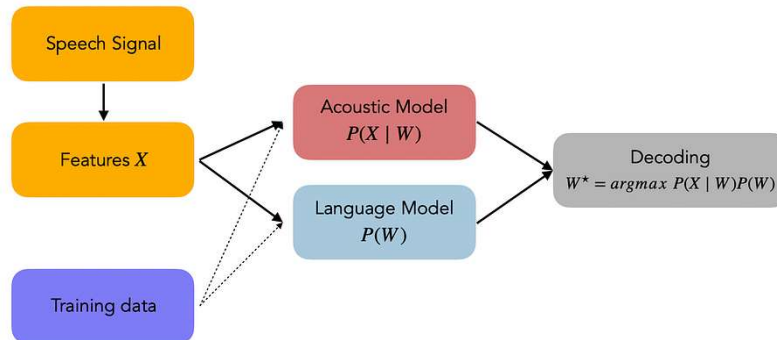


Speech Recognition



Speech recognition is not just about the algorithms we apply to get a list of words from an audio file. It is a complex topic that includes several key ideas and in the following article I will cover the most important one.

| *How do sounds appear?*

How Does the Human Body Produce Voice and

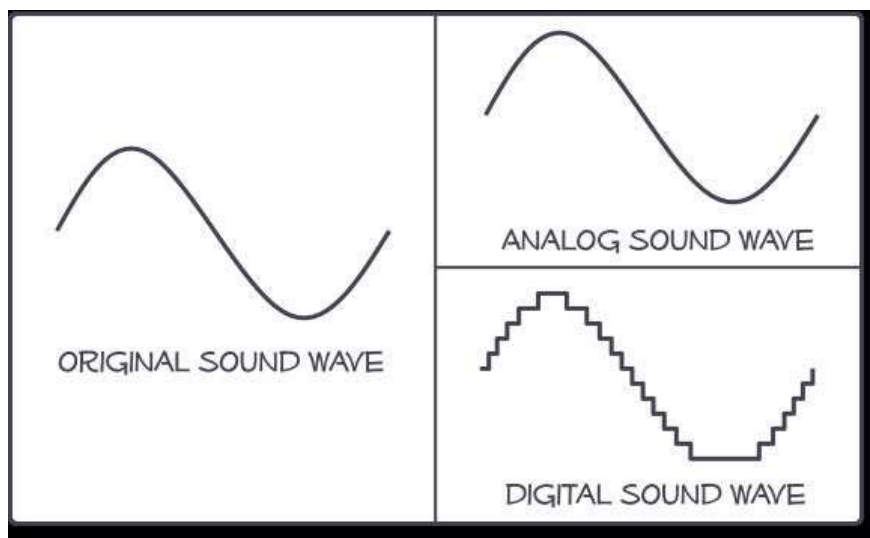
The primary source of speech sounds in most languages is the human vocal tract. It consists of the lungs, larynx (voice box), pharynx, oral cavity (mouth), and nasal cavity. Different speech sounds are produced by manipulating airflow, vocal cord¹ vibrations, and the shape of the vocal tract.



From analog to digital representation

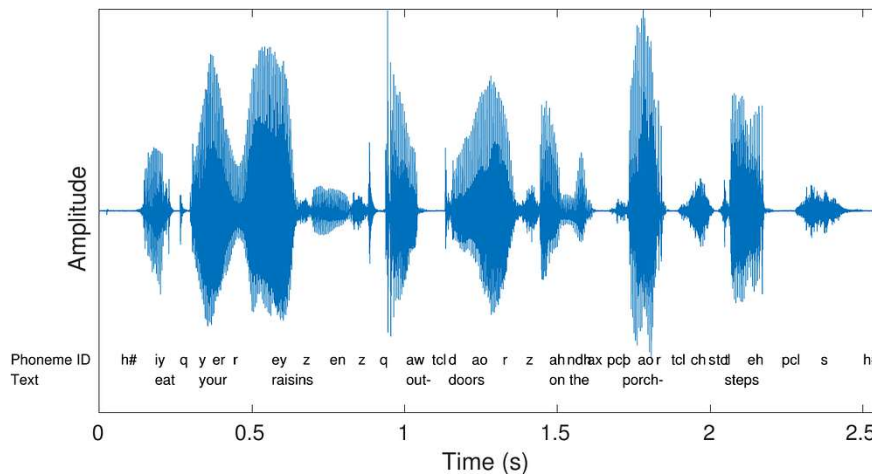
The first step in analog-to-digital conversion is sampling, where the continuous analog speech signal is converted into discrete samples taken at regular intervals.

