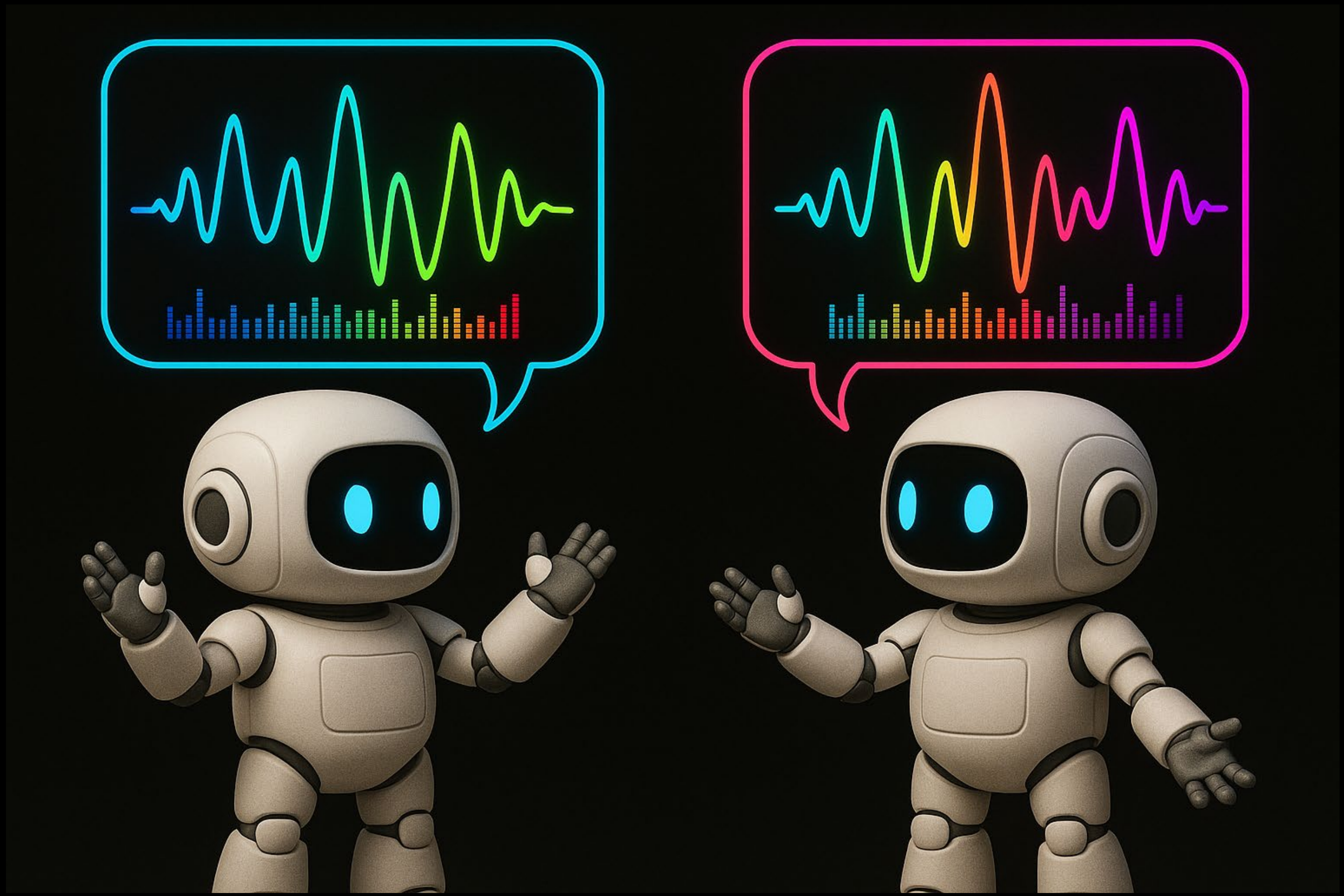


Introduction to Audio and Speech Processing

Understanding and Processing Spoken Language

ITAI 2373 Mod 03

Learning Objectives

By the end of this module, you will be able to:

Explain the fundamental properties of sound and speech

Describe how audio is digitally represented and processed

Apply basic audio preprocessing techniques

Extract and interpret acoustic features from speech

Understand the principles of speech-to-text and text-to-speech systems

Implement basic audio processing using Python libraries

Recap Mod 02

What We Covered

- **Text Preprocessing** - Cleaning & preparing text data
- **Audio as Language** - Processing speech signals
- **Multimodal Connection** - Converting between text ↔ audio

