README

----> Desde TERMINAL

#Ubicación

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/samples/

#Comando para transcribir

whisper prueba2.m4a --model base --language Spanish

----> Usando PYTHON

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/

python3 transcribe\_mult.py

----> Página web

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/

python3 app.py

MODEL: “LARGE”  
CHAT-GPT: “DAVINCI-002”

Terminal:

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/

python3 app.py

Código (Python)

“from flask import Flask, render\_template, request, send\_from\_directory

import os

import certifi

import whisper

import openai

import json

# Establecer la variable de entorno SSL\_CERT\_FILE

os.environ["SSL\_CERT\_FILE"] = certifi.where()

app = Flask(\_\_name\_\_)

# Configurar el directorio de carga

UPLOAD\_FOLDER = 'uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Cargar el modelo de Whisper

model = whisper.load\_model("large")

# Configurar la API Key de OpenAI

openai.api\_key = "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"

def improve\_transcription(text):

prompt = f"Por favor, corrige y mejora la siguiente transcripción de audio en español, agregando puntuación y uso correcto de mayúsculas: {text}"

response = openai.Completion.create(

engine="text-davinci-002",

prompt=prompt,

max\_tokens=1024,

n=1,

temperature=0.5,

)

improved\_text = response.choices[0].text.strip()

return improved\_text

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/upload', methods=['POST'])

def upload():

if request.method == 'POST':

# Obtener el archivo de audio de la solicitud

audio\_file = request.files['file']

# Guardar el archivo en el directorio de carga

audio\_file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], audio\_file.filename)

audio\_file.save(audio\_file\_path)

# Transcribir el archivo y guardar la transcripción en un archivo .txt

result = model.transcribe(audio\_file\_path, language="Spanish")

transcription\_file = os.path.splitext(audio\_file.filename)[0] + ".txt"

transcription\_path = os.path.join(app.config['UPLOAD\_FOLDER'], transcription\_file)

with open(transcription\_path, "w") as f:

improved\_text = improve\_transcription(result["text"])

f.write(improved\_text)

# Eliminar el archivo de audio original

os.remove(audio\_file\_path)

return send\_from\_directory(app.config['UPLOAD\_FOLDER'], transcription\_file, as\_attachment=True)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)”

MODEL: “LARGE”  
CHAT-GPT: “GPT-4”

Temperatura: 1

Terminal:

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/

python3 app.py

**“APP.PY”**

from flask import Flask, render\_template, request, send\_from\_directory

import os

import certifi

import whisper

import openai

import json

import requests

# Establecer la variable de entorno SSL\_CERT\_FILE

os.environ["SSL\_CERT\_FILE"] = certifi.where()

app = Flask(\_\_name\_\_)

# Configurar el directorio de carga

UPLOAD\_FOLDER = 'uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Cargar el modelo de Whisper

model = whisper.load\_model("large")

# Configurar la API Key de OpenAI

API\_KEY = "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"

API\_ENDPOINT = "https://api.openai.com/v1/chat/completions"

def generate\_chat\_completion(messages, model="gpt-4", temperature=1, max\_tokens=None):

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {API\_KEY}",

}

data = {

"model": model,

"messages": messages,

"temperature": temperature,

}

if max\_tokens is not None:

data["max\_tokens"] = max\_tokens

response = requests.post(API\_ENDPOINT, headers=headers, data=json.dumps(data))

if response.status\_code == 200:

return response.json()["choices"][0]["message"]["content"]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def improve\_transcription(text):

messages = [

{"role": "system", "content": "Eres mi asistente para la redacción de textos."},

{"role": "user", "content": f"Corrige la redacción de este texto. Cada vez que cambie el interlocutor, inicia un nuevo párrafo con un guión: {text}"}

]

improved\_text = generate\_chat\_completion(messages, model="gpt-4", temperature=1)

return improved\_text.strip()

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/', methods=['POST'])

def upload():

if request.method == 'POST':

# Obtener el archivo de audio de la solicitud

audio\_file = request.files['file']

# Guardar el archivo en el directorio de carga

audio\_file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], audio\_file.filename)

audio\_file.save(audio\_file\_path)

