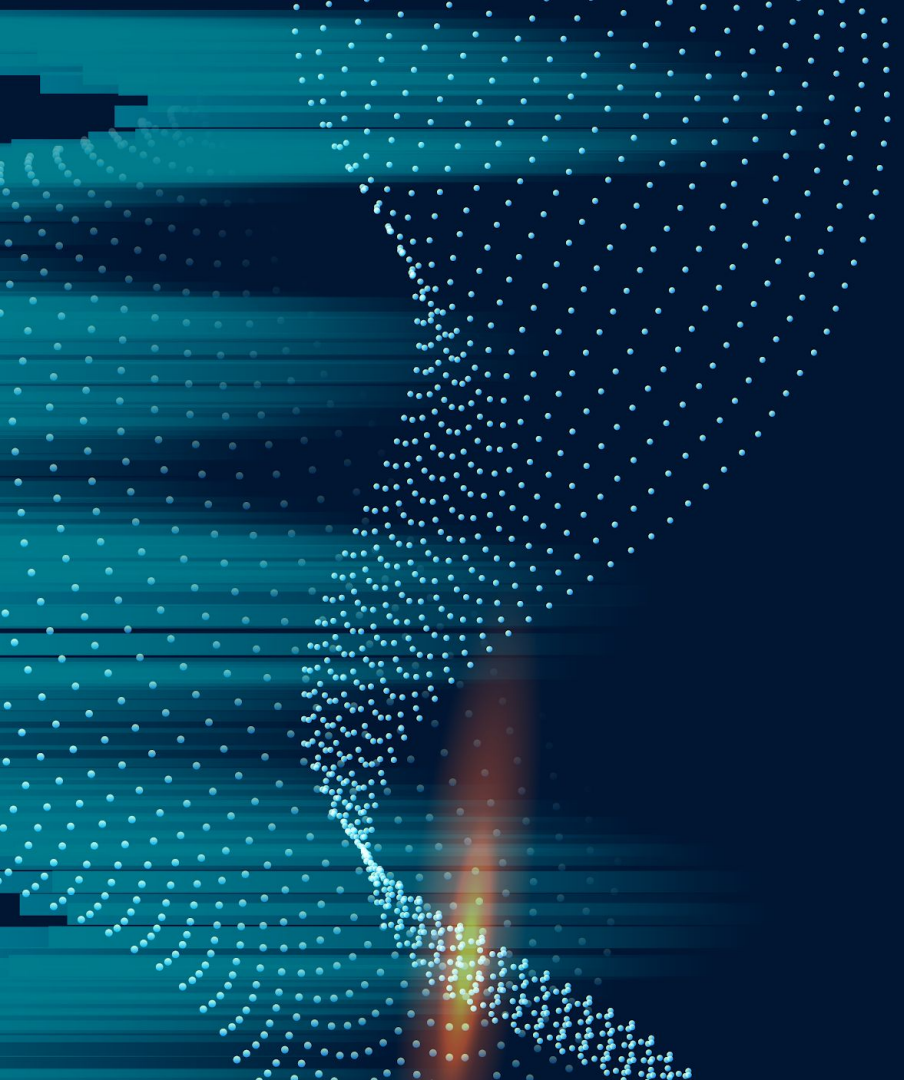
Kaldi ASR Modeling:

