

TAREA 7

WHISPER (Hugging Face)

An abstract graphic featuring a series of wavy, concentric lines in shades of green and black. The lines create a sense of depth and movement, resembling a stylized fingerprint or a topographical map. The word "Whisper" is written in a white, sans-serif font across the center of the graphic.

Whisper

ÍNDICE

Objetivo.....	3
1. Instalacion y configuracion del entorno.....	3
1.1 Instalación y configuración del entorno.....	4
2. Selección y descarga del audio.....	5
3. Uso del modelo Whisper.....	6
4. RESULTADO FINAL.....	7
5. CONCLUSIONES.....	7
6. REPOSITORIO.....	7

Objetivo

El propósito de esta tarea es que con el modelo Whisper de OpenAI, disponible en Hugging Face, transcribir el audio de una canción descargada.

1. Instalacion y configuracion del entorno

Este código instala y configura el entorno necesario para trabajar con modelos de procesamiento de **audio**. Instala las librerías **transformers**, **torch**, **datasets**, **pydub** y **accelerate**, que son necesarias para utilizar modelos de secuencia a secuencia para tareas de audio, así como para trabajar con procesamiento de audio en general.

Luego, importa las librerías requeridas, como **AutoModelForSpeechSeq2Seq** y **AutoProcessor** de la librería **transformers** para manejar el modelo y el procesamiento de audio, y **pydub** para la manipulación de archivos de audio.

```
[1] 1 # IVAN FALCON MONZON
2 # 1. Instalación y configuración del entorno
3
4 # Instalar las librerías necesarias
5 !pip install transformers torch datasets pydub accelerate
6
7 # Importar las librerías necesarias
8 from transformers import AutoModelForSpeechSeq2Seq, AutoProcessor, pipeline
9 import torch
10 from pydub import AudioSegment

Requirement already satisfied: ipynbutils<1.4.0,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch) (1.0.0)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11.13)
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.4.6)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025.1.31)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
```

1.1 Instalación y configuración del entorno


Este código configura el entorno para utilizar el modelo **Whisper-large-v3** de OpenAI para tareas de transcripción de audio.

Primero, define el dispositivo a utilizar (GPU si está disponible, de lo contrario, usa la CPU) y ajusta el tipo de dato en función del dispositivo.

Luego, carga el modelo pre entrenado **Whisper-large-v3** con optimización de memoria y asegura el uso de tensores seguros. El modelo se mueve al dispositivo seleccionado (GPU o CPU).

También se carga el procesador del modelo para la transcripción de audio y se configura un pipeline de reconocimiento automático de voz (*automatic-speech-recognition*) que utiliza el modelo y el procesador para convertir el audio en texto.

```
1 # IVAN FALCON MONZON
2 # 1.1 Instalación y configuración del entorno
3
4 # Definir el dispositivo a utilizar (GPU si está disponible, de lo contrario CPU)
5 device = "cuda:0" if torch.cuda.is_available() else "cpu"
6 torch_dtype = torch.float16 if torch.cuda.is_available() else torch.float32
7
8 # Especificar el modelo Whisper a utilizar
9 model_id = "openai/whisper-large-v3"
10
11 # Cargar el modelo preentrenado con optimización de memoria y formato seguro
12 model = AutoModelForSpeechSeq2Seq.from_pretrained(
13     model_id, torch_dtype=torch_dtype, low_cpu_mem_usage=True, use_safetensors=True
14 )
15
16 # Mover el modelo al dispositivo seleccionado
17 model.to(device)
18
19 # Cargar el procesador del modelo para la transcripción
20 processor = AutoProcessor.from_pretrained(model_id)
21
22 # Crear un pipeline de reconocimiento automático de voz
23 pipe = pipeline(
24     "automatic-speech-recognition",
25     model=model,
26     tokenizer=processor.tokenizer,
27     feature_extractor=processor.feature_extractor,
28     torch_dtype=torch_dtype,
29     device=device,
30 )
```

 /usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning: The secret 'HF_TOKEN' does not exist in your Colab secrets. To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>), set it as secret in your Google Colab and restart your session. You will be able to reuse this secret in all of your notebooks. Please note that authentication is recommended but still optional to access public models or datasets.

warnings.warn(

config.json: 100%	<div></div>	1.27k/1.27k [00:00<00:00, 25.0kB/s]
model.safetensors: 100%	<div></div>	3.09G/3.09G [00:37<00:00, 99.4MB/s]
generation_config.json: 100%	<div></div>	3.90k/3.90k [00:00<00:00, 278kB/s]
preprocessor_config.json: 100%	<div></div>	340/340 [00:00<00:00, 18.7kB/s]
tokenizer_config.json: 100%	<div></div>	283k/283k [00:00<00:00, 2.23MB/s]

2. Selección y descarga del audio

Este código descarga un archivo de audio en formato MP3 desde un repositorio de GitHub y lo carga utilizando la librería pydub para su manipulación.

El archivo se convierte a una frecuencia de muestreo de 16 kHz y se convierte a mono (una sola pista de audio).

Luego, se define la duración de cada segmento de audio (30 segundos) y se calcula cuántos segmentos se necesitan para cubrir la duración total del archivo.