Why It Matters

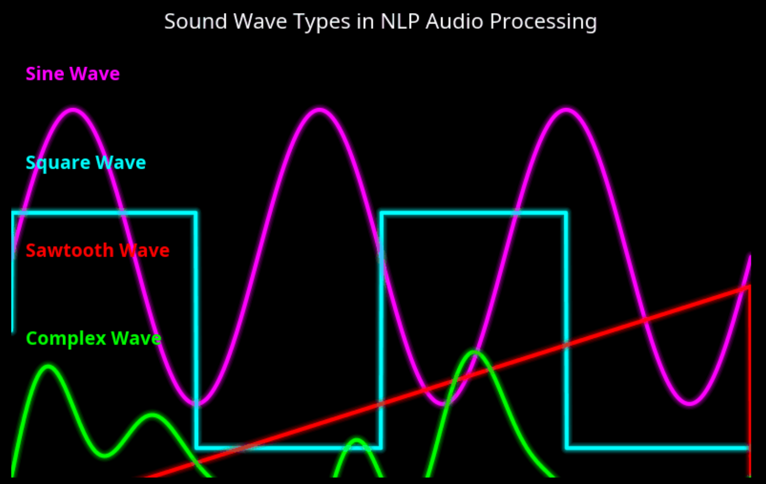
- Modern NLP needs both modalities
- Career advantage in multimodal AI
- Foundation for voice interfaces

Key Insight

Text	**Audio**
Discrete symbols	Continuous signals
Visual	Temporal

Fundamentals of Sound

- **Sound as a Wave:**
  - Compression and rarefaction of air molecules
  - Longitudinal wave propagation
- **Key Properties:**
  - **Frequency**: Number of cycles per second (Hertz, Hz)
  - **Amplitude**: Intensity or loudness of sound
  - **Phase**: Position in the cycle at a specific time
  - **Duration**: Length of time the sound persists

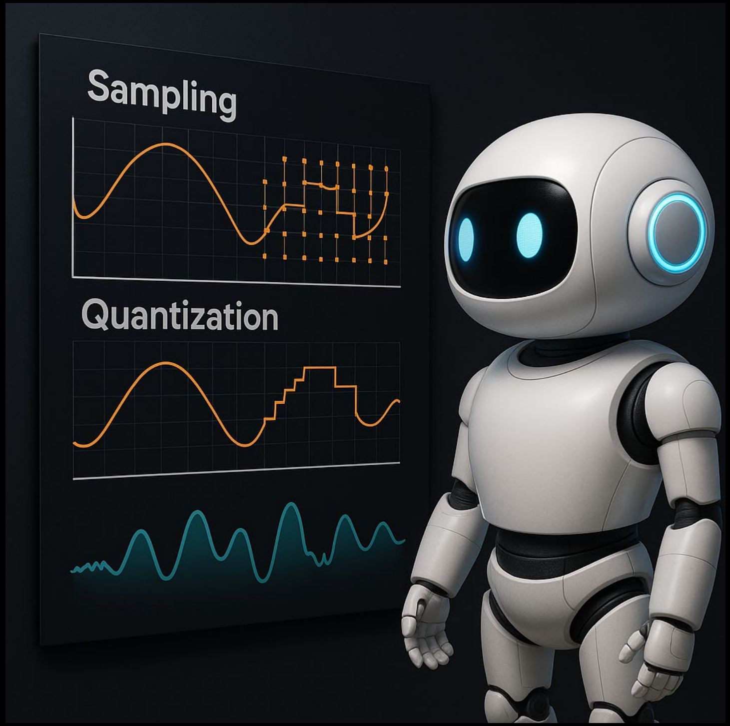


Fundamentals of Sound

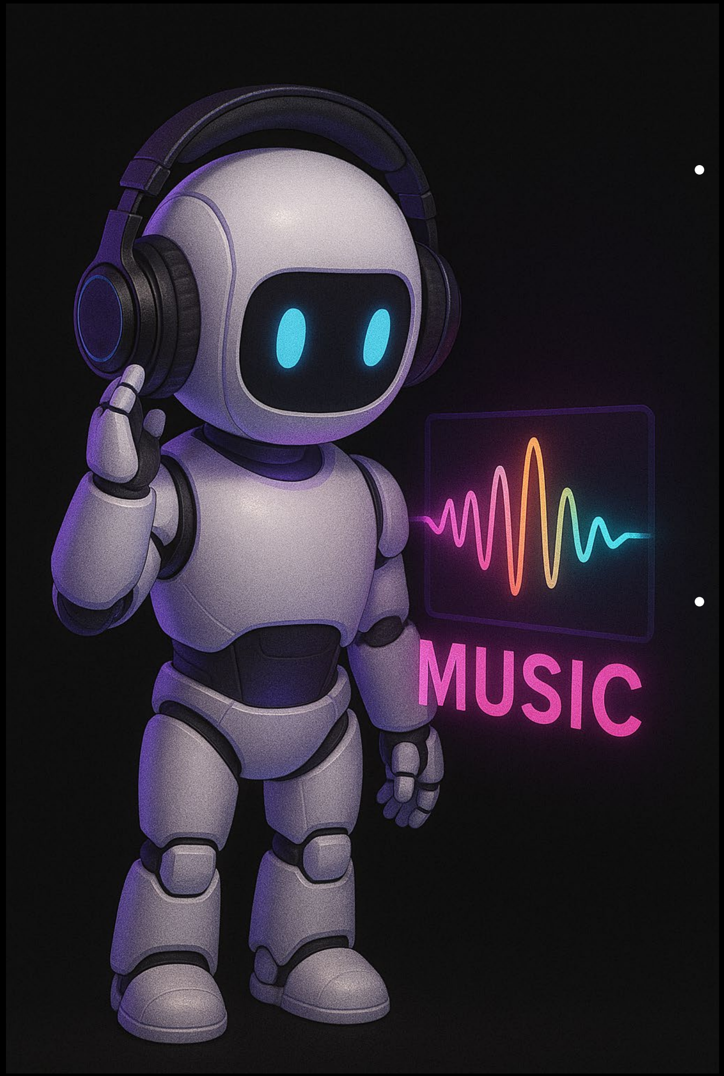
- **Human Hearing Range:**
  - Frequency: ~20 Hz to 20,000 Hz
  - Most sensitive: 2,000-5,000 Hz (speech range)