Robust Automatic Speech Recognition using Kaldi for Radiotelephonic Applications



Dr. Jianhua Liu & Andrew Schneider

Product Owners

Tabitha O'Malley

Team Lead, Research, Documentation

Milan Haruyama

Coding, Research, Documentation

David Serfaty

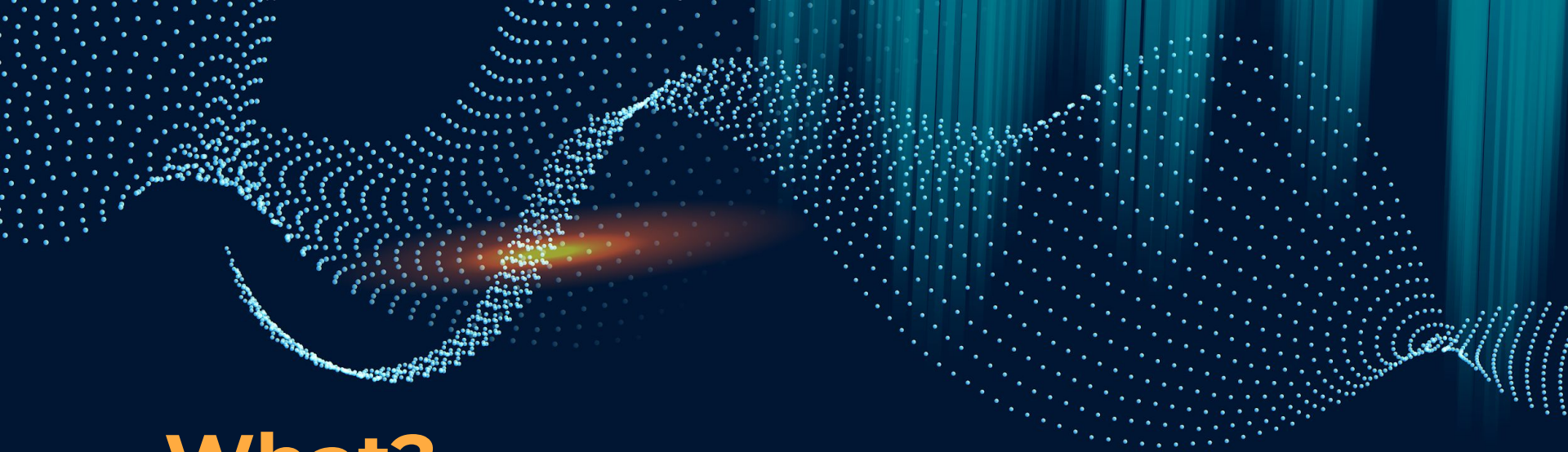
Coding, Research, Documentation

Tamina Tisha

Research, Documentation

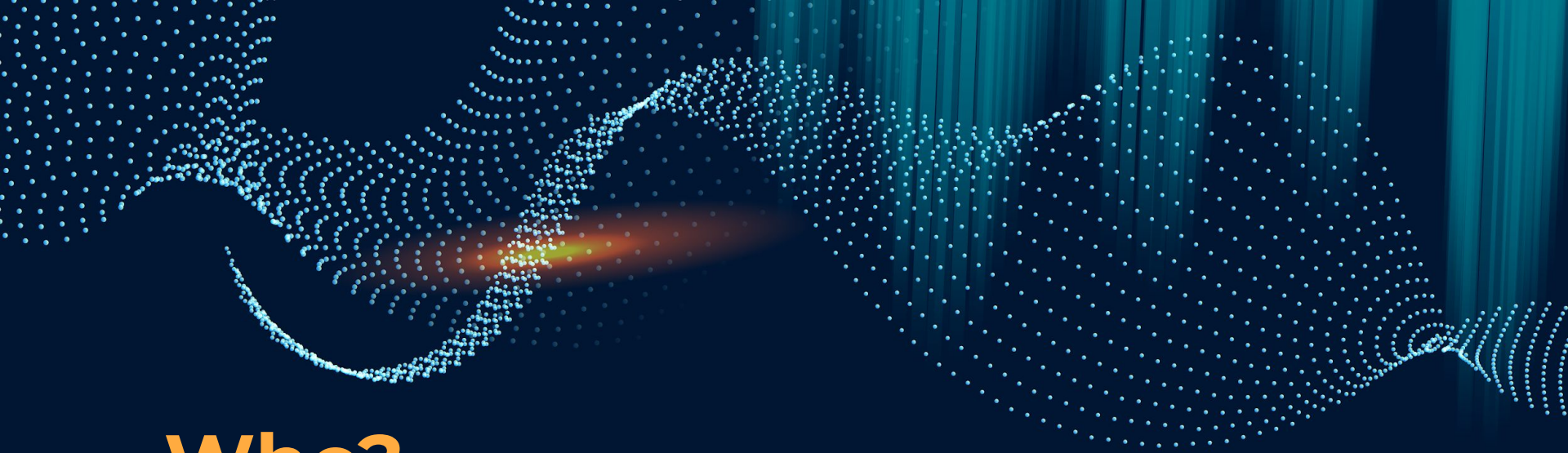
Adam Gallub

Coding, Documentation



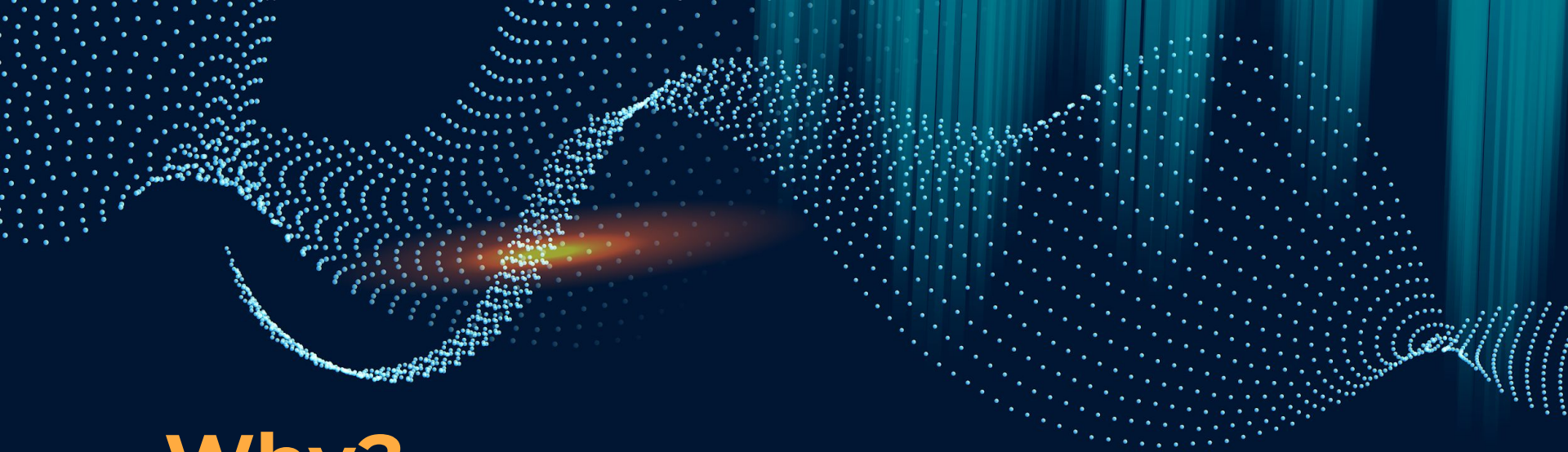
What?

Develop acoustic ASR models capable of transcribing live ATC transmissions in real-time using the Kaldi ASR Toolkit and the ATC02 corpus.



Who?

- Ab initio pilots
- Flight Instructors
- Air Traffic Control Operators
- Hobbyists



Why?

- Miscommunications between ATC and pilots due to a lack of formal training for pilots
- Ability to review communications on the ground after flights for training purposes

End to End vs Acoustic Models

End-to-End (E2E) Models

- Relationship between audio signals and the words
- Large dataset is required
- Larger and more complex models

Acoustic Models

- Relationships between audio signals and the phonetics
- Better results with small dataset
- Smaller models

Design Constraints (Training)

- Minimum storage size of 12.5 GB
- Minimum video memory (VRAM) of 12 GB for model training
  - NVidia GeForce RTX 4080
- AMD processor (recommended)

Design Constraints (Transcribing)

- All inputs as WAV files
  - Convert non-WAV files using FFMPEG utility
- General American English
  - E.g., “color” versus “colour”
- No punctuation or grammatical marks
  - Commas, colons, hyphens, etc.

Assumptions and Dependencies

- Clear and direct communication
  - Low interference and background noise
- General American English
- 20- to 100-hour dataset
- Sufficient storage
- Sufficient video memory
- Operating system
  - Linux (e.g., Ubuntu, Debian, etc.)
  - Windows Subsystem for Linux (WSL)
  - Linux virtual machine for MacOS

System Architecture (Preprocessing)

ATC0 Corpus

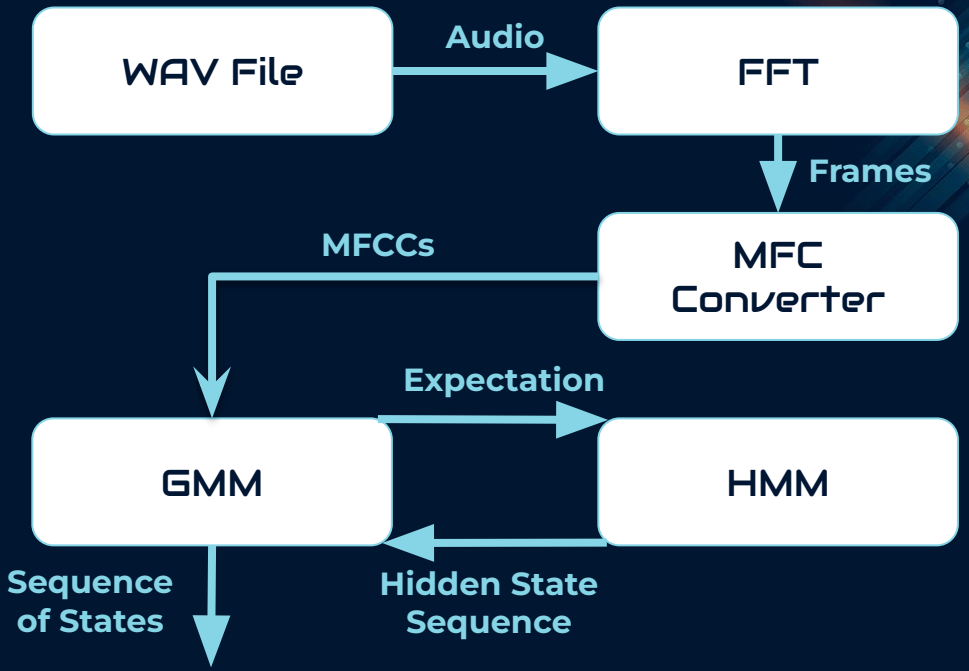
- Linguistic Data Consortium (LDC)
- 30-hour ATC dataset
  - Audio files
  - Text transcriptions

ATCO2 Corpus

- Repository for ASR and NLP research
- Provides preprocessing script written in Bash designed for Kaldi ASR Toolkit

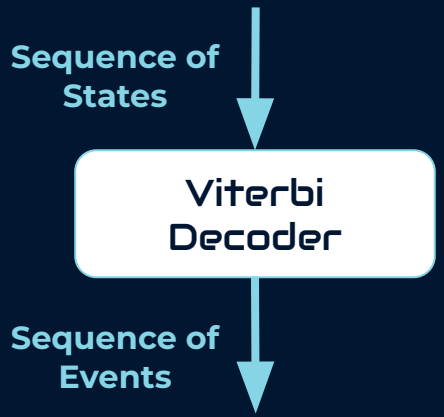
System Architecture (Preparation)

