**Bài 15: Nhận diện giọng nói phần 1 - Lập trình AI bằng Python**

Trong bài này, chúng ta sẽ tìm hiểu về nhận dạng giọng nói bằng AI với Python.

Lời nói là phương tiện giao tiếp cơ bản nhất của con người khi trưởng thành. Mục tiêu cơ bản của xử lý giọng nói là cung cấp sự tương tác giữa con người và máy móc.

Hệ thống xử lý giọng nói chủ yếu có ba tác vụ:

* Đầu tiên, nhận dạng giọng nói cho phép máy bắt các từ, cụm từ và câu ta nói
* Thứ hai, xử lý ngôn ngữ tự nhiên để cho phép máy hiểu những gì chúng ta nói và
* Thứ ba, tổng hợp giọng nói để cho phép máy nói.

Bài này tập trung vào nhận dạng giọng nói, quá trình hiểu các từ mà con người nói. Hãy nhớ rằng các tín hiệu giọng nói được ghi lại với sự trợ giúp của micrô và sau đó nó phải được hệ thống hiểu.

**1. Xây dựng hệ thống nhận dạng giọng nói :**

Nhận dạng giọng nói hoặc Nhận dạng giọng nói tự động (ASR) là trung tâm của sự chú ý đối với các dự án AI như robot. Nếu không có ASR, không thể tưởng tượng một robot nhận thức tương tác với con người. Tuy nhiên, việc xây dựng một công cụ nhận dạng giọng nói không phải là điều hoàn toàn dễ dàng.

Khó khăn trong việc phát triển hệ thống nhận dạng giọng nói

Phát triển một hệ thống nhận dạng giọng nói chất lượng cao thực sự là một bài toán khó. Khó khăn của công nghệ nhận dạng giọng nói có thể được mô tả chung theo một số khía cạnh như được thảo luận dưới đây:

* **Kích thước của từ vựng**: ảnh hưởng đến sự dễ dàng của việc phát triển một ASR.(kích thước từ vựng càng lớn thì việc nhận dạng càng khó.)
* **Đặc điểm của channel** - Chất lượng chanel cũng là một yếu tố quan trọng. Ví dụ, lời nói của con người có băng thông cao với dải tần đầy đủ, trong khi lời nói qua điện thoại bao gồm băng thông thấp với dải tần hạn chế. Lưu ý rằng nó khó hơn trong phần sau.
* **Chế độ nói**- Việc phát triển ASR dễ dàng cũng phụ thuộc vào chế độ nói, đó là liệu bài phát biểu có ở chế độ từ riêng biệt, hoặc chế độ từ được kết nối hay ở chế độ nói liên tục. Lưu ý rằng một bài phát biểu liên tục khó nhận ra hơn.
* **Phong cách nói**- Bài phát biểu được đọc có thể theo phong cách trang trọng, hoặc tự phát và đối thoại với phong cách bình thường. Cái sau khó nhận ra hơn.
* **Sự phụ thuộc người nó**i - Lời nói có thể phụ thuộc vào người nói, sự thích ứng của người nói hoặc độc lập với người nói. Rất khó nhất để xây dựng một diễn giả độc lập
* Loại tiếng ồn - Tiếng ồn là một yếu tố khác cần xem xét khi phát triển ASR. Tỷ lệ tín hiệu trên tiếng ồn có thể nằm trong nhiều phạm vi khác nhau, tùy thuộc vào môi trường âm thanh quan sát ít hơn so với nhiều tiếng ồn xung quanh   
  - Nếu tỷ lệ tín hiệu trên nhiễu lớn hơn 30dB, nó được coi là dải cao  
  - Nếu tỷ lệ tín hiệu trên nhiễu nằm trong khoảng từ 30dB đến 10db, nó được coi là SNR trung bình  
  - Nếu tỷ lệ tín hiệu trên nhiễu nhỏ hơn 10 dB, nó được coi là dải tần thấp
* **Đặc điểm của micrô** - Chất lượng của micrô có thể tốt, trung bình hoặc dưới trung bình. Ngoài ra, khoảng cách giữa miệng và micro phone có thể khác nhau. Các yếu tố này cũng cần được xem xét đối với hệ thống công nhận.