Digital Representation of Audio

- **Analog to Digital Conversion:**
  - **Sampling**: Measuring amplitude at fixed time intervals
    - Sampling Rate: Number of samples per second (e.g., 16kHz, 44.1kHz)
    - Nyquist Theorem: Sample at least twice the highest frequency
  - **Quantization**: Assigning discrete values to each sample
    - Bit Depth: Number of bits per sample (e.g., 16-bit, 24-bit)
    - Higher bit depth = more precise amplitude values



Digital Representation of Audio



- **Audio Data Formats:**
  - Uncompressed: WAV, AIFF
  - Compressed: MP3, AAC, OGG
  - Lossless: FLAC, ALAC
- **Real-world Example:**
  - CD Audio: 44.1kHz sampling rate, 16-bit depth, ~10MB per minute
  - Speech Recognition: 16kHz sampling rate, 16-bit depth, ~2MB per minute

Speech Production - The Mechanics

Source → Vocal cords generate sound waves

Filter → Vocal tract shapes the sound

Articulation → Tongue, lips, jaw control output

Source-Filter Model

Vocal Cords (vibrate) → **Vocal Tract** (resonates) → **Speech**

Different vocal tract shapes = Different sounds

Speech Production - The Structure

- **Speech Building Blocks**
  - 🔢 Phonemes
    - Basic sound units (~44 in English)
  - 🔗 Syllables
    - Phoneme combinations
  - 💬 Words
    - Syllable combinations
  - 🎵 Prosody
    - Rhythm • Stress • Intonation
- **Bottom Line**
  - Speech = Physical process + Linguistic structure

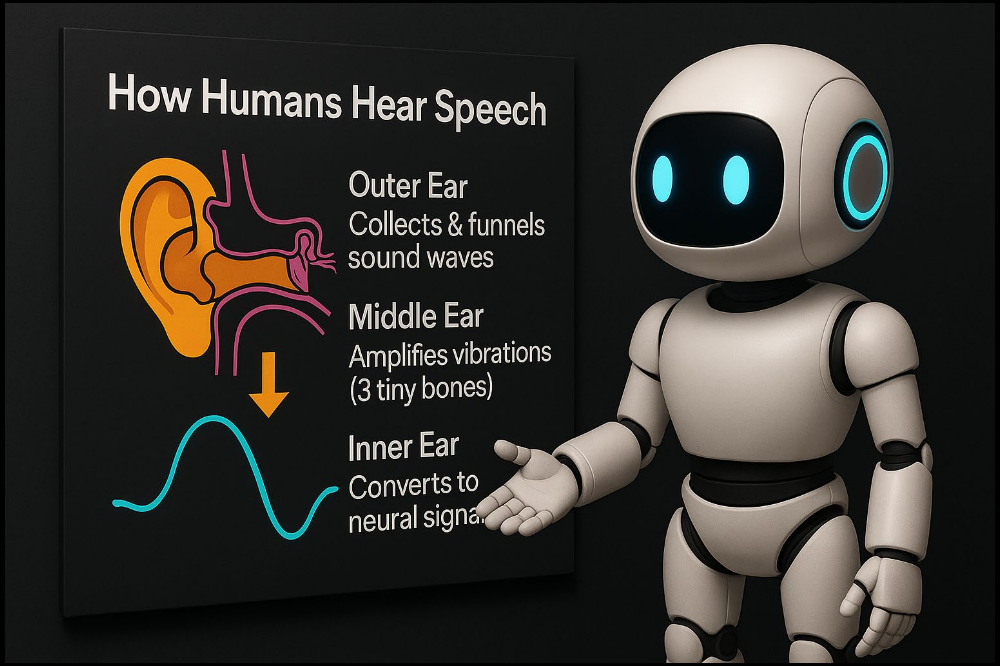
How We Hear Speech

The Auditory System

- **Outer Ear** → Collects & funnels sound waves
- **Middle Ear** → Amplifies vibrations (3 tiny bones)
- **Inner Ear** → Converts to neural signals

Frequency Analysis

- **Cochlea** = Natural spectrum analyzer
- Different regions detect different frequencies
- Creates "tonotopic map" in your brain