- Setting up audio for decoding
- Fast Fourier Transform (FFT)
  - Time-domain → Frequency domain
- Mel-Frequency Cepstrum (MFC)
  - Mel-Frequency Cepstral Coefficients (MFCCs)
- Gaussian Mixture Model (GMM)
- Hidden Markov Model (HMM)



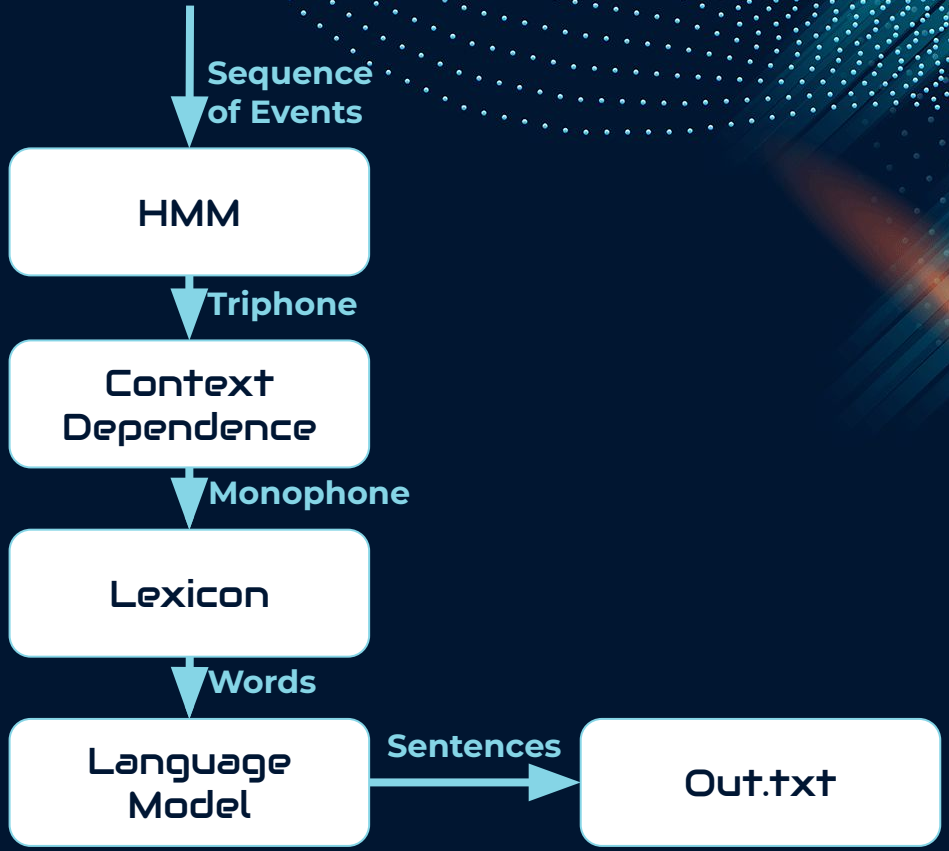
System Architecture (Preparation)

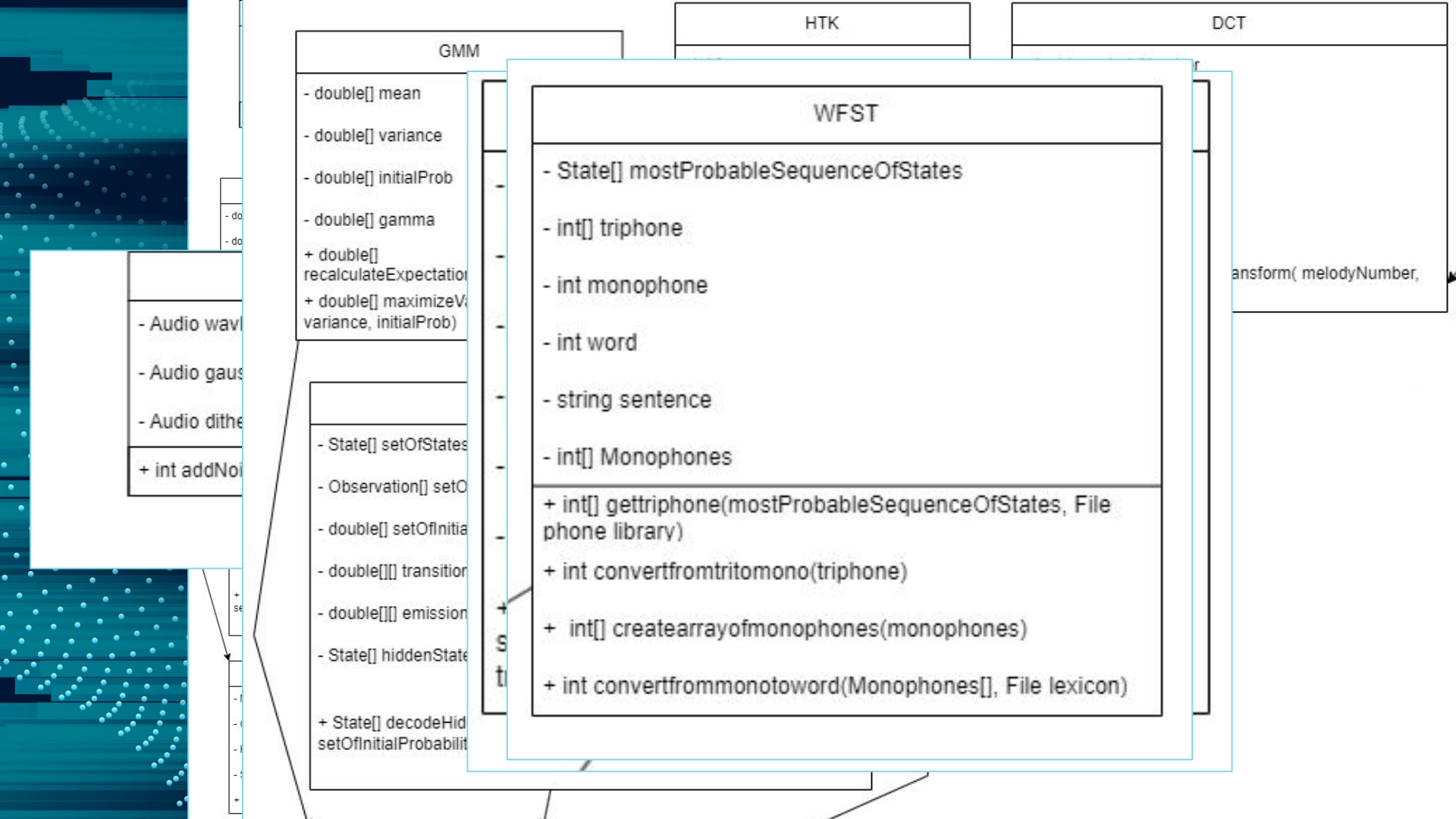
- Using hidden state sequence to find the most probable sequence of phonemes
- Viterbi decoder

