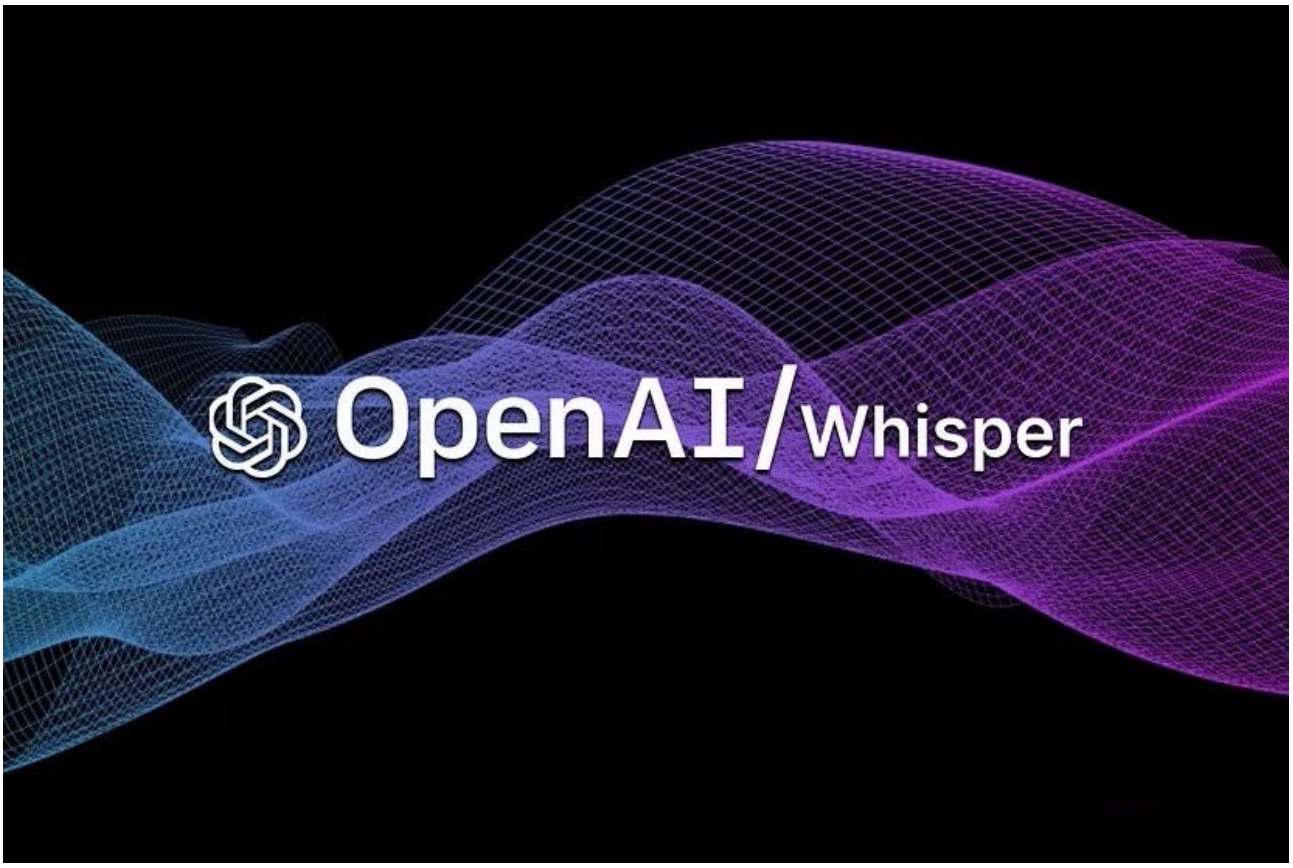
TAREA 7 (24-25) WHISPER (Hugging Face)



Link al cuaderno en github: Github Repo

ÍNDICE

índice

Importación de librerías y selección de modelo

Importamos todas las librerías que vamos a usar para esta tarea:

```
import whisper
#  INSTALAR WHISPER CON EL SIGUIENTE COMANDO:
# pip install git+https://github.com/openai/whisper.git

import os
import numpy as np
import librosa
import torch
from pydub import AudioSegment
from transformers import WhisperProcessor, WhisperForConditionalGeneration
```

Es importante añadir que para instalar la libreía “whisper”, hay que usar un comando distinto del habitual. Esto me confundió un poco, pero es necesario. De lo contrario, obtendremos un error diciendo que nos falta un archivo.