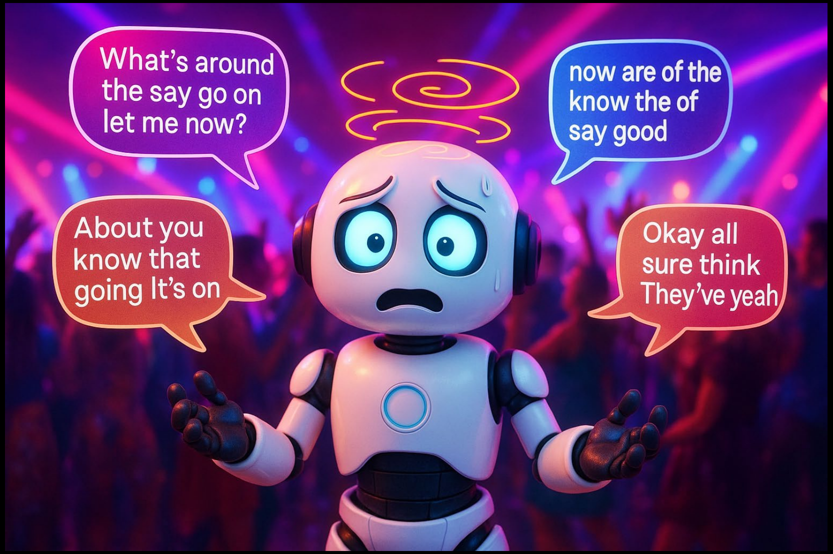
Speech Perception Challenges

What Makes Hearing Hard

- 🎉 **Cocktail Party Effect** - Focus on one voice in noise
- 🗣️ **Speaker Normalization** - Understand different voices
- 🔗 **Coarticulation** - Sounds blend together
- 📊 **Categorical Perception** - Hear discrete sounds

Why This Matters for AI

- Speech recognition mimics human perception
- Understanding hearing → Better algorithms
- Current limitations mirror human challenges



Acoustic Features - Waveforms

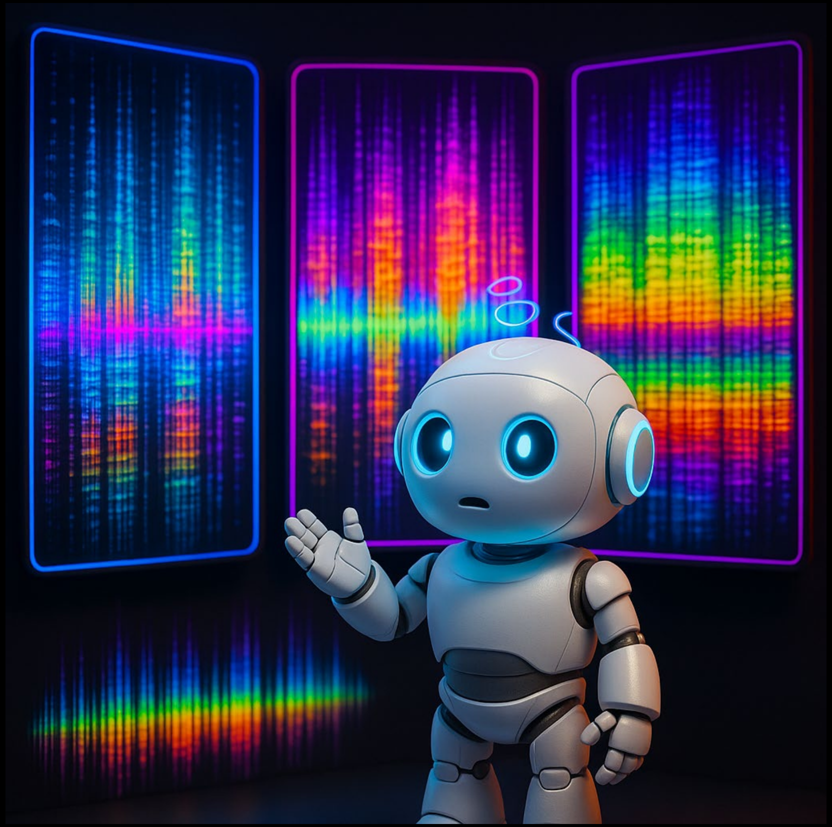
- **Time-Domain Representation**:
  - Amplitude vs. Time - raw audio signal
  - Shows volume changes over time
- **What We Can Extract:**
  - Energy levels (loudness patterns)
  - Duration (length of sounds)
  - Zero-crossing rate (frequency indicator)
- **Limitations for NLP:**
  - Hard to identify specific words/phonemes
  - Sensitive to noise and speaker variations
  - Limited frequency information

```python
import librosa
import matplotlib.pyplot as plt

# This loads a WAV file and gives us the waveform
y, sr = librosa.load('speech.wav', sr=16000)

# y is now an array of amplitude values
# sr is the sampling rate (16,000 samples per second)

# This creates a visual representation
librosa.display.waveshow(y, sr=sr)
plt.title('Speech Waveform - Our First Look at Audio')
```



Acoustic Features - Spectrograms

- **Time-Frequency Representation:**
  - Shows **what frequencies** are present **when**
  - Color/brightness = intensity at each frequency
  - [Visual: Example spectrogram with speech patterns]
- **How It's Made:**
  - Short-Time Fourier Transform (STFT)
  - Divides audio into overlapping frames
  - Analyzes frequency content of each frame
- **Reading Spectrograms:**
  - Horizontal axis = Time
  - Vertical axis = Frequency
  - Dark bands = Formants (vocal tract resonances)

```
# Create spectrogram
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)))
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz')
```

Acoustic Features - MFCCs (Part 1)

