Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey

Entregable NLP

TC3007C.501 Inteligencia Artificial Avanzada para la Ciencia de Datos II

Profesores:

Iván Mauricio Amaya Contreras

Blanca Rosa Ruiz Hernández

Félix Ricardo Botello Urrutia

Edgar Covantes Osuna

Felipe Castillo Rendón

Hugo Terashima Marín

Integrantes:

Julian Lawrence Gil Soares - A00832272

24 de Noviembre del 2023

Reto:
Utilizando el servicio de transcripción de audio a texto de openAL crear una aplicación web en la que se puede enviar un video con formato mp4 con la api de openAI para transcribir el audio y luego desplegar el texto transcrito en la aplicación web.

Implementación:

```
import streamlit as st
import ssl
import certifi
import openai
import whisper
import tempfile
import os

ssl._create_default_https_context = lambda:
ssl.create_default_context(cafile=certifi.where())

openai.api_key = ""

def load_whisper_model():
    try:
        model = whisper.load_model("base")
        return model
    except Exception as e:
        st.error(f"Error: {e}")
        return None

def transcribe_audio(model, file_path):
    try:
        transcript = model.transcribe(file_path)
        return transcript['text']
    except Exception as e:
        st.error(f"Error: {e}")
        return ""

def main():
    st.set_page_config(
        page_title="Speech to text transcription",
        page_icon="🎤"
    )
```

```
    model = load_whisper_model()

    st.title("Speech to text transcription")

    st.sidebar.markdown(
        """
           This app takes an mp4 file and using openAI speech to text
transcribes the audio of the file to text
        """
    )

    if model:
            uploaded_file = st.file_uploader("Upload a video file for
transcription", type=["mp4", "wav"])

        if uploaded_file is not None:
                    with tempfile.NamedTemporaryFile(delete=False) as
temp_file:
                temp_file.write(uploaded_file.read())
                temp_file_path = temp_file.name

            st.subheader("Transcription:")
            transcription = transcribe_audio(model, temp_file_path)
            st.code(transcription, language="plaintext")


            os.unlink(temp_file_path)

if __name__ == "__main__":
    main()
```

En mi solución, elegí Streamlit como el framework para desarrollar la interfaz web de la aplicación. Streamlit es una opción ideal porque proporciona una plataforma fácil de usar y requiere de muy poca configuración, permitiéndome centrarme en la lógica de la aplicación en lugar de la configuración del framework. Streamlit me permite crear una interfaz de usuario interactiva con sólo unas pocas líneas de código. Puedo diseñar rápidamente la estructura de la aplicación, incluyendo barras laterales informativas y áreas para la visualización de resultados.
Incorporé la biblioteca Whisper en mi aplicación para realizar el reconocimiento de audio. Whisper es una biblioteca desarrollada por OpenAI específicamente para la transcripción automática de la voz a texto. Al cargar el modelo Whisper en mi aplicación, obtengo la capacidad de transcribir archivos de audio, convirtiendo así las grabaciones habladas en texto escrito. Además, utilicé la biblioteca de OpenAI

para acceder a sus APIs. En este caso, no estoy utilizando la API de OpenAI directamente para el reconocimiento de audio, ya que eso se maneja a través de Whisper. En cambio, la API de OpenAI es crucial para interactuar con otros modelos, como Chat GPT, para realizar tareas específicas, como resumir el texto transcribir en esta aplicación.