Bất chấp những khó khăn này, các nhà nghiên cứu đã làm việc rất nhiều trên các khía cạnh khác nhau của lời nói như hiểu tín hiệu giọng nói, người nói và xác định các trọng âm.

Bạn sẽ phải làm theo các bước dưới đây để tạo trình nhận dạng giọng nói

**2. Hình dung tín hiệu âm thanh - Đọc từ tệp và làm việc:**

Đây là bước đầu tiên trong việc xây dựng hệ thống nhận dạng giọng nói vì nó cung cấp hiểu biết về cách một tín hiệu âm thanh được cấu trúc. Một số bước phổ biến có thể làm theo để làm việc với tín hiệu âm thanh như sau:

**Recording :**

Đầu tiên, khi bạn phải đọc tín hiệu âm thanh từ một tệp, sau đó ghi lại bằng micrô.

**Sampling :**

Khi ghi âm bằng micrô, các tín hiệu được lưu trữ dưới dạng số hóa. Nhưng để hoạt động, máy cần chúng ở dạng số rời rạc. Do đó, chúng ta nên thực hiện lấy mẫu ở một tần số nhất định và chuyển đổi tín hiệu thành dạng số rời rạc. Việc chọn tần số cao để lấy mẫu ngụ ý rằng khi con người nghe tín hiệu, họ cảm thấy nó như một tín hiệu âm thanh liên tục.

**Ví dụ :**

Ví dụ sau đây cho thấy một cách tiếp cận từng bước để phân tích tín hiệu âm thanh, sử dụng Python, được lưu trữ trong một tệp. Tần số của tín hiệu âm thanh này là 44.100 HZ.

**import** numpy as np

**import** matplotlib.pyplot as plt

from scipy.io **import** wavfile

Bây giờ, hãy đọc tệp âm thanh được lưu trữ. Nó sẽ trả về hai giá trị: tần số lấy mẫu và tín hiệu âm thanh. Cung cấp đường dẫn của tệp âm thanh nơi nó được lưu trữ, như được hiển thị ở đây

frequency\_sampling, audio\_signal = wavfile.read("/Users/admin/audio\_file.wav")

Hiển thị các thông số như tần số lấy mẫu của tín hiệu âm thanh, loại dữ liệu của tín hiệu và thời lượng của nó, sử dụng các lệnh được hiển thị:

print('\nSignal shape:', audio\_signal.shape)

print('Signal Datatype:', audio\_signal.dtype)

print('Signal duration:', round(audio\_signal.shape[0] /

float(frequency\_sampling), 2), 'seconds')

Bước này liên quan đến việc chuẩn hóa tín hiệu như hình dưới đây :

audio\_signal = audio\_signal / np.power(2, 15)

Trong bước này, tôi sẽ trích xuất 100 giá trị đầu tiên từ tín hiệu này để hình dung. Sử dụng các lệnh sau cho mục đích này:

audio\_signal = audio\_signal [:100]

time\_axis = 1000 \* np.arange(0, len(signal), 1) / float(frequency\_sampling)

Hình dung tín hiệu bằng cách sử dụng các lệnh dưới đây:

**plt**.plot(time\_axis, signal, color='blue')

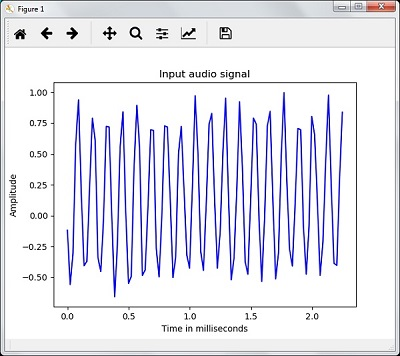
**plt**.xlabel('Time (milliseconds)')