Mel-Frequency Cepstral Coefficients:

Most important features for NLP speech processing

Compact representation of speech spectrum

Designed to match human hearing

Why MFCCs Matter for NLP:

Standard input for speech recognition systems

Bridge between audio signal and text processing

Capture vocal tract information (phonemes)

Relatively noise-robust

The Process (Simplified):

Spectrogram → 2. Mel filtering → 3. Log → 4. DCT → 5. Keep ~13 coefficients

Extracting MFCCs with Librosa

Code Example

```python
# Extract MFCCs (standard for speech recognition)
mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)

# Plot MFCCs
plt.figure(figsize=(10, 4))
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
plt.title('MFCCs - Ready for NLP Processing')
```

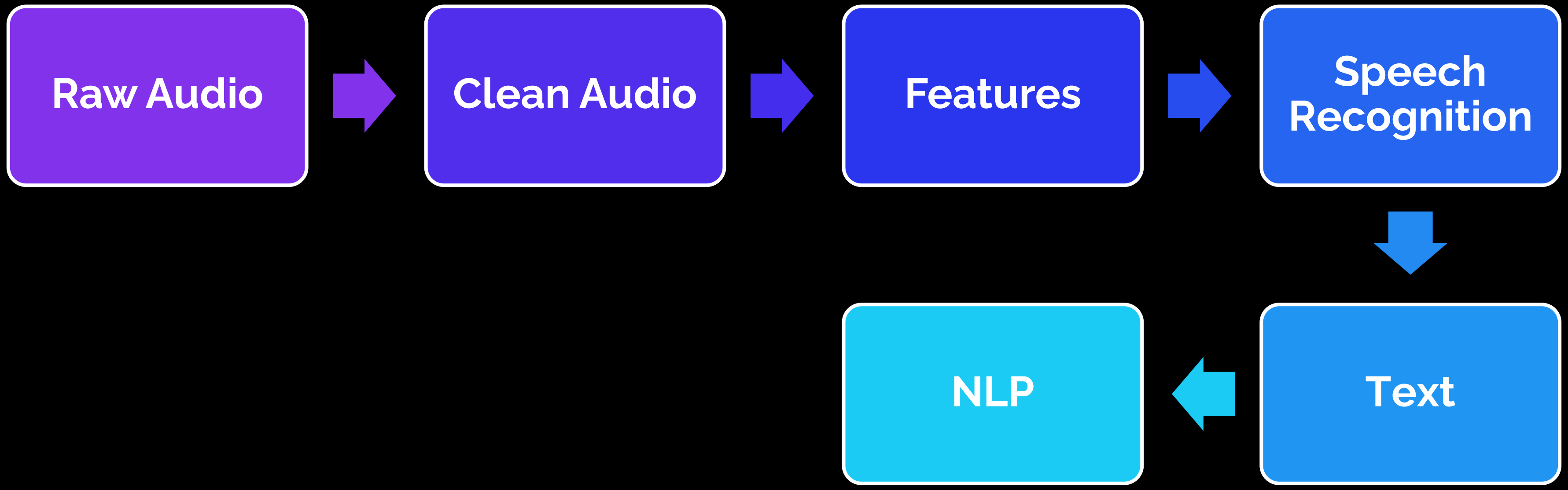
- **MFCCs:**
  - The Standard for Speech Recognition
- **Key Parameters:**
  - n_mfcc=13 → Standard number of coefficients
  - First coefficient (MFCC-0) → Overall energy
  - Higher coefficients → Fine spectral details
- **Connection to Text Processing:**
  - MFCCs are like "word features" for audio
  - Feed into speech recognition → get text
  - Then apply text NLP techniques!

Audio Preprocessing Overview

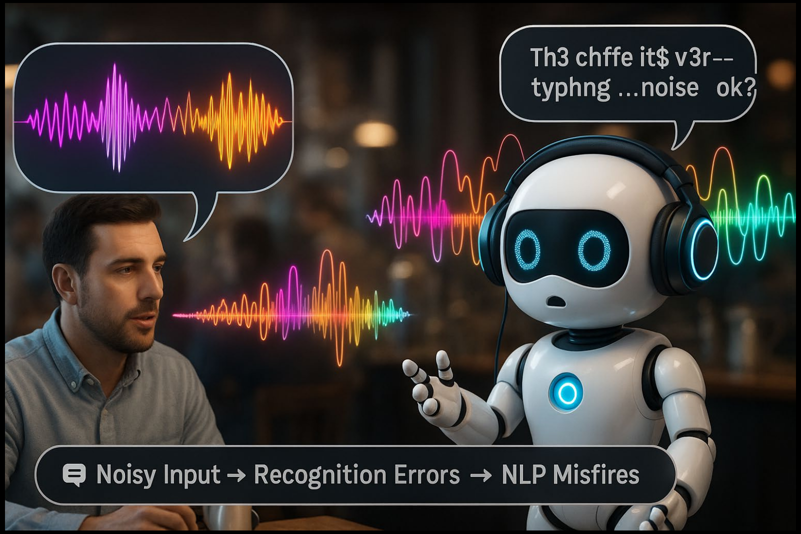
- **Why Preprocess Audio?**
  - Clean signal for better speech recognition
  - Standardize input across speakers/conditions
  - Prepare for NLP text processing pipeline
- **Preprocessing Steps:**
  - **Noise Reduction** → Remove background sounds
  - **Normalization** → Standardize volume levels
  - **Segmentation** → Split into words/utterances
  - **Feature Extraction** → Convert to MFCCs