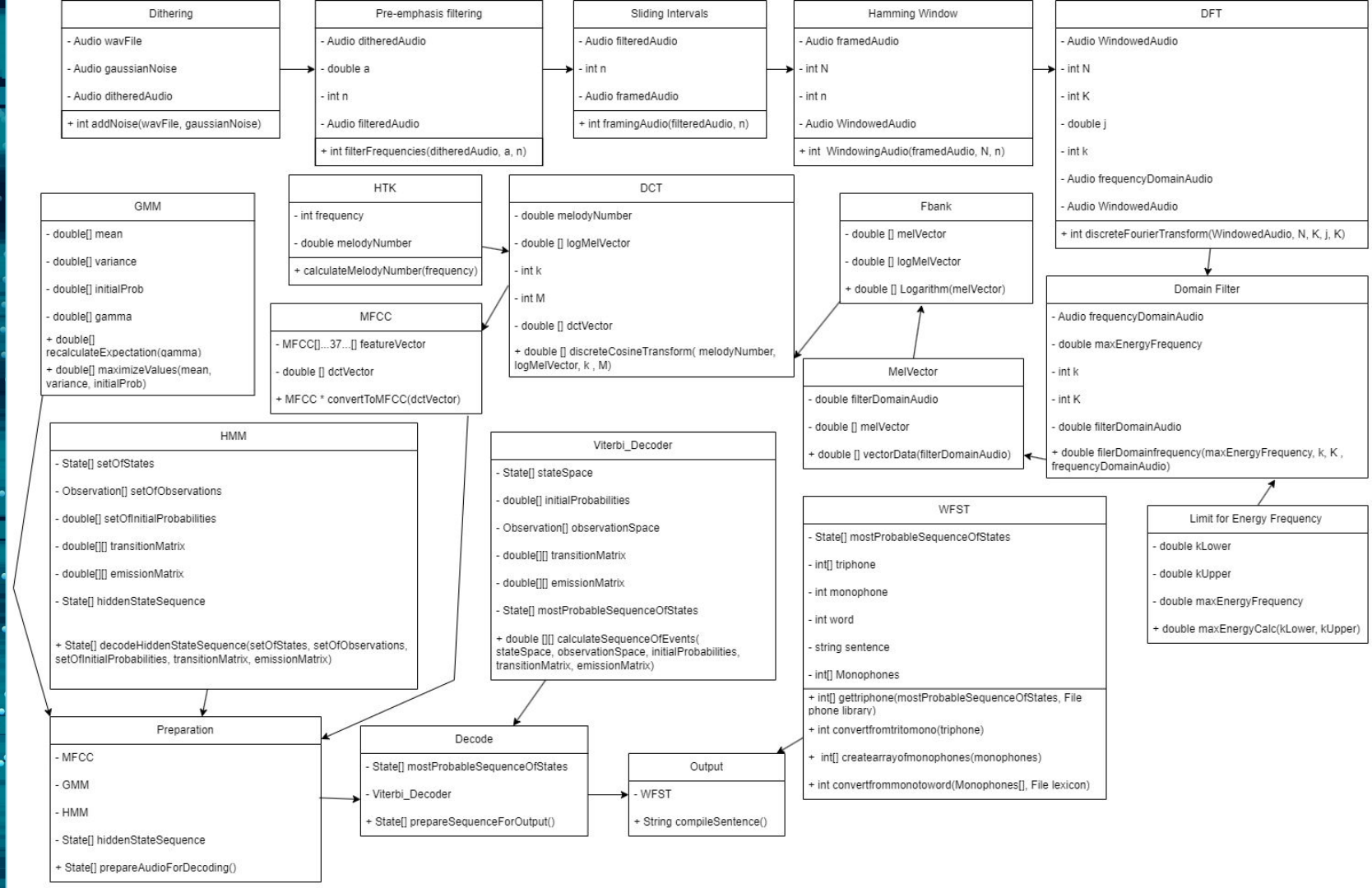


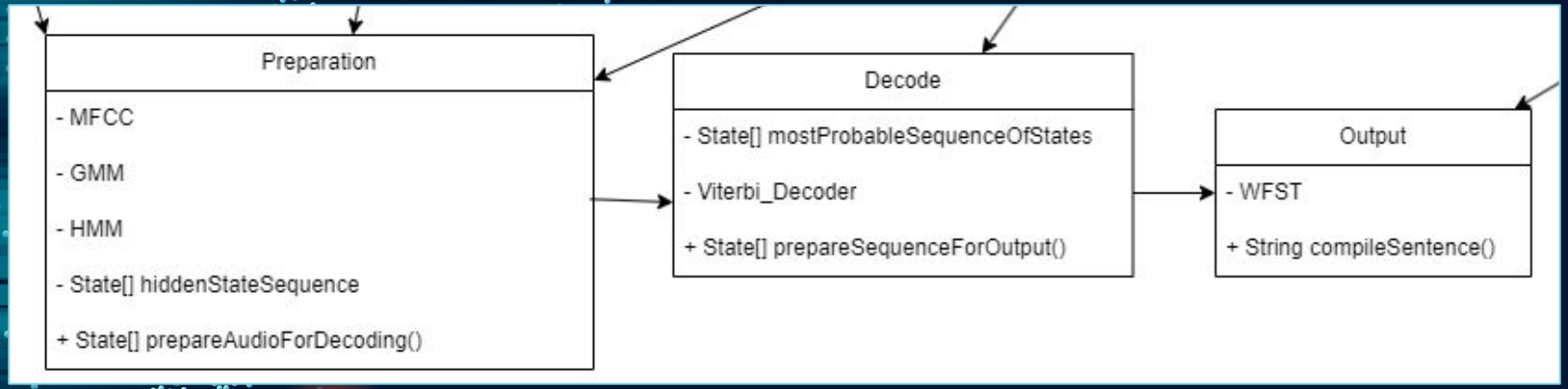
System Architecture (Output)

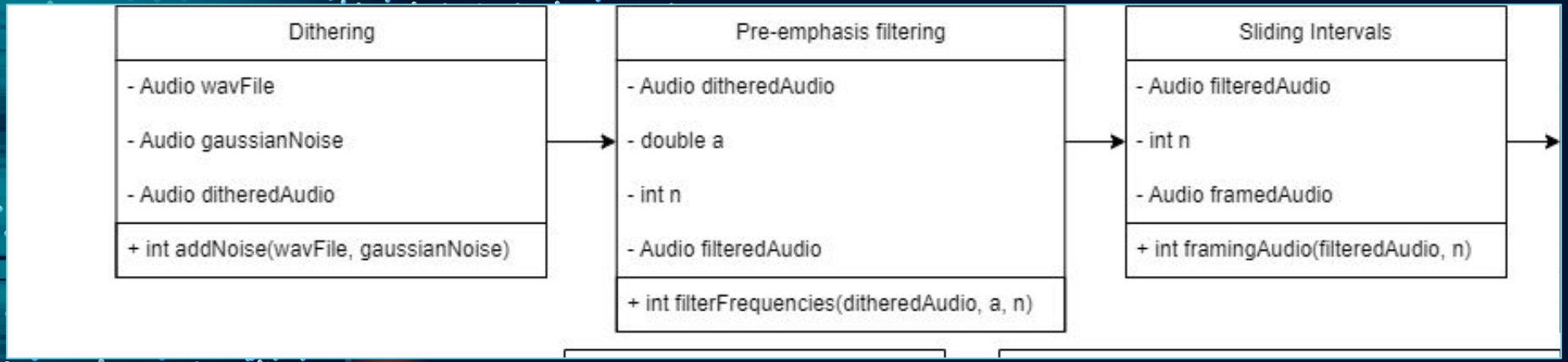
- Converting to sentences
- Weighted Finite State Transducers (WFSTs)
- HMM
- Context Dependence
- Lexicon
- Language Model