Analog sound is a continuous waveform that directly represents the variations in air pressure caused by sound vibrations. In the context of audio, analog signals are often electrical voltage fluctuations that mimic the original sound wave. Analog audio technology has been in use for centuries and includes mechanisms like vinyl records, cassette tapes, and analog telephony.



Digital sound is a representation of sound in discrete, numerical form. Digital audio technology involves converting analog sound waves into a series of discrete samples, which are then encoded and stored as binary data (0s and 1s). Digital audio is used in formats like CDs, MP3s, and streaming services.

The sampling rate, such as 8 kHz or 16 kHz, determines how frequently these samples are taken, representing the original analog signal more accurately in the digital domain.



After sampling the analog signal, the next step is quantization. This involves converting each sampled value from the continuous amplitude² range into a discrete digital value using binary numbers. The bit depth determines the number of bits used to represent each sample, with higher bit depths providing more precision but requiring more storage and resources. Common bit depths in speech recognition are 8, 16, and 24 bits. The digital values are then encoded into formats like PCM, used widely in speech recognition. The result is a digital signal composed of discrete samples, which can be stored or processed by speech recognition algorithms.

Analog to Digital Converters | Digital Signal Pro

Fast Fourier Transform (FFT)

Speech signals are time-varying, and the information about the sounds and phonemes⁴ is often encoded in the frequency³ domain. FFT allows us to transform the speech signal from the time domain to the frequency domain, revealing the specific frequencies and their magnitudes present in the signal.

Can you guess the song? Fourier Music Decon

Feature Extraction

The crucial speech recognition process of feature extraction includes breaking down the digital voice signal into a collection of significant and representative features. These characteristics act as inputs to voice recognition algorithms, enabling them to efficiently recognize and distinguish between various spoken words or phonemes. The following steps are commonly involved in the feature extraction process:


Prerequisites: The digital voice signal may go through some preprocessing stages to improve its quality and reduce noise before feature extraction. Filtering, normalizing, and silence elimination are common preprocessing methods.

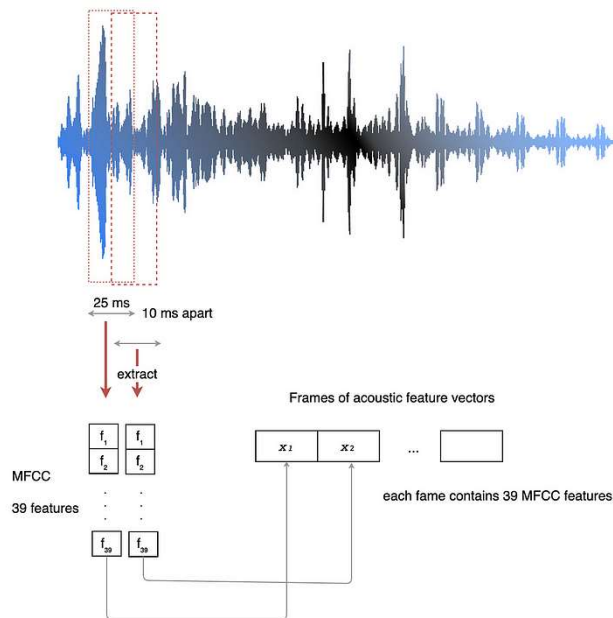
Frame blocking is the process of breaking up a continuous digital speech output into smaller, overlapping frames. A brief segment of the signal, often lasting 20 to 30 milliseconds, makes up each frame. When frames overlap, the flow of information is maintained.

Speech Recognition—Feature Extraction MFCC
& PLP



Machine learning ML extracts features from raw data and creates a dense representation of the...
jonathan-hui.medium.com





Acoustic Modeling


Input:

A speech signal.

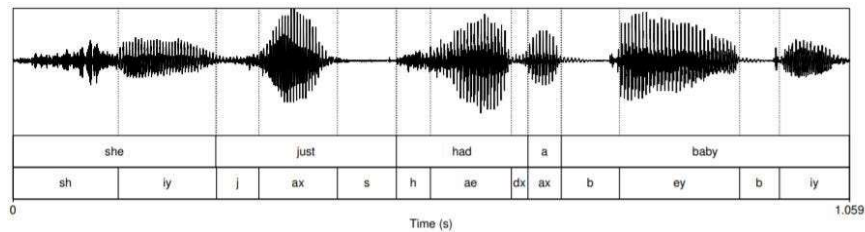
Output:

A recognized sequence of phones (the basic unit of speech).