# Transcribir el archivo y guardar la transcripción en un archivo .txt

result = model.transcribe(audio\_file\_path, language="Spanish")

transcription\_file = os.path.splitext(audio\_file.filename)[0] + ".txt"

transcription\_path = os.path.join(app.config['UPLOAD\_FOLDER'], transcription\_file)

with open(transcription\_path, "w") as f:

improved\_text = improve\_transcription(result["text"])

f.write(improved\_text)

# Eliminar el archivo de audio original

os.remove(audio\_file\_path)

return send\_from\_directory(app.config['UPLOAD\_FOLDER'], transcription\_file, as\_attachment=True)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**“INDEX.HTML”**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Transcriptor de Audio</title>

</head>

<body>

<h1>Transcriptor de Audio</h1>

<form method="POST" enctype="multipart/form-data" action="/">

<input type="file" name="file" accept="audio/\*" required>

<button type="submit">Transcribir</button>

</form>

</body>

</html>

**“STYLES.CSS”**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

background-color: #f5f5f5;

}

form {

background-color: #fff;

padding: 2rem;

border-radius: 4px;

box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);

}

h1 {

font-size: 1.5rem;

margin-bottom: 1.5rem;

text-align: center;

}

input[type="file"] {

margin-bottom: 1.5rem;

}

button[type="submit"] {

background-color: #007bff;

color: #fff;

border: none;

padding: 0.5rem 1rem;

border-radius: 4px;

cursor: pointer;

font-size: 1rem;

}

button[type="submit"]:hover {

background-color: #0056b3;

}

MODEL: “LARGE”  
CHAT-GPT: “GPT-4”

Temperatura: 1

+

-TEXTBOX-

Terminal:

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/

python3 app.py

PLAYGROUND

System:

Eres mi asistente de programación. Yo no tengo ninguna experiencia en programación ni desarrollo de aplicaciones. Deberás guiarme paso a paso y ser muy clara en tus indicaciones. Muy importante: espera a que te confirme que he hecho un paso antes de avanzar con el siguiente. Tengo una MacOS, Python3 y uso Safari. Cuando deba hacer una modificación de un código, transcríbelo completamente para que yo solamente deba copiar y pegarlo en su integridad.

Prompt:

Vamos a trabajar en la apariencia de la interfaz de mi programa para la transcripción de audio a texto. Actualmente, tengo tres archivos ("index.html"; "app.py" y "styles.css"), cuyo contenido te transcribiré en dos mensajes a continuación.

1. "index.html"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/templates/index.html

Contenido:

2. "app.py"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/app.py

Contenido:

3. "styles.css"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/templates/static/css/styles.css

Contenido:

**“APP.PY”**

from flask import Flask, render\_template, request, send\_from\_directory

import os

import certifi

import whisper

import openai

import json

import requests

# Establecer la variable de entorno SSL\_CERT\_FILE

os.environ["SSL\_CERT\_FILE"] = certifi.where()

app = Flask(\_\_name\_\_, static\_folder="templates/static")

# Configurar el directorio de carga

UPLOAD\_FOLDER = 'uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Cargar el modelo de Whisper

model = whisper.load\_model("large")

# Configurar la API Key de OpenAI

API\_KEY = "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"

API\_ENDPOINT = "https://api.openai.com/v1/chat/completions"

def generate\_chat\_completion(messages, model="gpt-4", temperature=1, max\_tokens=None):

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {API\_KEY}",

}

data = {

"model": model,

"messages": messages,

"temperature": temperature,

}

if max\_tokens is not None:

data["max\_tokens"] = max\_tokens

response = requests.post(API\_ENDPOINT, headers=headers, data=json.dumps(data))

if response.status\_code == 200:

return response.json()["choices"][0]["message"]["content"]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def improve\_transcription(text):

messages = [

{"role": "system", "content": "Eres mi asistente para la redacción de textos."},

{"role": "user", "content": f"Corrige la redacción de este texto. Cada vez que cambie el interlocutor, inicia un nuevo párrafo con un guión: {text}"}

]

improved\_text = generate\_chat\_completion(messages, model="gpt-4", temperature=1)

return improved\_text.strip()

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/upload', methods=['POST'])

def upload():

if request.method == 'POST':

# Obtener el archivo de audio de la solicitud

audio\_file = request.files['file']

# Guardar el archivo en el directorio de carga

audio\_file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], audio\_file.filename)

audio\_file.save(audio\_file\_path)