Seleccionamos el modelo que usaremos y el audio que queremos transcribir:

```
    # Seleccionamos el modelo que queremos utilizar
model_name= ["openai/whisper-tiny", "openai/whisper-base",
             "openai/whisper-small", "openai/whisper-medium",
             "openai/whisper-large", "openai/whisper-large-v2"][3]

    # Especificamos el archivo de audio que queremos utilizar
audio_path = "ytmp3free.cc_the-cure-friday-im-in-love-youtubemp3free.org.mp3"
```

Exportación y carga del modelo

Utilizamos esta función para exportar el modelo seleccionado

```
def export_model(model_name: str):
    """
        Función que exporta el modelo y el procesador de Whisper a un directorio
        Args:
            model_name: str, nombre del modelo a exportar
    """
    # Cargar el procesador desde el modelo preentrenado especificado
    processor = WhisperProcessor.from_pretrained(model_name)
    # Cargar el modelo desde el modelo preentrenado especificado
    model = WhisperForConditionalGeneration.from_pretrained(model_name)
    # Guardar el modelo en el directorio especificado por model_name
    model.save_pretrained(model_name)
    # Guardar el procesador en el mismo directorio
    processor.save_pretrained(model_name)
    return model, processor

# Verificar si el directorio del modelo no existe
if not os.path.exists(model_name):
    # Exportar el modelo y el procesador si el directorio no existe
    model, processor = export_model(model_name)
```

Y usaremos esta otra función para importar y cargar el modelo:

```python
def load_model(model_name: str):
    """
        Función que carga el modelo y el procesador de Whisper desde un directorio
        Args:
            model_name: str, nombre del directorio donde se encuentra el modelo y el procesador
    """
    # Cargar el procesador desde el directorio especificado
    processor = WhisperProcessor.from_pretrained(model_name)
    # Cargar el modelo desde el directorio especificado
    model = WhisperForConditionalGeneration.from_pretrained(model_name)
    return model, processor

# Cargar el modelo y el procesador utilizando el nombre del modelo especificado
model, processor = load_model("./" + model_name)
```

Segmentación del audio

Como el modelo por defecto no es capaz de transcribir audios de más de 30 segundos, decidí dividir el audio en varios segmentos en formato wav y pasarlos uno tras otro al modelo.

```python
def segment_audio(audio_path: str, segment_duration_ms: int = 30000):
    """
        Función generador que segmenta un audio en segmentos de duración segment_duration_ms y los exporta a archivos .wav
        Args:
            audio_path: str, ruta al audio a segmentar
            segment_duration_ms: int, duración de los segmentos en milisegundos
    """

    # Cargar el archivo de audio
    audio = AudioSegment.from_file(audio_path)
    # Obtener la duración del audio en milisegundos
    duration_ms = len(audio)

    # Iterar sobre el audio en pasos de segment_duration_ms
    for start_ms in range(0, duration_ms, segment_duration_ms):
        # Calcular el final del segmento
        end_ms = min(start_ms + segment_duration_ms, duration_ms)
        # Extraer el segmento del audio
        segment = audio[start_ms:end_ms]
        # Definir el nombre del archivo del segmento
        segment_path = f"segment_{start_ms // segment_duration_ms}.wav"
        # Exportar el segmento a un archivo .wav
        segment.export(segment_path, format="wav")
        # Devolver la ruta del archivo del segmento
        yield segment_path
```

(También tengo entendido que hay una opción que quita el máximo de los 30 segundos, pero yo no la encontré. Esta es otra solución).

Transcripción del audio

Usamos esta función para transcribir todos los fragmentos del audio y volver a juntarlos

```python
def transcribe_long_audio(model: WhisperForConditionalGeneration, processor: WhisperProcessor, audio_path: str):
    """
        Función que transcribe un audio largo dividiéndolo en segmentos de 30 segundos
        Args:
            model: WhisperForConditionalGeneration, modelo de Whisper
            processor: WhisperProcessor, procesador de Whisper
            audio_path: str, ruta al audio
    """

    # Determinar si se utilizará GPU o CPU
    device = "cuda" if torch.cuda.is_available() else "cpu"
    model.to(device)

    full_transcription = ""  # Variable para almacenar la transcripción completa

    # Segmentar el audio y transcribir cada segmento
    for segment_path in segment_audio(audio_path):
        # Cargar el segmento de audio
        audio_data, _ = librosa.load(segment_path, sr=16000)
        # Procesar el audio para obtener las características de entrada
        input_features = processor(audio_data, return_tensors="pt", sampling_rate=16000).input_features
        input_features = input_features.to(device)
        # Generar la transcripción del segmento
        generated_ids = model.generate(input_features)
        transcription = processor.batch_decode(generated_ids, skip_special_tokens=True)[0]
        # Agregar la transcripción del segmento a la transcripción completa
        full_transcription += transcription + "\n"
        # Eliminar el archivo de segmento de audio
        os.remove(segment_path)

    return full_transcription.strip()  # Devolver la transcripción completa sin espacios en blanco al inicio y al final
```

Ejecución del código

Por último, ejecutamos el código

```python
# Ejecutamos el código
if __name__ == "__main__":
    transcription = transcribe_long_audio(model, processor, audio_path)
    print("Transcripción completa:")
    print(transcription)
```

✓ 1m 15.7s

```
Transcripción completa:
Stand by everyone and...CUT!
 I don't care if Monday's blue Tuesday's gray and Wednesday too Thursday, I don't care about you It's Friday, I'm in love Monday
 Saturday way Sunday always comes to me Friday never has it end I don't care if Monday's black Tuesday, Wednesday, heart attack
 Tuesday Wednesday
 See your shoes
 It's Friday, I'm in love! I don't care if Monday's blue Tuesday's gray and Wednesday too Thursday, I don't care about you It's
 Oh
 you
```

Este resultado lo obtuve usando el modelo “openai/whisper-medium”.