A Basic Introduction to Speech Recognitio...

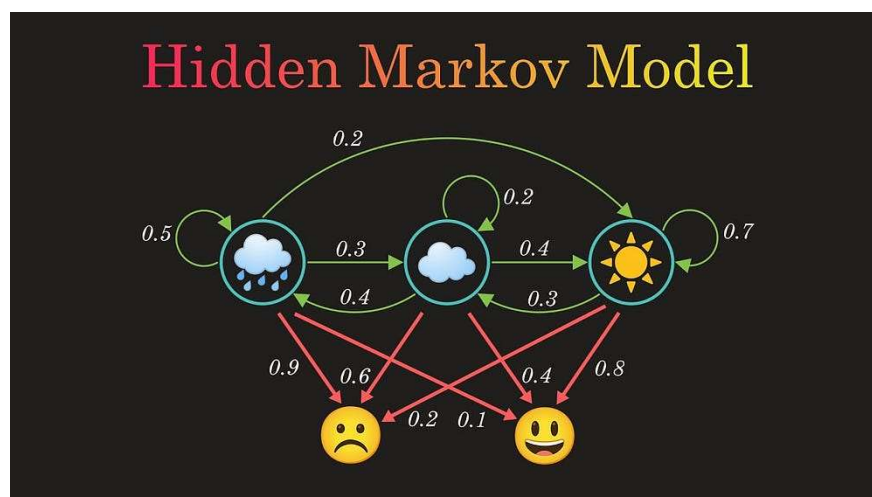


In voice recognition systems, acoustic models are a particular kind of machine-learning model. The model is trained to identify the acoustic features of human speech and provide an appropriate text transcription. Huge datasets of spoken words, phrases, and sentences are often used to train acoustic models that ‘map’ the sounds of human speech.



Hidden Markov Model (HMM)

A Hidden Markov Model (HMM) is a mathematical model that deals with sequences of observations, where each observation is tied to a hidden state. The model uses probabilities to transition between states and emit observations. It's commonly used in tasks like speech recognition or DNA sequence analysis. The HMM's goal is to find the best sequence of hidden states that explains the observed data. It learns from training data and is used for decoding sequences in various applications.



Example of how we can implement HMM

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from hmmlearn import hmm

# Define the state space
states = ["Word0", "Word1", "Word2", "Word3"]
n_states = len(states)

# Define the observation space
observations = ["Loud", "Soft"]
n_observations = len(observations)

# Define the initial state distribution
start_probability = np.array([0.8, 0.1, 0.1, 0.0])

# Define the transition probabilities
transition_probability = np.array([
    [0.7, 0.2, 0.0, 0.1],
    [0.0, 0.6, 0.4, 0.0],
    [0.0, 0.0, 0.0, 0.0],
    [0.0, 0.0, 0.0, 0.0]
])

# Define the observation likelihoods
emission_probability = np.array([
    [0.7, 0.3],
    [0.4, 0.6],
    [0.6, 0.4],
    [0.3, 0.7]
])

# Fit the model
model = hmm.CategoricalHMM(n_components=n_states)
model.startprob_ = start_probability
model.transmat_ = transition_probability
model.emissionprob_ = emission_probability