Audio Preprocessing for NLP

- **Parallel to Text Preprocessing:**
  - Audio noise removal ≈ Text cleaning
  - Audio normalization ≈ Text normalization
  - Audio segmentation ≈ Text tokenization
- **Pipeline:**



Audio Preprocessing – Noise Reduction for Speech Processing



Common Noise Types:

- Background chatter (coffee shops, offices)
- Electronic interference (buzzing, humming)
- Environmental sounds (traffic, AC, typing)
- Recording artifacts (poor mics, compression)

Why Computers Struggle More Than Humans:

- Humans have natural "cocktail party effect"
- Computers hear everything equally
- No selective attention filtering

Impact on NLP Applications:

- Noisy audio → Poor speech recognition
- Poor recognition → Wrong text for NLP
- Garbage in, garbage out principle

Noise Reduction Techniques: Cleaning Audio for Better Speech Recognition

- **Spectral Subtraction:**
  - Learn noise pattern during silence
  - Subtract noise "fingerprint" from speech
  - Works well for constant background noise
- **Adaptive Filtering:**
  - Real-time noise adjustment
  - Handles changing noise environments
- **Simple Technique- Spectral Gating:**

```
import noisereduce as nr
# Reduce background noise
clean_audio = nr.reduce_noise(y=noisy_audio, sr=sr)
```

- **Real-World Applications:**
  - Voice assistants in noisy homes
  - Online meeting platforms (Zoom, Teams)
  - Call center transcription

Audio Normalization - The Volume Problem

Volume Normalization:

Peak Normalization → Scale to max amplitude (Loudest Moment)

- Simple but sensitive to outliers

RMS Normalization → Scale by average energy (better)

- More natural, consistent results

Why Important for NLP:

Consistent input to speech recognition

Works across different microphones/speakers

Improves text extraction reliability

Speaker Normalization:

Compensate for different voice characteristics

Helps speech recognition work across users

- **Implementation:**

```
# RMS normalization
rms = np.sqrt(np.mean(y**2))
target_rms = 0.1
normalized = y * (target_rms / rms)
```

Audio Segmentation - Tokenization

- **The Challenge:**
  - No spaces between words in speech
  - Continuous audio stream
  - Need to find meaningful boundaries
- **Significance**
  - Focus processing on relevant audio
  - Prepare for speech recognition
  - Improve NLP pipeline efficiency
- **Types of Segmentation:**
  - **Voice Activity Detection (VAD)** - Speech vs. silence
  - **Utterance Segmentation** - Phrases/sentences
  - **Word Segmentation** - Individual word
- **Example:**
  - "Iscreamforicecream" → "I scream for ice cream"

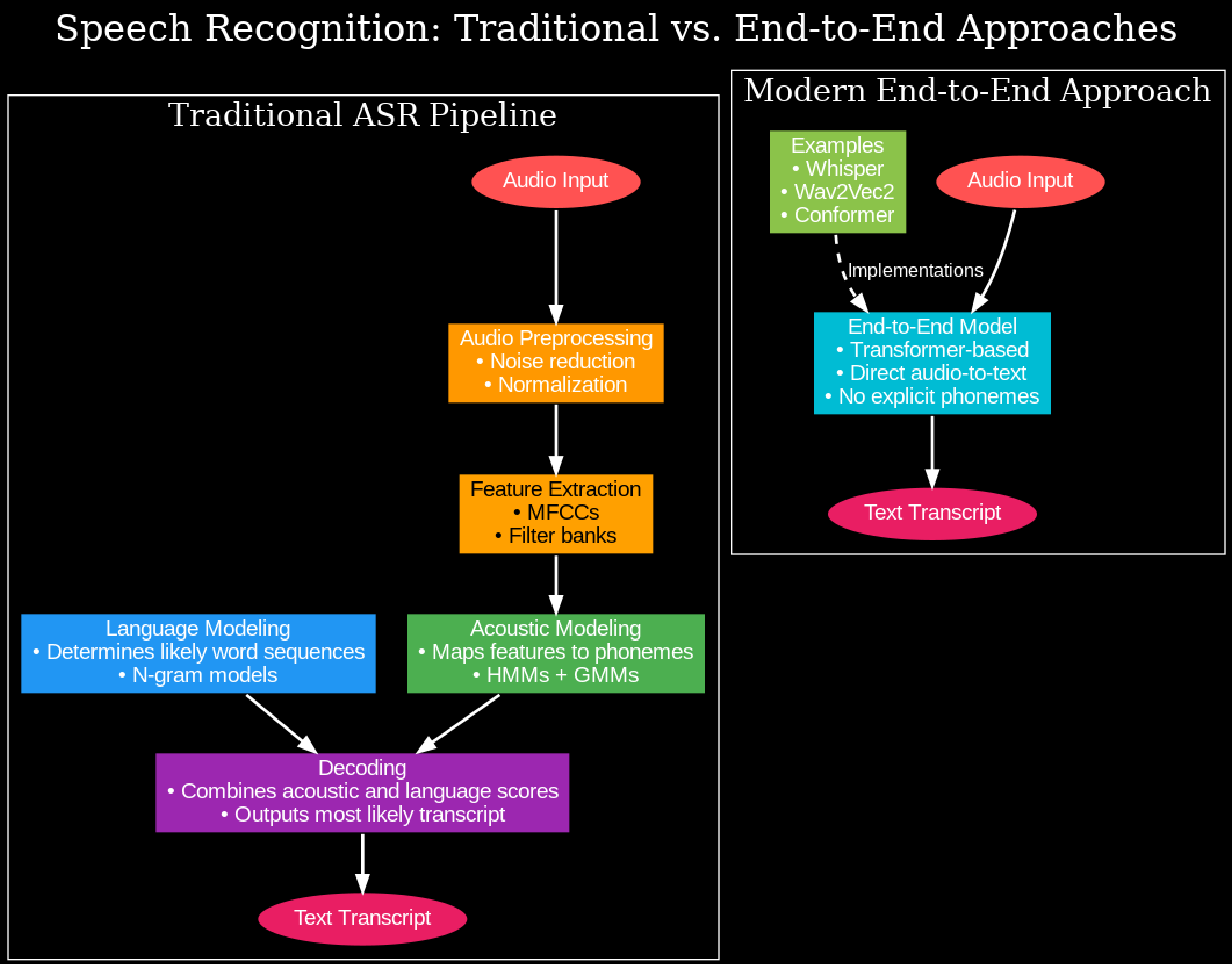
Voice Activity Detection – VAD Detecting Speech vs. Silence