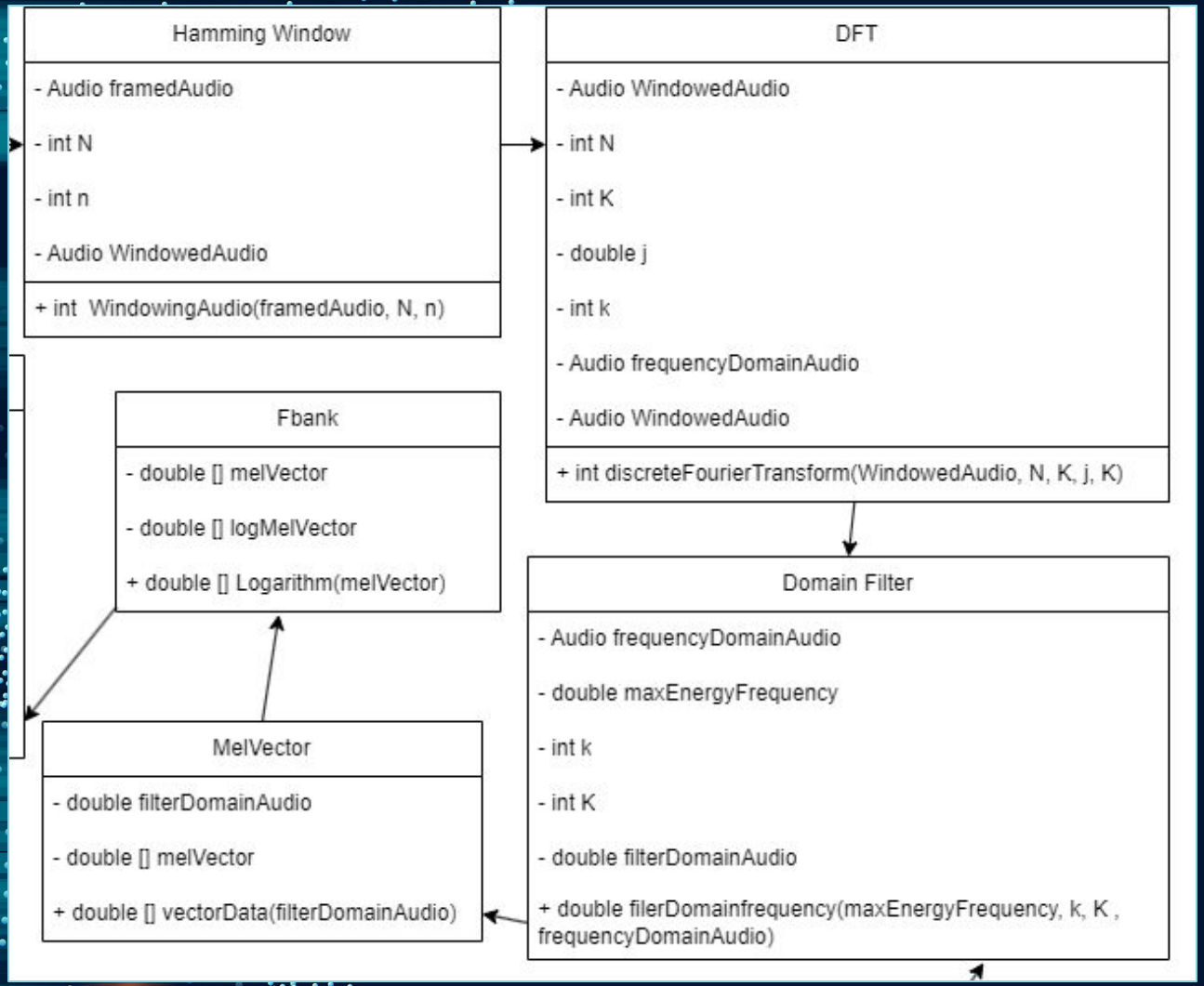


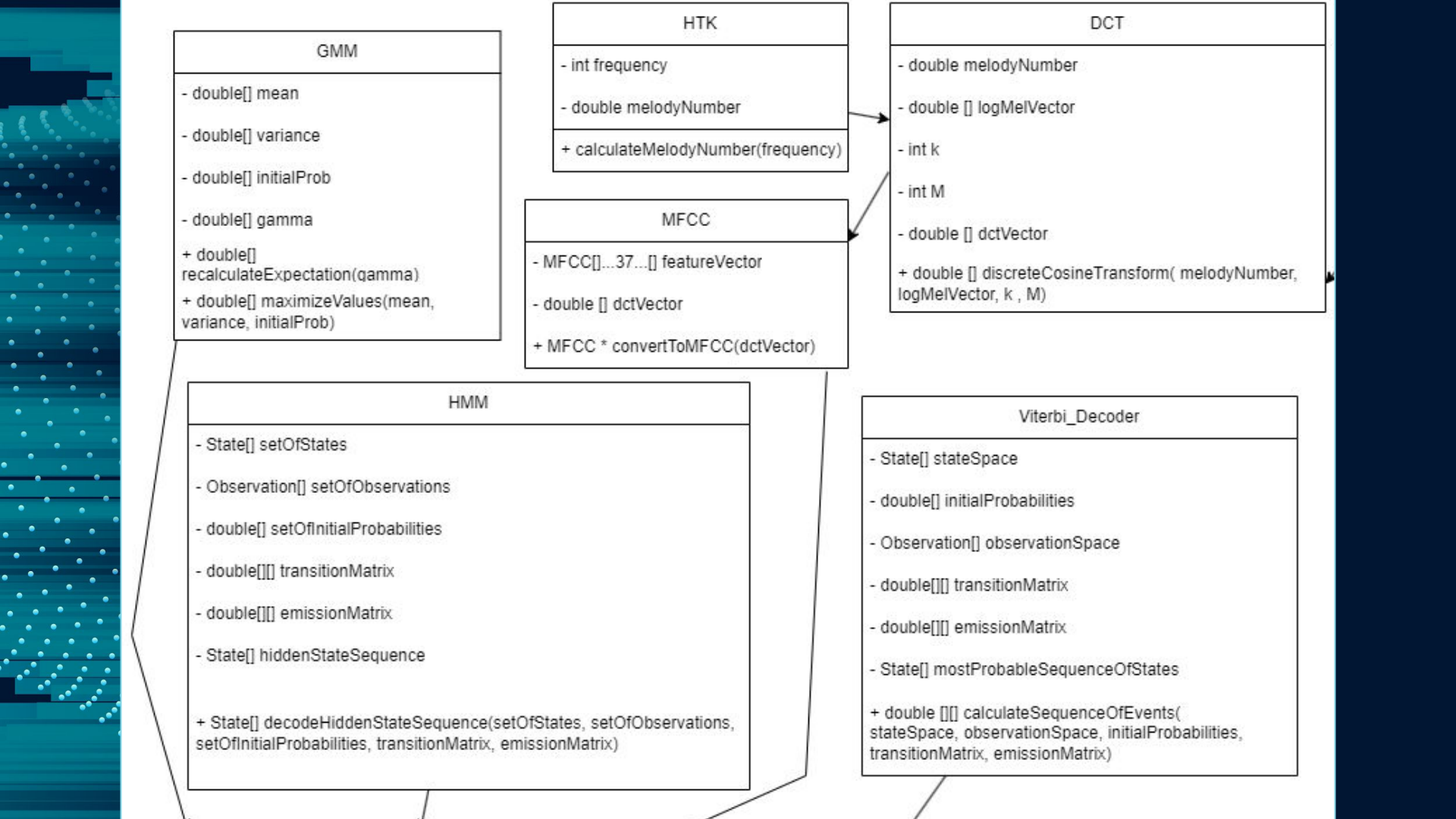

HTK
DCT
GMM
- double[] mean
- double[] variance
- double[] initialProb
- double[] gamma
+ double[] recalculateExpectatio
+ double[] maximizeV
variance, initialProb)
ansform(melodyNumber,
- Audio wav
- Audio gaus
- Audio dithe
+ int addNoi
- State[] setOfStates
- Observation[] setO
- double[] setOfInitia
- double[][] transitio
- double[][] emission
- State[] hiddenState
+ State[] decodeHid
setOfInitialProbabilit
WFST
- State[] mostProbableSequenceOfStates
- int[] triphone
- int monophone
- int word
- string sentence
- int[] Monophones
+ int[] gettriphone(mostProbableSequenceOfStates, File phone library)
+ int convertfromtritomono(triphone)
+ int[] createarrayofmonophones(monophones)
+ int convertfrommonotoword(Monophones[], File lexicon)