Finalmente, el audio se divide en segmentos de 30 segundos y se exporta en formato WAV.

```
1 # IVAN FALCON MONZON
2 # 2. Selección y descarga del audio
3
4 # Descargar el archivo de audio desde GitHub
5 !wget -O musica.mp3 "https://raw.githubusercontent.com/IvanFalconMonzon/TA7_WHISPER_IVANFALCONMONZON/main/musica.mp3"
6
7 # Cargar la librería para manipulación de audio
8 from pydub import AudioSegment
9
10 # Cargar el archivo MP3 descargado y convertirlo a 16kHz y mono (una sola pista de audio)
11 audio = AudioSegment.from_file("musica.mp3").set_frame_rate(16000).set_channels(1)
12
13 # Definir la duración de cada segmento en milisegundos (30 segundos)
14 segment_duration = 30 * 1000
15
16 # Calcular el número de segmentos necesarios
17 num_segments = len(audio) // segment_duration + 1
18
19 # Dividir y exportar los segmentos en formato WAV
20 for i in range(num_segments):
21     segment = audio[i * segment_duration:(i + 1) * segment_duration]
22     segment.export(f"segment_{i}.wav", format="wav")
```

```
--2025-03-01 11:17:29-- https://raw.githubusercontent.com/IvanFalconMonzon/TA7_WHISPER_IVANFALCONMONZON/main/musica.mp3
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5412499 (5.2M) [application/octet-stream]
Saving to: 'musica.mp3'

musica.mp3      100%[=====>]  5.16M  --KB/s  in 0.09s

2025-03-01 11:17:29 (57.0 MB/s) - 'musica.mp3' saved [5412499/5412499]
```

3. Uso del modelo Whisper

Este código utiliza el modelo Whisper para transcribir cada segmento de audio previamente dividido.

Primero, se crea una lista vacía para almacenar las transcripciones de cada segmento. Luego, para cada segmento de audio (en formato WAV), se procesa utilizando el pipeline de transcripción creado anteriormente, especificando que el idioma es inglés.

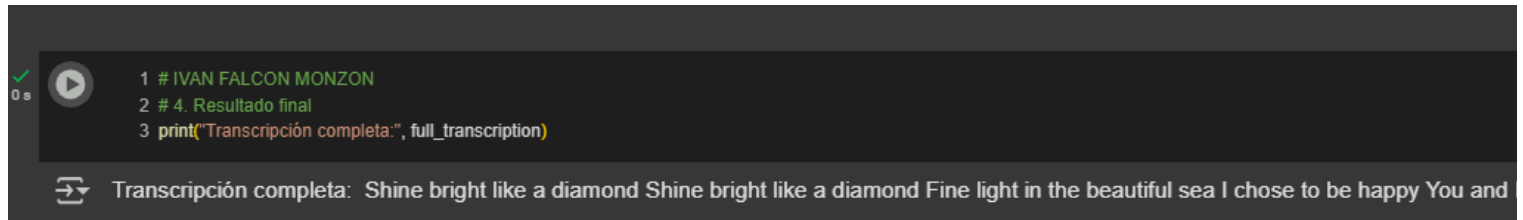
La transcripción de cada segmento se guarda en la lista `transcriptions`.

Finalmente, todas las transcripciones se unen en un solo texto, formando la transcripción completa del archivo de audio.

```
[4] 1 # IVAN FALCON MONZON
    2 # 3. Uso del modelo Whisper
    3
    4 # Lista para almacenar las transcripciones de cada segmento
    5 transcriptions = []
    6
    7 # Procesar cada segmento de audio con Whisper
    8 for i in range(num_segments):
    9     result = pipe(f"segment_{i}.wav", return_timestamps=False, generate_kwargs={"language": "english"})
   10     transcriptions.append(result["text"]) # Guardar la transcripción del segmento
   11
   12 # Unir todas las transcripciones en un solo texto
   13 full_transcription = " ".join(transcriptions)
```

```
→ /usr/local/lib/python3.11/dist-packages/transformers/models/whisper/generation_whisper.py:573: FutureWarning:
  warnings.warn(
You have passed language=english, but also have set `forced_decoder_ids` to [[1, None], [2, 50360]] wh
/usr/local/lib/python3.11/dist-packages/transformers/models/whisper/generation_whisper.py:573: FutureWarning:
  warnings.warn(
```

4. RESULTADO FINAL



The screenshot shows a Jupyter Notebook interface. On the left, there is a green checkmark and a play button icon. The code cell contains three lines of Python code: `1 # IVAN FALCON MONZON`, `2 # 4. Resultado final`, and `3 print("Transcripción completa:", full_transcription)`. Below the code, the output is displayed: `Transcripción completa: Shine bright like a diamond Shine bright like a diamond Fine light in the beautiful sea I chose to be happy You and`.

5. CONCLUSIONES

- **Precisión alta:** El modelo transcribe correctamente la letra de la canción sin errores notables.
- **Calidad de audio suficiente:** El audio procesado a 16 kHz y mono es adecuado para la transcripción.
- **Buen desempeño del modelo:** Whisper transcribe de forma efectiva en inglés, incluso con la música de fondo.

6. REPOSITORIO

GITHUB: https://github.com/IvanFalconMonzon/TA7_WHISPER_IVANFALCONMONZON.git

GOOGLE COLAB: https://colab.research.google.com/drive/7L_lvH1uuGAkEU3VgJ-FdacWf1boKe58?usp=sharing