- **What is VAD?**
  - Automatic detection of speech segments
  - Filter out silence and background noise
  - Like smart "mute button" for processing
- **Simple Energy-Based Approach:**

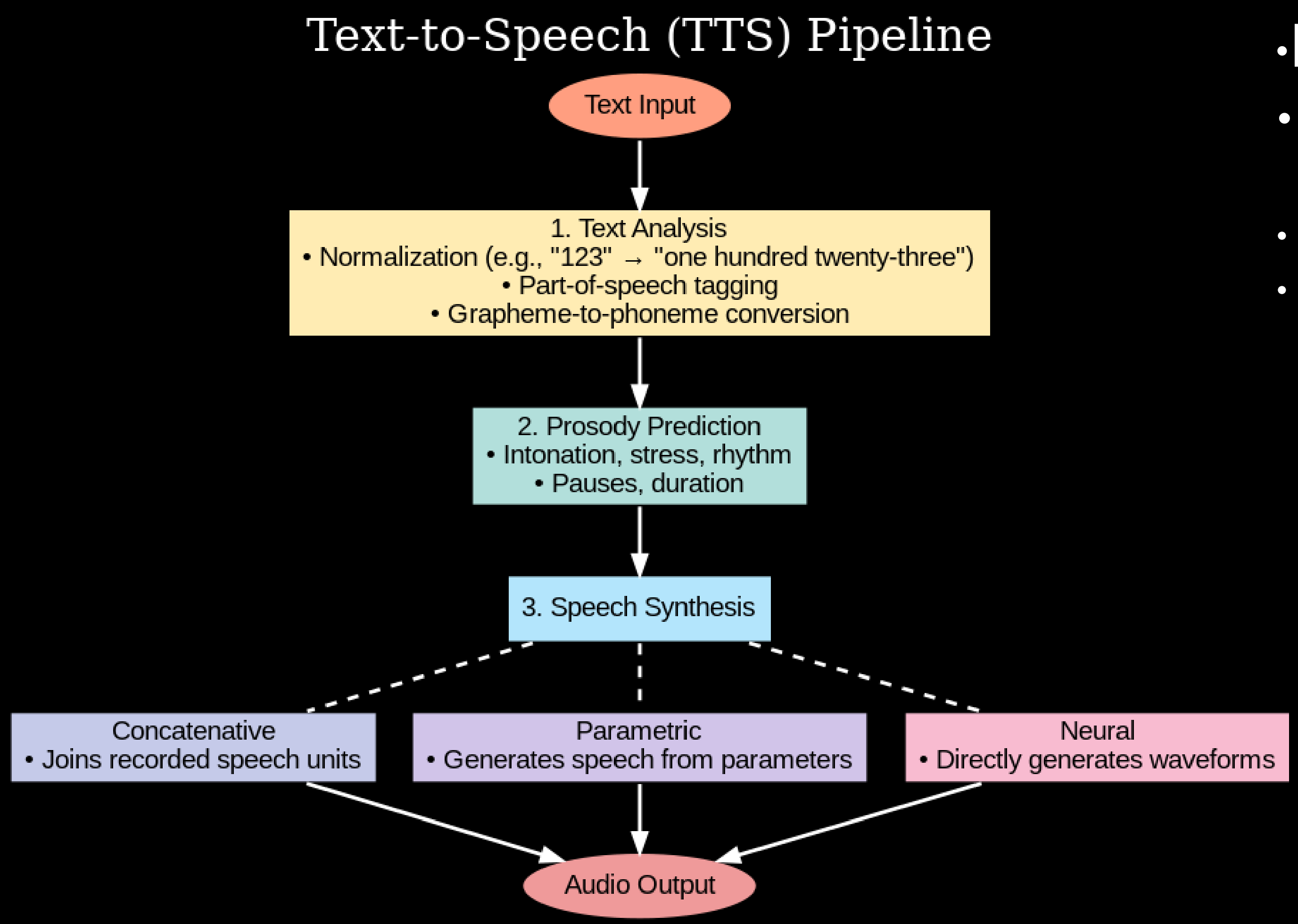
```
# Calculate energy
energy = librosa.feature.rms(y=y)[0]
 # Set threshold
 speech_frames = energy > 0.01
```