# Transcribir el archivo y guardar la transcripción en un archivo .txt

result = model.transcribe(audio\_file\_path, language="Spanish")

transcription\_file = os.path.splitext(audio\_file.filename)[0] + ".txt"

transcription\_path = os.path.join(app.config['UPLOAD\_FOLDER'], transcription\_file)

with open(transcription\_path, "w") as f:

improved\_text = improve\_transcription(result["text"])

f.write(improved\_text)

# Eliminar el archivo de audio original

os.remove(audio\_file\_path)

return improved\_text

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**“INDEX.HTML”**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>ARIEL</title>

<link rel="stylesheet" href="static/css/styles.css">

<script>

async function submitForm(event) {

event.preventDefault();

// Mostrar el mensaje de "Procesando..."

const processingMessage = document.getElementById("processing-message");

processingMessage.textContent = "Procesando...";

const form = document.getElementById("transcription-form");

const formData = new FormData(form);

const response = await fetch("/upload", {

method: "POST",

body: formData,

});

const transcription = await response.text();

document.getElementById("transcription").value = transcription;

// Ocultar el mensaje de "Procesando..."

processingMessage.textContent = "";

}

function downloadTranscription() {

const transcription = document.getElementById("transcription").value;

if (transcription === "") {

alert("No hay transcripción para descargar.");

return;

}

const blob = new Blob([transcription], { type: "text/plain;charset=utf-8" });

const url = URL.createObjectURL(blob);

const link = document.createElement("a");

link.href = url;

link.download = "transcripcion.txt";

link.click();

URL.revokeObjectURL(url);

}

function updateFileName() {

const fileInput = document.getElementById("file-input");

const fileName = document.getElementById("file-name");

if (fileInput.files.length > 0) {

fileName.textContent = fileInput.files[0].name;

} else {

fileName.textContent = "";

}

}

</script>

</head>

<body>

<p> </p>

<h1>TRANSCRIPTOR DE AUDIO</h1>

<p>¡Hola!</p>

<p>

Soy ARIEL, tu asistente para la redacción inteligente de escritos legales. En este módulo, utilizo tecnología de procesamiento natural del lenguaje e inteligencia artificial proveída por OpenAi (Whisper + Chat GPT-4) para transcribir tus archivos de audio a texto.

</p>

<p>

Intentaré ser lo más fiel posible al contenido original. Sin embargo, si no entiendo lo que se dice, usaré el contexto para dar sentido a la transcripción.

</p>

<p>

Recuerda: estoy en fase de entrenamiento, así que siempre revisa el producto final y contrástalo con el audio que has cargado.

</p>

<form id="transcription-form" onsubmit="submitForm(event)">

<div class="file-upload">

<input type="file" id="file-input" name="file" accept="audio/\*" required onchange="updateFileName()">

<label for="file-input">Seleccionar archivo</label>

<span id="file-name"></span>

</div>

<p> </p>

<button type="submit">Transcribir</button>

<div id="processing-message"></div>

</form>

<div id="status-message"></div>

<textarea id="transcription" rows="10" cols="50" readonly></textarea>

<br>

<button id="download-btn" onclick="downloadTranscription()">Descargar transcripción</button>

</body>

</html>

**“STYLES.CSS”**

\* {

font-family: Arial, sans-serif;

}

body {

margin: 0;

padding: 0;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

min-height: 100vh;

background-color: #FFFFFF;

}

h1 {

font-size: 2rem;

margin-bottom: 1.5rem;

text-align: center;

}

p {

font-size: 1rem;

line-height: 1.5;

text-align: justify;

max-width: 600px;

}

form {

background-color: #fff;

padding: 2rem;

margin-bottom: 1.5rem;

border-radius: 4px;

box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);

display: flex;

flex-direction: column;

align-items: center;

}

.file-upload {

display: flex;

flex-direction: column;

align-items: center;

margin-bottom: 1.5rem;

}

.file-upload input[type="file"] {

display: none;

}

.file-upload label {

background-color: #007bff;

color: #fff;

border: none;

padding: 0.5rem 1rem;

border-radius: 4px;

cursor: pointer;

font-size: 1rem;

text-align: center;

display: inline-block;

overflow: hidden;

white-space: nowrap;

text-overflow: ellipsis;

width: 100%;

max-width: 200px;

}

.file-upload label:hover {

background-color: #0056b3;