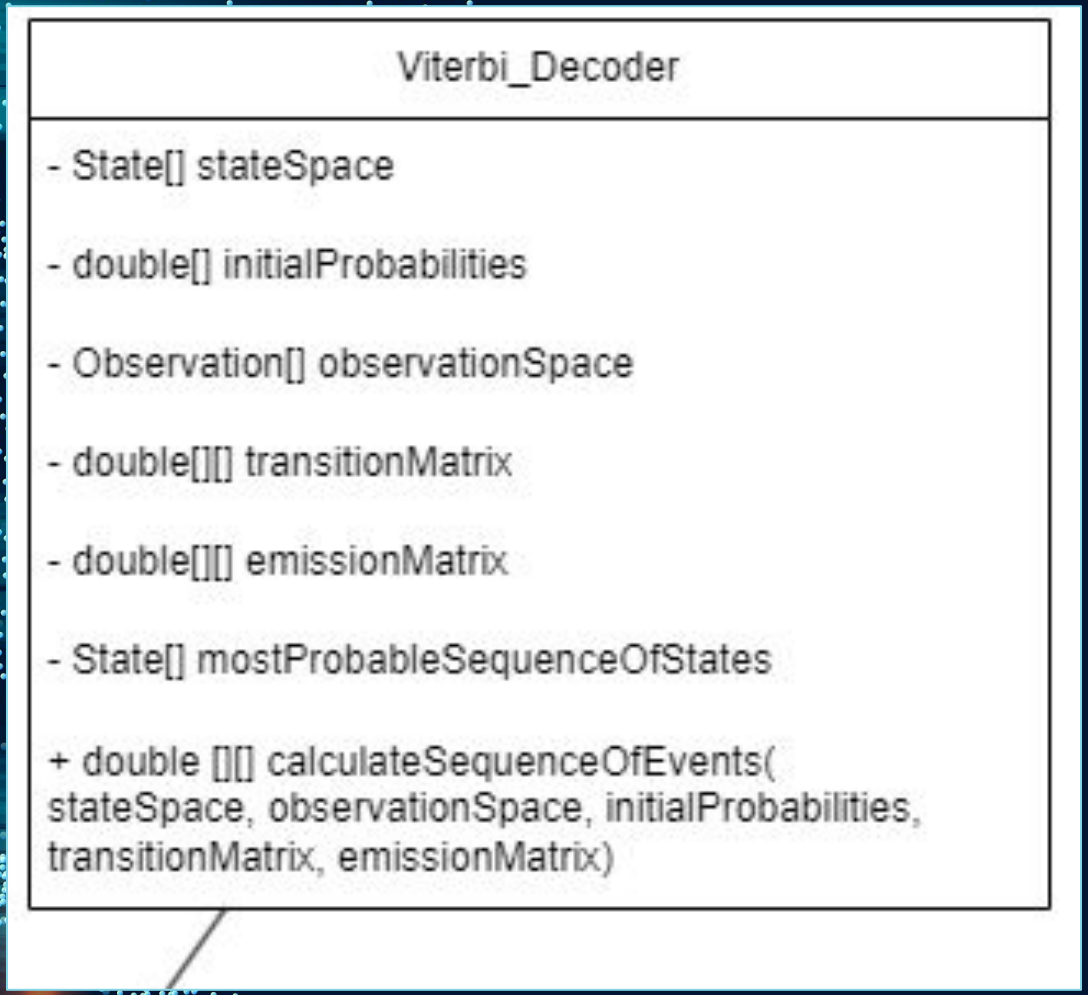

Dithering
- Audio wavFile
- Audio gaussianNoise
- Audio ditheredAudio
+ int addNoise(wavFile, gaussianNoise)
Pre-emphasis filtering
- Audio ditheredAudio
- double a
- int n
- Audio filteredAudio
+ int filterFrequencies(ditheredAudio, a, n)
Sliding Intervals
- Audio filteredAudio
- int n
- Audio framedAudio
+ int framingAudio(filteredAudio, n)
Hamming Window
- Audio framedAudio
- int N
- int n
- Audio WindowedAudio
+ int WindowingAudio(framedAudio, N, n)
DFT
- Audio WindowedAudio
- int N
- int K
- double j
- int k
- Audio frequencyDomainAudio
- Audio WindowedAudio
+ int discreteFourierTransform(WindowedAudio, N, K, j, K)
GMM
- double[] mean
- double[] variance
- double[] initialProb
- double[] gamma
+ double[] recalculateExpectation(gamma)
+ double[] maximizeValues(mean, variance, initialProb)
HTK
- int frequency
- double melodyNumber
+ calculateMelodyNumber(frequency)
DCT
- double melodyNumber
- double [] logMelVector
- int k
- int M
- double [] dctVector
+ double [] discreteCosineTransform(melodyNumber, logMelVector, k , M)
Fbank
- double [] melVector
- double [] logMelVector
+ double [] Logarithm(melVector)
Domain Filter
- Audio frequencyDomainAudio
- double maxEnergyFrequency
- int k
- int K
- double filterDomainAudio
+ double filerDomainfrequency(maxEnergyFrequency, k, K , frequencyDomainAudio)
MFCC
- MFCC[]...37...[] featureVector
- double [] dctVector
+ MFCC * convertToMFCC(dctVector)
MelVector
- double filterDomainAudio
- double [] melVector
+ double [] vectorData(filterDomainAudio)
HMM
- State[] setOfStates
- Observation[] setOfObservations
- double[] setOfInitialProbabilities
- double[][] transitionMatrix
- double[][] emissionMatrix
- State[] hiddenStateSequence
+ State[] decodeHiddenStateSequence(setOfStates, setOfObservations, setOfInitialProbabilities, transitionMatrix, emissionMatrix)
Viterbi_Decoder
- State[] stateSpace
- double[] initialProbabilities
- Observation[] observationSpace
- double[][] transitionMatrix
- double[][] emissionMatrix
- State[] mostProbableSequenceOfStates
+ double [][] calculateSequenceOfEvents(stateSpace, observationSpace, initialProbabilities, transitionMatrix, emissionMatrix)
WFST
- State[] mostProbableSequenceOfStates
- int[] triphone
- int monophone
- int word
- string sentence
- int[] Monophones
+ int[] gettriphone(mostProbableSequenceOfStates, File phone library)
+ int convertfromtritomono(triphone)
+ int[] createarrayofmonophones(monophones)
+ int convertfrommonotoword(Monophones[], File lexicon)
Limit for Energy Frequency
- double kLower
- double kUpper
- double maxEnergyFrequency
+ double maxEnergyCalc(kLower, kUpper)
Preparation
- MFCC
- GMM
- HMM
- State[] hiddenStateSequence
+ State[] prepareAudioForDecoding()
Decode
- State[] mostProbableSequenceOfStates
- Viterbi_Decoder
+ State[] prepareSequenceForOutput()
Output
- WFST
+ String compileSentence()