# Define the sequence of observations
observations_sequence = np.array([0, 0, 0, 0, 1, 0])

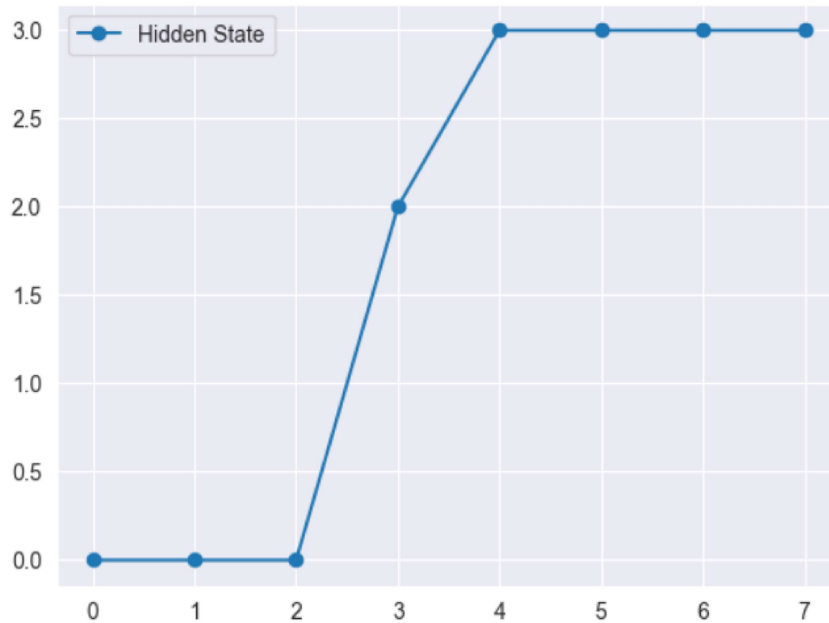
# Predict the most likely hidden states
hidden_states = model.predict(observations_sequence)
print("Most likely hidden states:", hidden_states)

# Plot the results
sns.set_style("darkgrid")
plt.plot(hidden_states, '-o', label="Hidden States")

```

```
plt.legend()
plt.show()
```

Most likely hidden states: [0 0 0 2 3 3 3 3]



- In our case (“Speech recognition”) observation is the audio file that we take as input, and the states are the [phonemes,words] we need to find out.

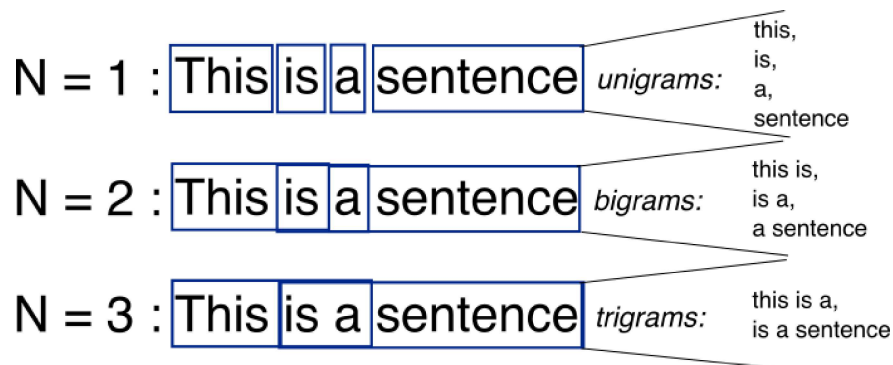
Language Modeling

- A language model is a probabilistic model that assigns probabilities to any sequence of words.
- A good language model has to assign high probabilities to well performed sentences.

Statistical Approach (N-grams):

- Traditional language models, like N-gram models, estimate the likelihood of a word based on its previous N-1 words.
- The model calculates probabilities of word sequences based on the frequency of co-occurrences in the training data.

- For example, a bigram model predicts the next word based on the previous word, while a trigram model considers the previous two words.



Decoding

- Combine the outputs of the acoustic model and the language model to decode the most likely transcription for the given audio.
- Apply algorithms like Viterbi or beam search to find the best sequence of words that match the audio features.

The code below shows how the Viterbi algorithm works in the context of speech recognition.

```
# Define possible observations and states
obs = ("sound1", "sound2", "sound3")
states = ("Word1", "Word2")

# Define initial state probabilities
start_p = {"Word1": 0.6, "Word2": 0.4}

# Define transition probabilities between states
trans_p = {
    "Word1": {"Word1": 0.7, "Word2": 0.3},
    "Word2": {"Word1": 0.4, "Word2": 0.6},
}

# Define emission probabilities of observations
emit_p = {
    "Word1": {"sound1": 0.5, "sound2": 0.4, "sound3": 0.1},
    "Word2": {"sound1": 0.1, "sound2": 0.3, "sound3": 0.6},
}
```

```

# Define the Viterbi algorithm function
def viterbi(obs, states, start_p, trans_p, emit_p):
    V = [{}]
```

Initialize Viterbi matrix

