Instituto Tecnológico y de Estudios Superiores de Monterrey

Evidencia NLP. Aplicación Web Speech-to-Summary con uso de API

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

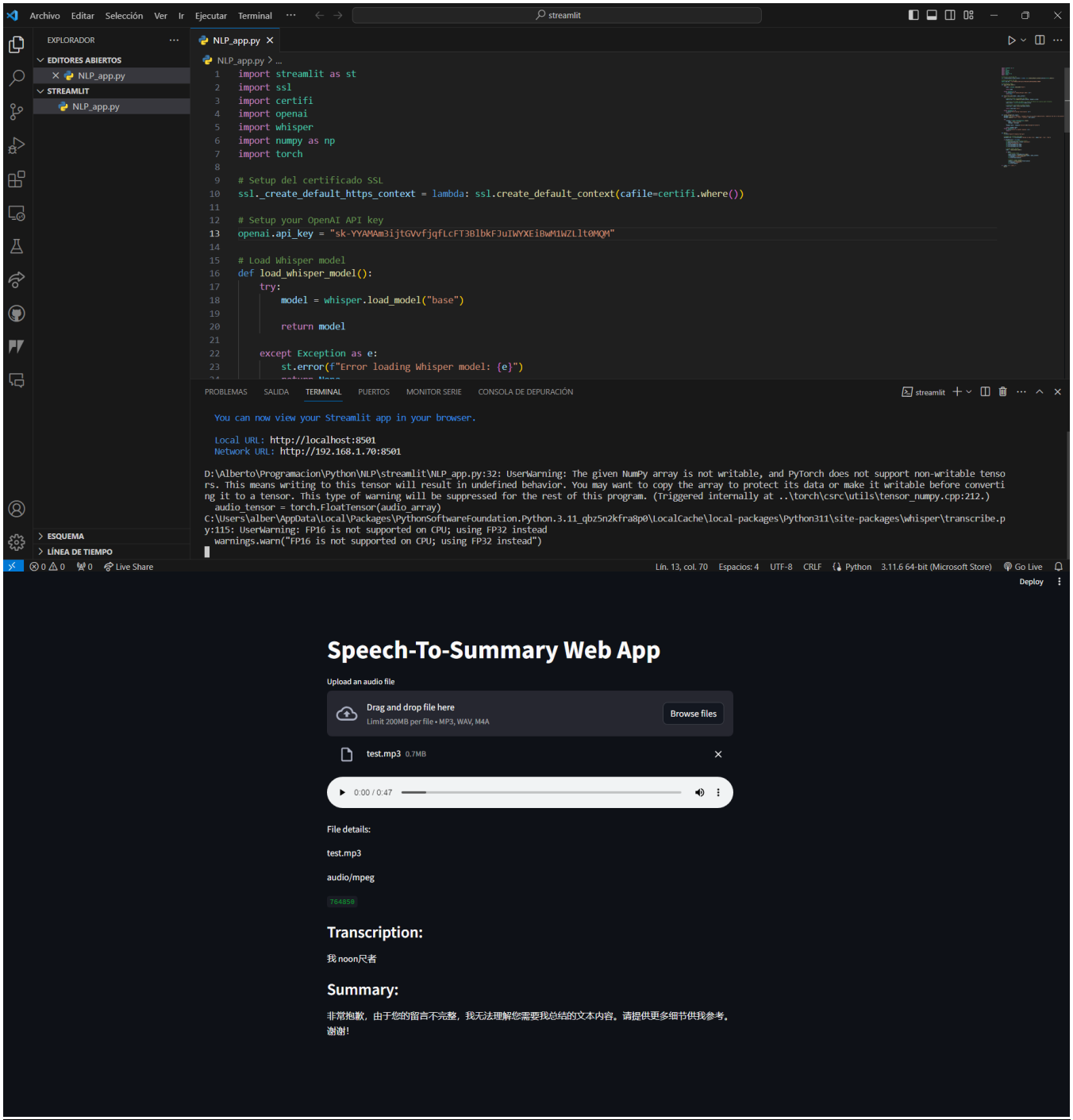
Hugo Terashima Marín

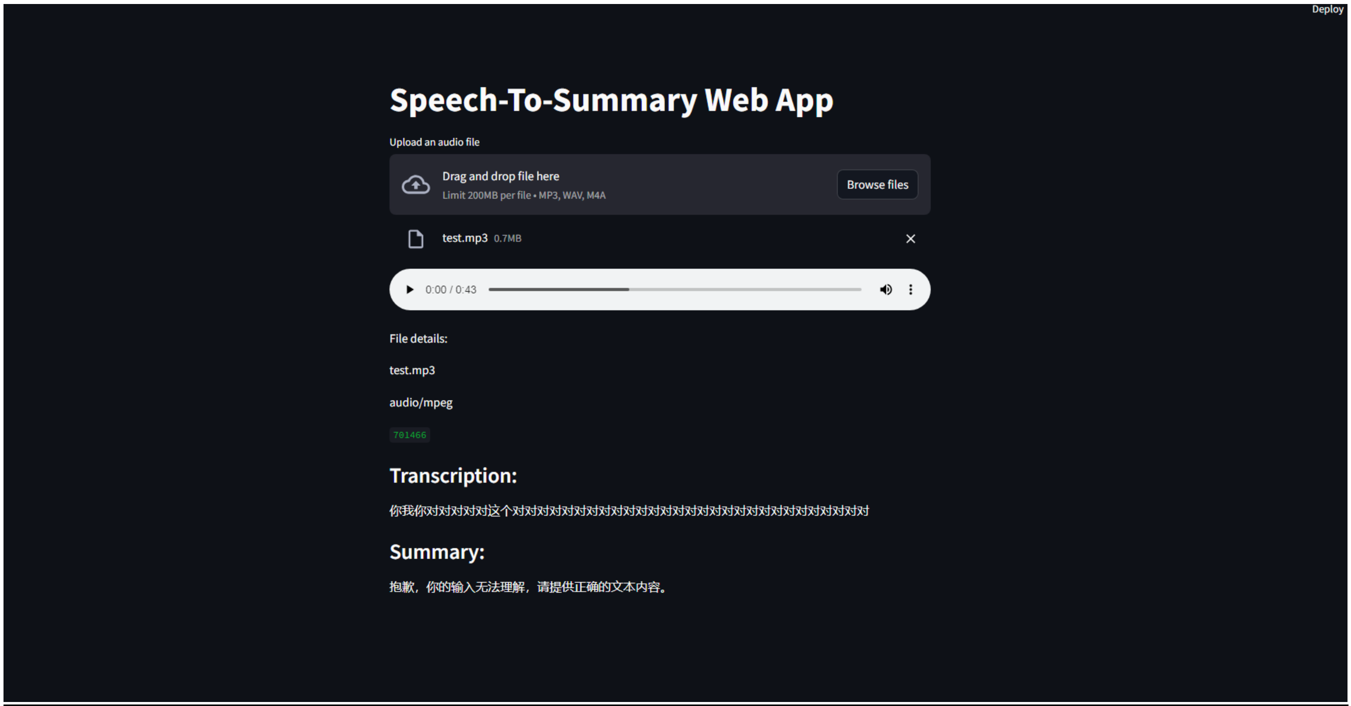
Alumno:

Alberto H Orozco Ramos - A00831719

24 de Noviembre de 2023

La evidencia correspondiente al modulo de NLP consiste en la implementación de un modelo de lenguaje de cualquier tipo dentro de una interfaz web que hace uso de la API de OpenAI para generar textos en base a los datos de entrada recibidos, ya sean texto directo o audio, generando salidas como resúmenes, traducciones o interpretaciones de los datos de entrada. En mi caso, adapté uno de los códigos utilizados en clase para alojarlo en una aplicación web de StreamLit como podemos ver en las siguientes imágenes:





Importamos las librerías necesarias:

```
import streamlit as st
import ssl
import certifi
import openai
import whisper
import numpy as np
import torch
```

Establecer el context SSL utilizando la librería “certifi”, la cual provee una colección curada de certificados raíz. También se coloca la llave a utilizar de la cuenta de OpenAI respectiva:

```
# Setup del certificado SSL
ssl._create_default_https_context = lambda:
ssl.create_default_context(cafile=certifi.where())

# Setup your OpenAI API key
openai.api_key = ""
```

Carga del modelo “Whisper”:

```
# Load Whisper model
def load_whisper_model():
    try:
        model = whisper.load_model("base")

        return model
```

```
    except Exception as e:
        st.error(f"Error loading Whisper model: {e}")
        return None
```

Función encargada de convertir el archivo de audio a un arreglo de Numpy y después a un tensor de PyTorch para su posterior procesamiento de transcripción:

```
def transcribe_audio(model, audio_content):
    try:
        # Covertir bytes a un arreglo de Numpy
        audio_array = np.frombuffer(audio_content, dtype=np.int16)

        # Convertir de arreglo de NumPy a un tensor de PyTorch con valores punto
flotantes
        audio_tensor = torch.FloatTensor(audio_array)