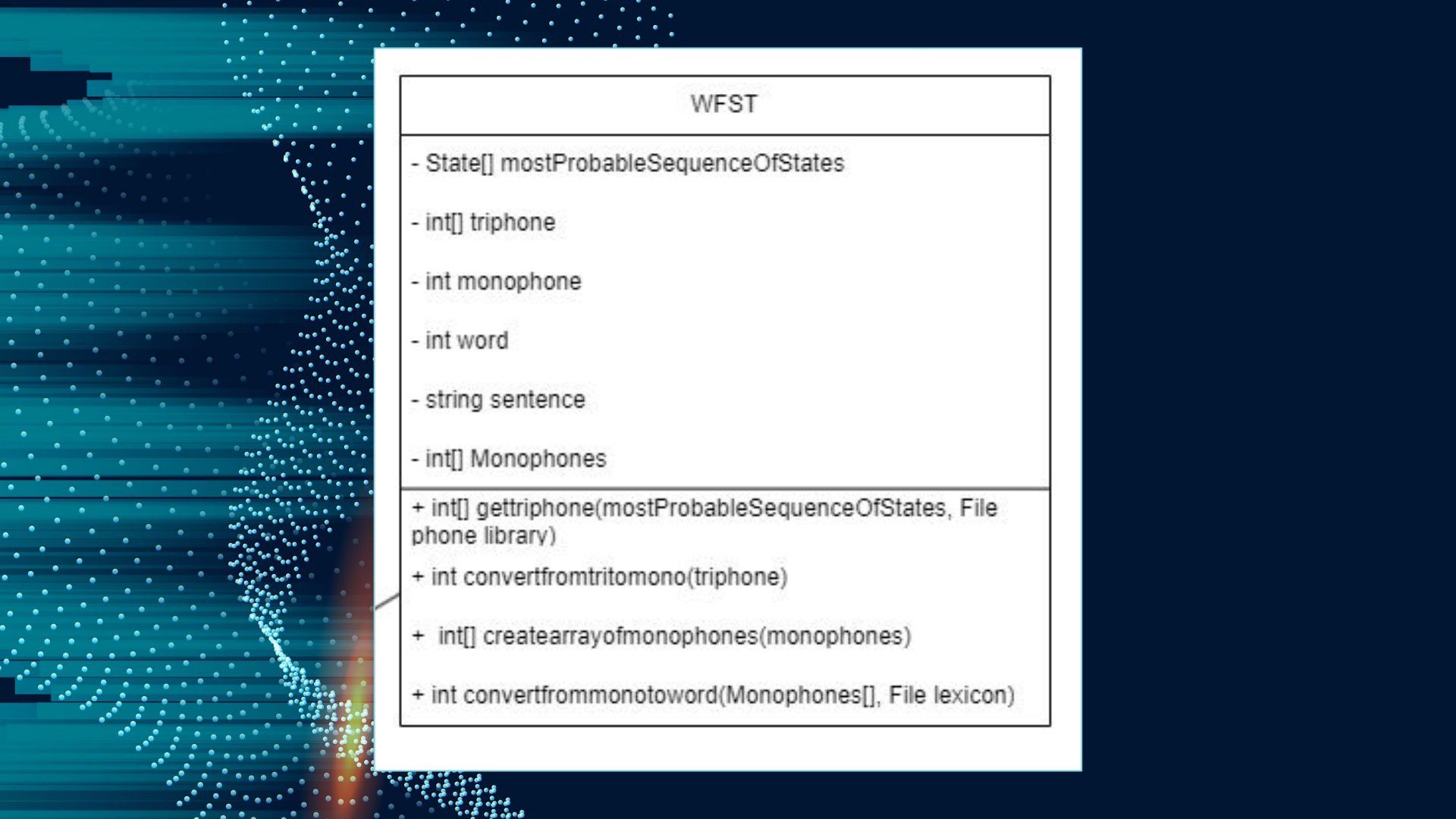

Preparation
- MFCC
- GMM
- HMM
- State[] hiddenStateSequence
+ State[] prepareAudioForDecoding()
Decode
- State[] mostProbableSequenceOfStates
- Viterbi_Decoder
+ State[] prepareSequenceForOutput()
Output
- WFST
+ String compileSentence()


Dithering
- Audio wavFile
- Audio gaussianNoise
- Audio ditheredAudio
+ int addNoise(wavFile, gaussianNoise)
Pre-emphasis filtering
- Audio ditheredAudio
- double a
- int n
- Audio filteredAudio
+ int filterFrequencies(ditheredAudio, a, n)
Sliding Intervals
- Audio filteredAudio
- int n
- Audio framedAudio
+ int framingAudio(filteredAudio, n)

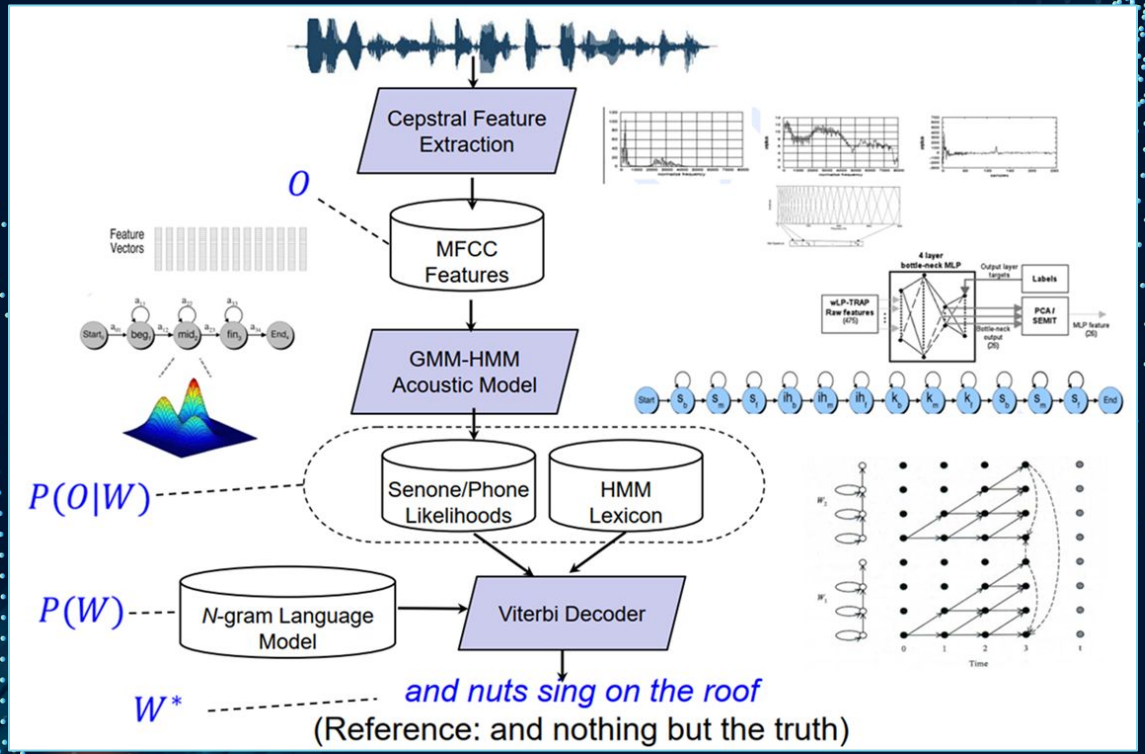

Hamming Window
- Audio framedAudio
- int N
- int n
- Audio WindowedAudio
+ int WindowingAudio(framedAudio, N, n)
DFT
- Audio WindowedAudio
- int N
- int K
- double j
- int k
- Audio frequencyDomainAudio
- Audio WindowedAudio
+ int discreteFourierTransform(WindowedAudio, N, K, j, K)
Fbank
- double [] melVector
- double [] logMelVector
+ double [] Logarithm(melVector)
Domain Filter
- Audio frequencyDomainAudio
- double maxEnergyFrequency
- int k
- int K
- double filterDomainAudio
+ double filerDomainfrequency(maxEnergyFrequency, k, K , frequencyDomainAudio)
MelVector
- double filterDomainAudio
- double [] melVector
+ double [] vectorData(filterDomainAudio)


