Nombre y Apellido: Nehuen Parra
Curso: 5to 2da
Fecha de entrega: 12/12/2024

T.P Configurador

1. Explicar el funcionamiento del código.
2. Sacar a relucir los rasgos asociados a los distintos paradigmas de programaci´on en el
código.
3. Explicar la función de cada script y cómo interactúan entre sí.
4. Indicar los criterios y decisiones tomadas para el planteamiento del código.
5. Explicar las fortalezas y debilidades del software diseñado

---

1. Explicación del funcionamiento del código

El programa analiza la frecuencia base de un conjunto de archivos de audio (masculinos y femeninos) para calcular estadísticas como la media y desviación estándar. Estas estadísticas se almacenan en un archivo JSON para su posterior uso en tareas de clasificación de voces.

Flujo general:

Define dos listas de archivos de audio: masculinos y femeninos.

Agrega muestras adicionales a la lista femenina.

Procesa cada archivo de audio para calcular su frecuencia base usando un objeto de la clase `Audio`.

Calcula las estadísticas (media y desviación estándar) de las frecuencias base para cada categoría.

Guarda los resultados en formato JSON en un archivo llamado `configuracionVoces.txt`.

2. Rasgos asociados a los paradigmas de programación

Paradigma procedimental:

Uso de bucles `for` para iterar sobre las listas de archivos de audio.

Estructura secuencial para el procesamiento de datos y almacenamiento.

Paradigma orientado a objetos:

La clase `Audio` encapsula la funcionalidad relacionada con la manipulación de archivos de audio, como la obtención de la frecuencia base.

Paradigma funcional:

Uso de funciones específicas como `np.mean`, `np.std`, y la inicialización de arrays con `np.zeros` para realizar cálculos y transformar datos.

3. Funciones principales y su interacción

`Audio`:

Proporciona métodos para cargar y analizar los archivos de audio.

Interactúa con el resto del código para calcular la frecuencia base de cada archivo.

Bloques `for`:

Procesan las listas de audios masculinos y femeninos para obtener estadísticas de cada conjunto.

Invocan métodos de la clase `Audio` en cada iteración.

Serialización JSON:

Se utiliza para guardar las estadísticas calculadas en un formato legible y reutilizable.

4. Criterios y decisiones tomadas en el planteamiento

Modularidad:

La clase `Audio` abstrae el manejo de datos de audio, lo que facilita su reutilización en otros contextos.

Formato de salida (JSON):

Se eligió JSON por ser un formato ampliamente usado, fácil de leer y manipular.

Uso de estadísticas básicas:

La media y desviación estándar son criterios útiles y simples para modelar la distribución de las frecuencias base de las voces.

5. Fortalezas y debilidades del software diseñado

Fortalezas:

Modularidad y claridad en la estructura del código.

Uso de una clase para encapsular la funcionalidad relacionada con el audio.

Exportación de datos en un formato estándar (JSON) para su reutilización.

Debilidades:

Falta de manejo de errores:

No hay validación de archivos o manejo de excepciones para entradas no válidas o archivos corruptos.

Escalabilidad limitada:

El enfoque actual puede no ser eficiente para grandes conjuntos de datos debido a la falta de paralelización.

Dependencia de valores arbitrarios:

La variable `orden` y los archivos predeterminados se establecen sin justificación clara.