Relatório: Análise de Características Temporais em Sinais de Áudio – ATD

 # Notas a ter em conta

Devido ao facto de o MATLAB não se encontrar a funcionar no meu computador, tentei fazer a primeira meta em Python.

Sem conhecer a razão, o meu Python não assume os ficheiros a partir do número 29, tentei com outras pastas e nada. Ou seja, mostro o plot de os 29 ficheiros e os boxplots destes 29.

 # Introdução

Neste projeto, realizei uma análise das características temporais em sinais de áudio. O objetivo é identificar características que possam ser úteis para diferenciar esses áudios. Para isso, importei os sinais de áudio, realizei um pré-processamento para garantir consistência na duração e amplitude dos sinais, e em seguida, calculei e visualizei diferentes características temporais.

 # Importação dos Sinais de Áudio

Inicialmente, realizei a importação dos sinais de áudio utilizando a biblioteca `librosa`. Utilizei a função `librosa.load()` para carregar os arquivos de áudio, passando o caminho do diretório onde os arquivos estão localizados. Um exemplo de sinal de áudio foi submetido a plot para verificar se a importação estava a funcionar corretamente.

```python
import os
import librosa
import matplotlib.pyplot as plt
import numpy as np

path = r'C:\Users\carli\OneDrive\ATD_23-24\01'

# 1) Show one example to see if it´s working
audioExample, samplingRate = librosa.load(os.path.join(path, "0_01_0.wav"), sr=None)
plt.figure(figsize=(10, 4))
plt.plot(np.linspace(0, len(audioExample) / samplingRate, len(audioExample)), audioExample)
plt.xlabel('Tempo (s)')
plt.ylabel('Amplitude')
plt.title('Exemplo de Sinal de Áudio')
plt.grid(True)
plt.show()
```

Fig. 1 -> Exemplo de áudio, para ver se tudo decorria da forma correta.

```
for x in range(10):
    for y in range(50):
        fileName = os.path.join(path, f"{x}_01_{y}.wav")
```

Fig. 2 -> Loops para aceder a todos os ficheiros áudio.

 # Pré-Processamento dos Sinais de Áudio

Após a importação, realizei um pré-processamento dos sinais de áudio para garantir que todos tinham a mesma duração e amplitude. Isso envolve a remoção do silêncio inicial, a normalização da amplitude e a adição ou remoção de silêncio para ajustar a duração dos sinais.

```
# Pre-processing: removal of initial silence, normalization of amplitude and addition of silence at the end
audio = librosa.effects.trim(audio)[0]  # Removal of initial silence
audio = audio / np.max(np.abs(audio))   # Normalization of amplitude
```

Fig. 3 -> Pré-Processamento dos áudios.

 # Cálculo das Características Temporais

Em seguida, calculei diferentes características temporais nos sinais de áudio. Algumas das características consideradas incluem energia total, amplitude máxima, amplitude mínima, razão de amplitudes e desvio padrão da amplitude. Essas características são calculadas para cada sinal de áudio e são usadas para identificar padrões que possam distinguir os diferentes dígitos.

```
# Add or remove silence to make them all the same time
duration = 3  # Seconds
if len(audio) < samplingRate * duration:
    audio = np.pad(audio, (0, samplingRate * duration - len(audio)), mode='constant')
elif len(audio) > samplingRate * duration:
    audio = audio[:samplingRate * duration]

#
totalEnergy.append(np.sum(audio ** 2))
maxAmplitude.append(np.max(audio))
minAmplitude.append(np.min(audio))
rAmplitudes.append(np.max(audio) / np.abs(np.min(audio)))
standartDeviation.append(np.std(audio))
```

Fig. 4 -> Adição ou remoção de silêncio e calculo das características temporais.

 # Análise das Características Temporais

Para analisar as características temporais, utilizamos representações gráficas como boxplots. Esses gráficos permitem visualizar a distribuição das características para cada áudio e identificar quais características proporcionam uma melhor discriminação entre os mesmos.

 # Conclusão

Devido aos erros que passei durante o processo de realização, sei que a entrega está aquém do esperado. Resta melhorar e entregar algo decente na meta 2.

Carlos Soares.

Nº 2020230124

ATD – 2023/2024 – PL4