- **Applications:**
  - Voice assistants (when you stop talking)
  - Call center analytics
  - Meeting transcription systems
- **Advanced Methods:**
  - Multi-feature VAD (energy + spectral features)
  - Machine learning-based detection

Automatic Speech Recognition (ASR)



Text-to-Speech Systems (TTS)

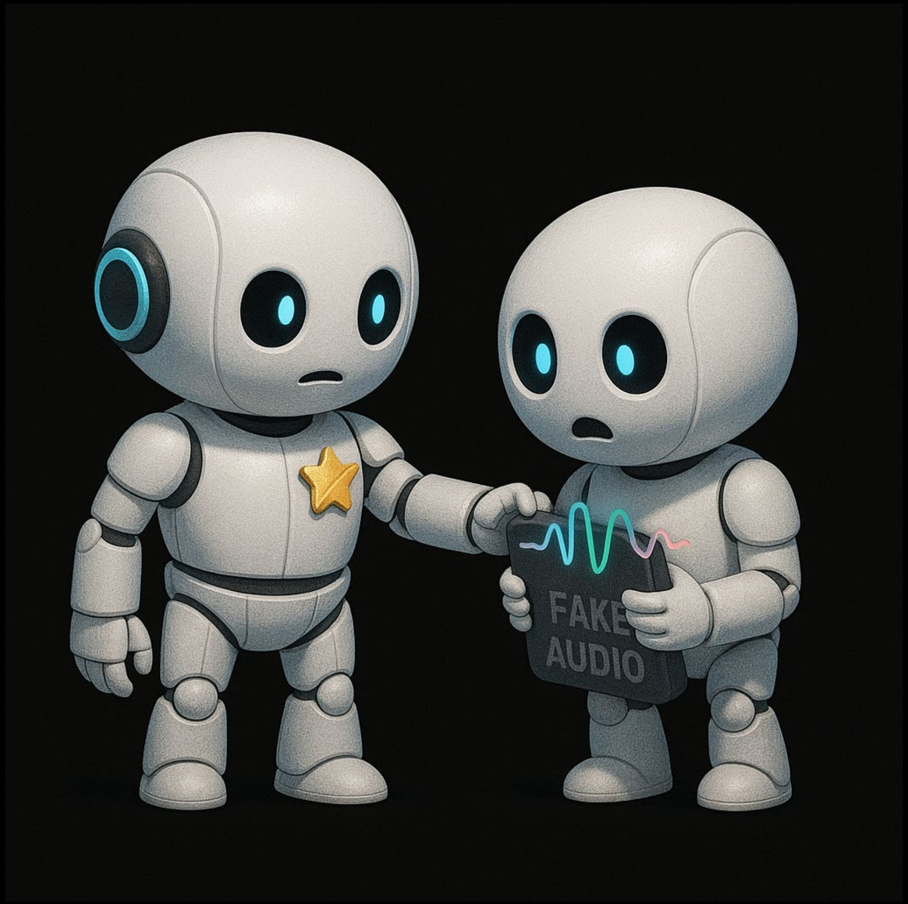


·Modern Neural TTS:

- End-to-end models (Tacotron, WaveNet, FastSpeech)
- Voice cloning and style transfer
- Emotional and expressive speech

Ethics in Audio Processing

- **Privacy Concerns:**
  - Voice contains biometric information
  - Always-on listening devices
  - Voice cloning potential for misuse
- **Bias in Speech Recognition:**
  - Performance varies by accent, gender, age
  - Training data representation issues
  - Impact on accessibility and fairness
- **Responsible Development:**
  - Diverse training data
  - Transparent data use policies
  - User control over voice data
  - Regular bias auditing



Key Takeaways

🎵 Audio Fundamentals:

- Sound waves → Digital representation → ML features
- Progressive sophistication: Waveforms → Spectrograms → MFCCs

🔧 Preprocessing Pipeline:

- Essential steps: Noise reduction → Normalization → Segmentation
- Quality principle: Clean audio = Better speech recognition = Better NLP

🗣️ Speech Systems:

- **ASR**: Converts speech to text (enables text NLP on spoken language)
- **TTS**: Converts text to speech (enables voice interfaces)

💼 Real-World Impact:

- Powers voice assistants, accessibility tools, business analytics
- Foundation for multimodal NLP applications

🚀 Your New Skills:

- Ready to build voice-enabled NLP systems!