# Instituto Tecnológico y de Estudios Superiores de Monterrey

**TC3007C.503**
Inteligencia Artificial Avanzada para la Ciencia de Datos II

**Escuela de Ingeniería y Ciencias**
Ingeniería en Ciencias de Datos y Matemáticas

## Proyecto Final NLP

**Profesor**
Dr. Juan Arturo Nolazco

**Equipo**
Equipo 2

**Alumno**

Alejandro José Murcia Alfaro A00828513

Monterrey, Nuevo León. 01 de Diciembre de 2023

# 1. Objetivo del Proyecto y Comentarios

Este proyecto se centró en el desarrollo de una interfaz de usuario implementada a través de Streamlit, diseñada específicamente para interactuar con APIs avanzadas de lenguaje natural y reconocimiento de Voz. El desafío principal era construir una herramienta que pudiera leer archivos de voz, transcribirlos y, a partir de esa transcripción, generar un resumen conciso y significativo utilizando técnicas de Procesamiento de Lenguaje Natural (NLP).

La interfaz desarrollada actúa como un puente entre el usuario y las complejas tecnologías de NLP y reconocimiento de voz. Utilizando la biblioteca Streamlit, se logró crear una interfaz intuitiva y amigable que permite a los usuarios cargar archivos de audio de manera sencilla. Este enfoque práctico y centrado en el usuario subraya la importancia de una buena ingeniería de software y diseño de UX, enfocándose no solo en la funcionalidad, sino también en la experiencia del usuario final. La interfaz fue diseñada para ser lo suficientemente flexible como para manejar diferentes formatos de audio y responder de manera eficiente a las entradas del usuario.

En el corazón de este proyecto se encuentra la tecnología de Whisper, una herramienta de reconocimiento de voz desarrollada por OpenAI. Whisper se caracteriza por su capacidad para transcribir voz a texto con una precisión alta, incluso en situaciones de audio desafiantes, como ruido de fondo o acentos variados. Esto se logra a través de modelos de aprendizaje profundo entrenados en un vasto conjunto de datos multilingües, permitiendo que Whisper maneje una gama diversa de idiomas y dialectos con una eficiencia notable. En paralelo, para el procesamiento de lenguaje natural y generación de resúmenes, se utilizó la API de GPT-3.5-Turbo de OpenAI. Esta API representa uno de los modelos de lenguaje más avanzados disponibles, destacando por su habilidad para entender y generar texto de manera coherente y contextual. GPT-3.5-Turbo, basado en arquitecturas de redes neuronales transformadoras, puede realizar una amplia gama de tareas de NLP, desde generar texto creativo hasta ofrecer resúmenes concisos y precisos. La combinación de Whisper y GPT-3.5-Turbo ilustra el poder de la inteligencia artificial moderna y su aplicabilidad en soluciones prácticas de NLP, abriendo un abanico de posibilidades para futuras innovaciones en el campo.

Para la transcripción del audio, se seleccionó la API de Whisper de OpenAI, destacando por su alta eficiencia en la conversión de voz a texto. La precisión y velocidad de esta API jugaron un papel crucial en el procesamiento eficiente de archivos de audio, mostrando un rendimiento notable incluso en condiciones de audio desafiantes. Posteriormente, para resumir el texto transcrito, se integró la API de GPT-3.5-Turbo de OpenAI. Esta API no solo proporcionó resúmenes coherentes y concisos, sino que también exhibió su versatilidad en otras áreas de NLP como clasificación, análisis de sentimientos y traducción. Sin embargo, el proyecto se enfocó principalmente en la generación de resúmenes, dejando otras funcionalidades como posibles áreas para futuras exploraciones.

A lo largo de este proyecto, he ganado una apreciación más profunda del poder y la versatilidad de las APIs de NLP y reconocimiento de voz. La integración de estas tecnologías en una interfaz de usuario amigable ha sido un desafío gratificante. He aprendido no solo sobre el desarrollo técnico, sino también sobre la importancia de crear interfaces que faciliten la interacción entre la tecnología avanzada y los usuarios finales.

Algo que me ha generado mucho valor y me servirá muchísimo para mi carrera y desarrollo profesional es llevar los scripts de Python fuera de la esfera de ejecución en consola o solo dejarlos en la etapa de desarrollo y prueba. Streamlit me permite darle una nueva vida a mis proyectos, donde un usuario de forma intuitiva y sencilla puede hacer uso de las funcionalidades que he programado en Python, pero de manera sencilla y que las personas puedan visualizar fácilmente. En vez de cargar un archivo de audio en consola como se haría originalmente, puedo desarrollar una UI y sitio web donde el usuario puede subir su archivo y procesarlo todo a través de botones de manera intuitiva.