GMM
- double[] mean
- double[] variance
- double[] initialProb
- double[] gamma
+ double[] recalculateExpectation(gamma)
+ double[] maximizeValues(mean, variance, initialProb)
HTK
- int frequency
- double melodyNumber
+ calculateMelodyNumber(frequency)
DCT
- double melodyNumber
- double [] logMelVector
- int k
- int M
- double [] dctVector
+ double [] discreteCosineTransform(melodyNumber, logMelVector, k , M)
MFCC
- MFCC[]...37...[] featureVector
- double [] dctVector
+ MFCC * convertToMFCC(dctVector)
HMM
- State[] setOfStates
- Observation[] setOfObservations
- double[] setOfInitialProbabilities
- double[][] transitionMatrix
- double[][] emissionMatrix
- State[] hiddenStateSequence
+ State[] decodeHiddenStateSequence(setOfStates, setOfObservations, setOfInitialProbabilities, transitionMatrix, emissionMatrix)
Viterbi_Decoder
- State[] stateSpace
- double[] initialProbabilities
- Observation[] observationSpace
- double[][] transitionMatrix
- double[][] emissionMatrix
- State[] mostProbableSequenceOfStates
+ double [][] calculateSequenceOfEvents(stateSpace, observationSpace, initialProbabilities, transitionMatrix, emissionMatrix)


Viterbi_Decoder
- State[] stateSpace
- double[] initialProbabilities
- Observation[] observationSpace
- double[][] transitionMatrix
- double[][] emissionMatrix
- State[] mostProbableSequenceOfStates
+ double [][] calculateSequenceOfEvents(
stateSpace, observationSpace, initialProbabilities,
transitionMatrix, emissionMatrix)

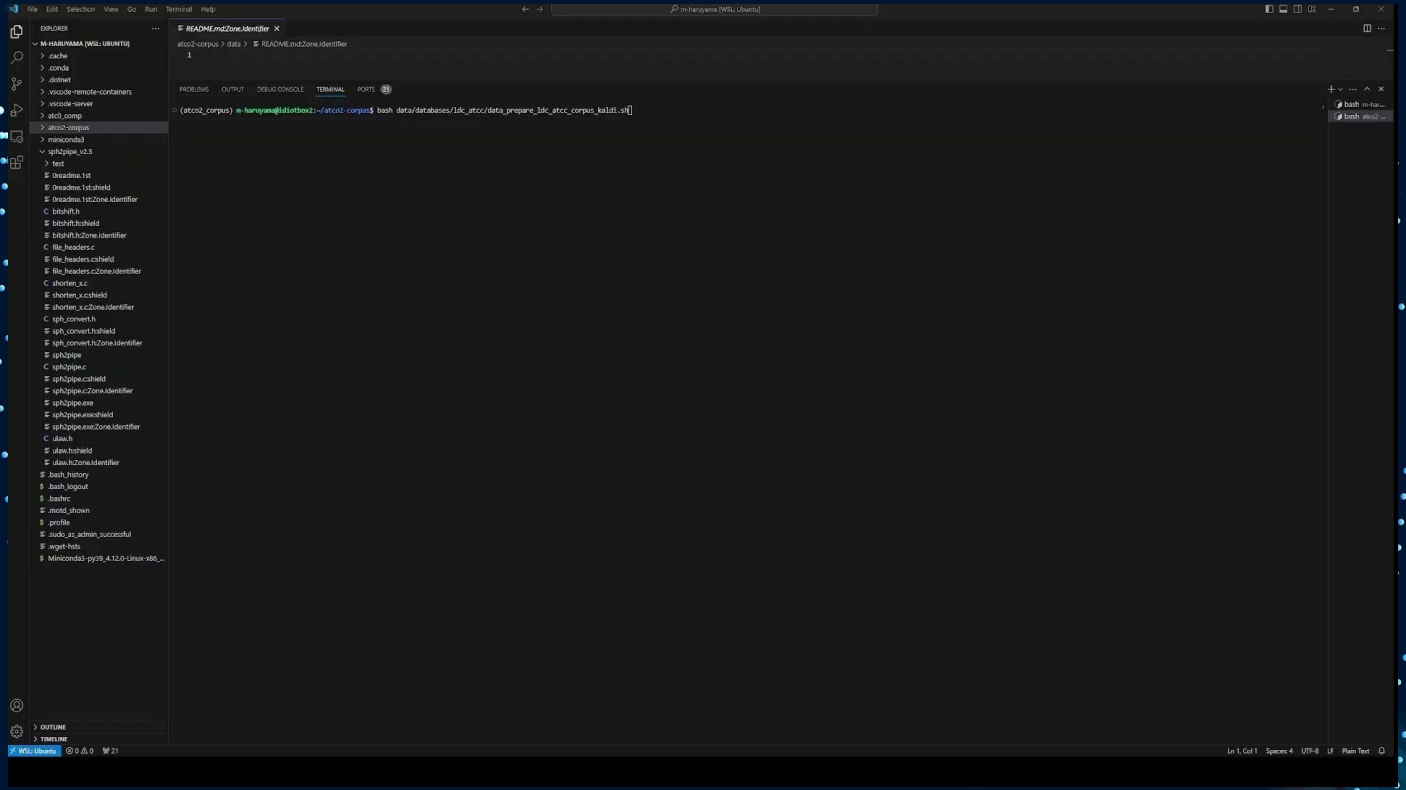

WFST
- State[] mostProbableSequenceOfStates
- int[] triphone
- int monophone
- int word
- string sentence
- int[] Monophones
+ int[] gettriphone(mostProbableSequenceOfStates, File phone library)
+ int convertfromtritomono(triphone)
+ int[] createarrayofmonophones(monophones)
+ int convertfrommonotoword(Monophones[], File lexicon)

Sub-System Design



Iteration 7.0.0	Iteration 13.1.0	Iteration 17.0.5	Iteration 17.1.0
WER = 18.06%	WER = 13.50%	WER = 13.21%	WER = 13.23%
SER = 78.88%	SER = 72.36%	SER = 71.26%	SER = 72.08%

Demo


File Edit Selection View Go Run Terminal Help
m-haruyama [WSL: Ubuntu]
EXPLORER
README.md:Zone.Identifier
atco2-corpus > data > README.md:Zone.Identifier
1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 21
(atco2_corpus) m-haruyama@idiotbox2:~/atco2-corpus$ bash data/databases/ldc_atcc/data_prepare_ldc_atcc_corpus_kaldi.sh
bash m-har...
bash atco2-...
M-HARUYAMA [WSL: UBUNTU]
.cache
.conda
.dotnet
.vscode-remote-containers
.vscode-server
atc0_comp
atco2-corpus
miniconda3
sph2pipe_v2.5
test
0readme.1st
0readme.1st:shield
0readme.1st:Zone.Identifier
bitshift.h
bitshift.h:shield
bitshift.h:Zone.Identifier
file_headers.c
file_headers.c:shield
file_headers.c:Zone.Identifier
shorten_x.c
shorten_x.c:shield
shorten_x.c:Zone.Identifier
sph_convert.h
sph_convert.h:shield
sph_convert.h:Zone.Identifier
sph2pipe
sph2pipe.c
sph2pipe.c:shield
sph2pipe.c:Zone.Identifier
sph2pipe.exe
sph2pipe.exe:shield
sph2pipe.exe:Zone.Identifier
ulaw.h
ulaw.h:shield
ulaw.h:Zone.Identifier
.bash_history
.bash_logout
.bashrc
.motd_shown
.profile
.sudo_as_admin_successful
.wget-hsts
Miniconda3-py39_4.12.0-Linux-x86_...
OUTLINE
TIMELINE
WSL: Ubuntu
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text

Q&A