```

    # Initialization for t = 0
    for st in states:
        V[0][st] = {"prob": start_p[st] * emit_p[obs[0]]}

    # Run Viterbi for t > 0
    for t in range(1, len(obs)):
        V.append({})
        for st in states:
            max_tr_prob = V[t - 1][states[0]]["prob"]
            prev_st_selected = states[0]
            for prev_st in states[1:]:
                tr_prob = V[t - 1][prev_st]["prob"]
                if tr_prob > max_tr_prob:
                    max_tr_prob = tr_prob
                    prev_st_selected = prev_st

            max_prob = max_tr_prob
            V[t][st] = {"prob": max_prob, "prev": prev_st_selected}

    # Print the Viterbi matrix
    for line in dptable(V):
        print(line)

    # Backtrack to find the most likely sequence
    opt = []
    max_prob = 0.0
    best_st = None
    for st, data in V[-1].items():
        if data["prob"] > max_prob:
            max_prob = data["prob"]
            best_st = st
    opt.append(best_st)
    previous = best_st

    # Follow the backtrack till the first observation
    for t in range(len(V) - 2, -1, -1):
        opt.insert(0, V[t + 1][previous]["prev"])
        previous = V[t + 1][previous]["prev"]

    # Print the result
    print("The steps of states are " + " ".join(opt))

# Define a function to print the Viterbi matrix
def dptable(V):

```

```

        yield " " * 5 + " ".join("%3d" % i) for
        for state in V[0]:
            yield "%.7s: " % state + " ".join("%.7s"

# Call the Viterbi function with defined paramet
viterbi(obs, states, start_p, trans_p, emit_p)

```

```

      0      1      2
Word1: 0.30000 0.08400 0.00588
Word2: 0.04000 0.02700 0.01512
The steps of states are Word1 Word1 Word2 with highest probability of 0.01512

```

Output

A simple approach of speech recognition

```

# Import required libraries
from gtts import gTTS # For text-to-speech synt
import os
from os import path
from deep_translator import GoogleTranslator #
import speech_recognition as sp # For speech re

# Define a class named 'pevervodcik'
class pevervodcik:
    # Constructor with 'language' as a parameter
    def __init__(self, language):
        self.language = language

    # Method to capture audio input using a micr
    def get_audio(self):
        mic = sp.Microphone(device_index=0) # M
        r = sp.Recognizer() # Speech recognizer
        with mic as source:
            print("Spuneti acum") # Prompt for
            audio = r.listen(source) # Listen f
        try:
            c = r.recognize_google(audio, langua
        except:
            print("Something go wrong") # Error
            return c # Return recognized text

    # Method to translate the provided text
    def translate(self, c):

```

```

        # Translate the text using GoogleTransla
        translated = GoogleTranslator(source='au
        print(translated) # Print translated te
        return translated # Return translated t

    # Method to convert translated text to voice
    def voice(self, trans):
        # Convert translated text to speech usin
        myobj = gTTS(text=trans, lang=self.langu
        myobj.save("welcome.mp3") # Save the ge
        os.system("welcome.mp3") # Play the sav

# Create an instance of 'pevrvodcik' with targe
Ion = pevrvodcik('en')
# Capture audio input from the user
c = Ion.get_audio()
# Translate the captured text
trans = Ion.translate(c)
# Convert the translated text to speech and play
Ion.voice(trans)

# Note: Instead of using a microphone, the code

```

In the code above we have created a simple translator based on speech recognition algorithm. In our class „pevrvodcik,, we have 3 methods.

First method is responsible for recognizing the speech and saving it as an audio file.

Second method is responsible for translating the audio file according to the required language and saving it in an audio new file using GoogleTranslator library.

The last method plays the audio file using gTTS library.

Conclusion

In conclusion, speech recognition stands as a remarkable technological achievement that bridges the gap between human communication and machines. With the ability to convert spoken language into textual representation, speech recognition has

revolutionized the way we interact with technology and has opened doors to a plethora of applications across industries.

Key words:

Amplitude, Frequency, Vocal Cords, Phoneme.

[1]: The **vocal cords** (also called vocal folds) are two bands of smooth muscle tissue found in the larynx (voice box). The vocal cords vibrate and air passes through the cords from the lungs to produce the sound of your voice.

[2]: **Amplitude** is the maximum extent of a vibration or oscillation, measured from the position of equilibrium.

[3]: **Frequency** represents the number of times something happens within a particular period, or the fact of something happening often or a large number of times.

[4]: A **phoneme** is any of the perceptually distinct units of sound in a specified language that distinguish one word from another, for example *p*, *b*, *d*, and *t* in the English words *pad*, *pat*, *bad*, and *bat*.