**plt**.ylabel('Amplitude')

**plt**.title('Input audio signal')

**plt**.show()

Bạn sẽ có thể thấy đồ thị đầu ra và dữ liệu được trích xuất cho tín hiệu âm thanh ở trên như thể hiện trong hình ảnh tại đây :



Signal shape: (132300,)

Signal Datatype: int16

Signal duration: 3.0 seconds

**3. Đặc trưng cho tín hiệu âm thanh: Chuyển đổi sang dãy tần số**

Đặc trưng cho một tín hiệu âm thanh liên quan đến việc chuyển đổi tín hiệu miền thời gian thành miền tần số và hiểu các thành phần tần số của nó. Đây là bước quan trọng vì nó cho biết nhiều thông tin về tín hiệu. Bạn có thể sử dụng một công cụ toán học như Fourier Transform để thực hiện phép biến đổi này.

Ví dụ sau đây cho thấy, từng bước, cách mô tả đặc tính của tín hiệu, sử dụng Python, được lưu trữ trong một tệp. Lưu ý rằng ở đây mình đang sử dụng công cụ toán học Fourier Transform để chuyển đổi nó thành miền tần số.

**import** numpy as np

**import** matplotlib.pyplot as plt

from scipy.io **import** wavfile

Bây giờ, hãy đọc tệp âm thanh được lưu trữ. Nó sẽ trả về hai giá trị: tần số lấy mẫu và tín hiệu âm thanh. Cung cấp đường dẫn của tệp âm thanh nơi nó được lưu trữ như được hiển thị trong lệnh ở đây -

frequency\_sampling, audio\_signal = wavfile.read("/Users/admin/sample.wav")

Trong bước này, mình sẽ hiển thị các thông số như tần số lấy mẫu của tín hiệu âm thanh, loại dữ liệu của tín hiệu và thời lượng của nó, bằng cách sử dụng các lệnh được đưa ra bên dưới:

print('\nSignal shape:', audio\_signal.shape)

print('Signal Datatype:', audio\_signal.dtype)

print('Signal duration:', round(audio\_signal.shape[0] /

float(frequency\_sampling), 2), 'seconds')

chúng ta cần chuẩn hóa tín hiệu như sau :

audio\_signal = audio\_signal / np.power(2, 15)

Bước này liên quan đến việc trích xuất độ dài và nửa độ dài của tín hiệu :

length\_signal = len(audio\_signal)

half\_length = np.ceil((length\_signal + 1) / 2.0).astype(np.int)

Bây giờ, chúng ta cần áp dụng các công cụ toán học để biến đổi thành miền tần số. Ở đây mình đang sử dụng Biến đổi Fourier.

signal\_frequency = np.fft.fft(audio\_signal)

thực hiện chuẩn hóa tín hiệu miền tần số và bình phương nó

signal\_frequency = abs(signal\_frequency[0:half\_length]) / length\_signal

signal\_frequency \*\*= 2

Tiếp theo, trích xuất độ dài và nửa độ dài của tín hiệu được biến đổi tần số :

len\_fts = len(signal\_frequency)

Lưu ý rằng tín hiệu biến đổi Fourier phải được điều chỉnh cho trường hợp chẵn cũng như lẻ.

if length\_signal % 2:

signal\_frequency[1:len\_fts] \*= 2

else:

signal\_frequency[1:len\_fts-1] \*= 2

Bây giờ, trích xuất công suất bằng decibel (dB) -

signal\_power = 10 \* np.log10(signal\_frequency)

Điều chỉnh tần số tính bằng kHz cho trục X -

x\_axis = np.arange(0, len\_half, 1) \* (frequency\_sampling / length\_signal) / 1000.0

**plt**.figure()

**plt**.plot(x\_axis, signal\_power, color='black')

**plt**.xlabel('Frequency (kHz)')

**plt**.ylabel('Signal power (dB)')

**plt**.show()