## 2. Ejecución


Figura 1: Interfaz de Streamlit Antes de Usarse

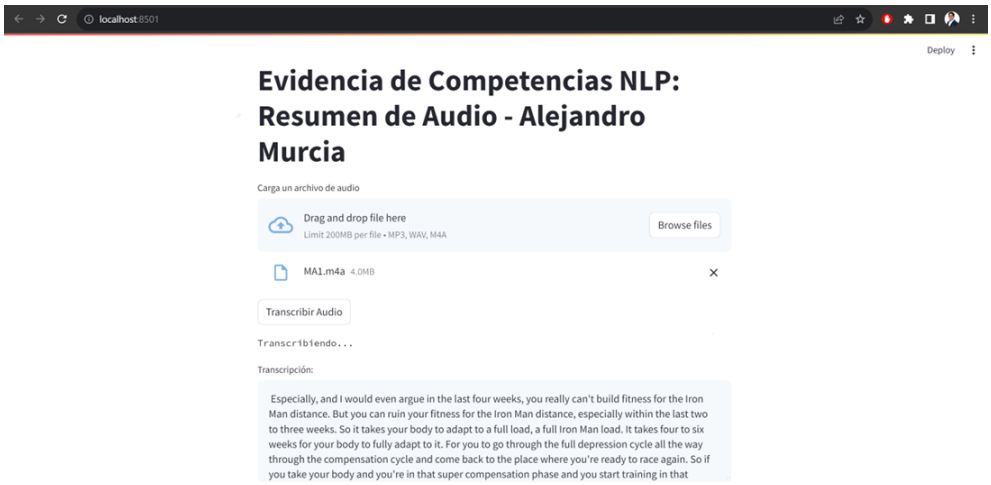


Figura 2: Interfaz de Streamlit Durante Transcripción

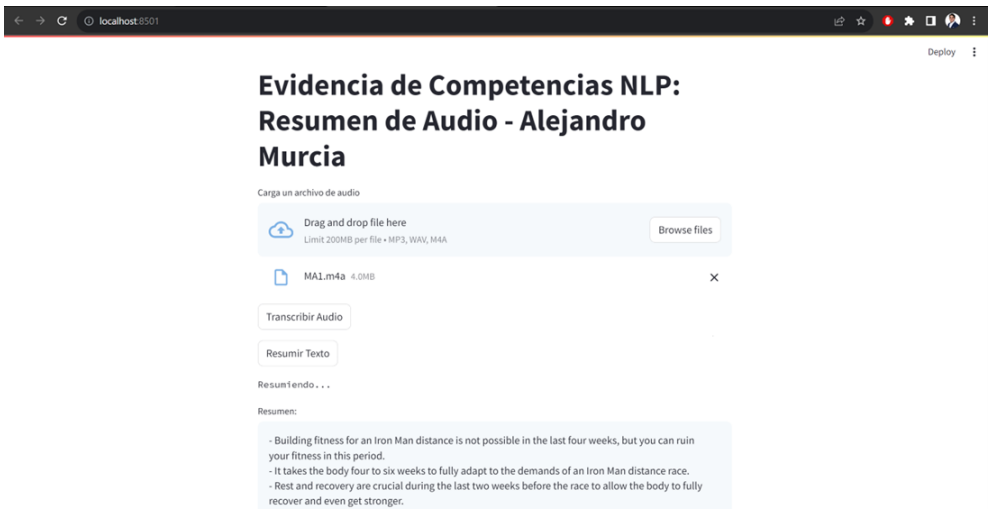


Figura 3: Interfaz de Streamlit Durante Resumen

## 3. Anexo con el Código

```python
import streamlit as st
import openai
import whisper
import os

# Clave API de OpenAI
openai.api_key = 'mi-llave'
model = whisper.load_model("base")

def transcribe_audio(model, audio_file):
    # Guardar el archivo de audio temporalmente
    temp_file_path = "temp_audio." + audio_file.name.split('.')[-1]
    with open(temp_file_path, "wb") as f:
        f.write(audio_file.getbuffer())

    # Transcribir el audio
    transcript = model.transcribe(temp_file_path)

    # Eliminar el archivo temporal
    os.remove(temp_file_path)

    return transcript['text']

def CustomChatGPT(user_input):
    messages = [{"role": "system", "content": "You are an office administrator, summarize the text in key points"}]
    messages.append({"role": "user", "content": user_input})
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=messages
    )
    ChatGPT_reply = response["choices"][0]["message"]["content"]
    return ChatGPT_reply

# Interfaz de usuario de Streamlit
st.title('Evidencia de Competencias NLP: Resumen de Audio - Alejandro Murcia')

uploaded_file = st.file_uploader("Carga un archivo de audio", type=['mp3', 'wav', 'm4a'])

if uploaded_file is not None:
    if st.button('Transcribir Audio'):
        st.text("Transcribiendo...")
        transcription = transcribe_audio(model, uploaded_file)
        st.text_area("Transcripcion:", transcription, height=150)
        st.session_state['transcription'] = transcription

    if st.button('Resumir Texto'):
        if 'transcription' in st.session_state and st.session_state['transcription']:
            st.text("Resumiendo...")
            summary = CustomChatGPT(st.session_state['transcription'])
            st.text_area("Resumen:", summary, height=150)
        else:
            st.warning("Primero transcribe un audio para resumir.")
```

Listing 1: Código de Streamlit para Transcripción y Resumen