        # Transcribir utilizando el modelo Whisper
        transcript = model.transcribe(audio_tensor)

        return transcript['text']

    except Exception as e:
        st.error(f"Error during transcription: {e}")
        return ""
```

Llamada de un modelo de ChatGPT por medio de su API para generar una respuesta en base a la transcripción realizada por la función “transcribe_audio”:

```
def custom_chatgpt(user_input):
    messages = [{"role": "system", "content":"You are an office administrator,
summarize the text in key points"}]
    messages.append({"role": "user", "content": user_input})

    try:
        response = openai.ChatCompletion.create(
            model = "gpt-3.5-turbo",
            messages = messages
        )
        chatgpt_reply = response["choices"][0]["message"]["content"]

        return chatgpt_reply
    except Exception as e:
        st.error(f"Error in ChatGPT response: {e}")
        return ""
```

Función principal encargada de manejar la carga de archivos de audio (“mp3”, “wav”, “m4a”), cargar el modelo “Whisper”, llamar a la función de transcripción y generar la respuesta por parte de ChatGPT:

```
def main():
    st.title("Speech-To-Summary Web App")

    # Cargador de archivos para audio
    uploaded_file = st.file_uploader("Upload an audio file", type=["mp3", "wav",
"m4a"])

    if uploaded_file is not None:
        # Desplegamos los detalles del archivo
        st.audio(uploaded_file, format='audio/mp3')
        st.write("File details:")
        st.write(uploaded_file.name)
        st.write(uploaded_file.type)
        st.write(uploaded_file.size)

        # Cargar modelo Whisper
        model = load_whisper_model()

        if model:
            # Transcribir audio
            audio_content = uploaded_file.read()
            transcription = transcribe_audio(model, audio_content)
            st.subheader("Transcription:")
            st.write(transcription)

            # Resumir usando ChatGPT
            summary = custom_chatgpt(transcription)
            st.subheader("Summary:")
            st.write(summary)

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