}

.file-upload span {

display: block;

margin-top: 0.5rem;

font-size: 0.9rem;

color: #333;

}

button[type="submit"], #download-btn {

background-color: #007bff;

color: #fff;

border: none;

padding: 0.5rem 1rem;

border-radius: 4px;

cursor: pointer;

font-size: 1.2rem;

margin-bottom: 1.5rem;

}

button[type="submit"]:hover, #download-btn:hover {

background-color: #0056b3;

}

textarea {

width: 100%;

max-width: 600px;

border: 1px solid #ccc;

border-radius: 4px;

padding: 0.5rem;

}

MODEL: “LARGE”  
CHAT-GPT: “GPT-4”

Temperatura: 1

+

-TEXTBOX-

+

RESUMEN

Terminal:

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/

python3 app.py

PLAYGROUND

System:

Eres mi asistente de programación. Yo no tengo ninguna experiencia en programación ni desarrollo de aplicaciones. Deberás guiarme paso a paso y ser muy clara en tus indicaciones. Muy importante: espera a que te confirme que he hecho un paso antes de avanzar con el siguiente. Tengo una MacOS, Python3 y uso Safari. Cuando deba hacer una modificación de un código, transcríbelo completamente para que yo solamente deba copiar y pegarlo en su integridad.

Prompt:

Vamos a trabajar en la apariencia de la interfaz de mi programa para la transcripción de audio a texto. Actualmente, tengo tres archivos ("index.html"; "app.py" y "styles.css"), cuyo contenido te transcribiré en dos mensajes a continuación.

1. "index.html"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/templates/index.html

Contenido:

2. "app.py"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/app.py

Contenido:

3. "styles.css"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/templates/static/css/styles.css

Contenido:

**“APP.PY”**

from flask import Flask, render\_template, request, send\_from\_directory

import os

import certifi

import whisper

import openai

import json

import requests

# Establecer la variable de entorno SSL\_CERT\_FILE

os.environ["SSL\_CERT\_FILE"] = certifi.where()

app = Flask(\_\_name\_\_, static\_folder="templates/static")

# Configurar el directorio de carga

UPLOAD\_FOLDER = 'uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Cargar el modelo de Whisper

model = whisper.load\_model("large")

# Configurar la API Key de OpenAI

API\_KEY = "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"

API\_ENDPOINT = "https://api.openai.com/v1/chat/completions"

def generate\_chat\_completion(messages, model="gpt-4", temperature=1, max\_tokens=None):

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {API\_KEY}",

}

data = {

"model": model,

"messages": messages,

"temperature": temperature,

}

if max\_tokens is not None:

data["max\_tokens"] = max\_tokens

response = requests.post(API\_ENDPOINT, headers=headers, data=json.dumps(data))

if response.status\_code == 200:

return response.json()["choices"][0]["message"]["content"]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def improve\_transcription(text):

messages = [

{"role": "system", "content": "Eres mi asistente para la redacción de textos."},

{"role": "user", "content": f"Corrige la redacción de este texto. Cada vez que cambie el interlocutor, inicia un nuevo párrafo con un guión: {text}"}

]

improved\_text = generate\_chat\_completion(messages, model="gpt-4", temperature=1)

return improved\_text.strip()

def summarize\_text(text):

messages = [

{"role": "system", "content": "Eres mi asistente para resumir textos."},

{"role": "user", "content": f"Por favor, resume este texto: {text}"}

]

summary = generate\_chat\_completion(messages, model="gpt-4", temperature=1)

return summary.strip()

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/upload', methods=['POST'])

def upload():

if request.method == 'POST':

# Obtener el archivo de audio de la solicitud

audio\_file = request.files['file']

# Guardar el archivo en el directorio de carga

audio\_file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], audio\_file.filename)

audio\_file.save(audio\_file\_path)

# Transcribir el archivo y guardar la transcripción en un archivo .txt

result = model.transcribe(audio\_file\_path, language="Spanish")

transcription\_file = os.path.splitext(audio\_file.filename)[0] + ".txt"

transcription\_path = os.path.join(app.config['UPLOAD\_FOLDER'], transcription\_file)

with open(transcription\_path, "w") as f:

improved\_text = improve\_transcription(result["text"])

f.write(improved\_text)

# Eliminar el archivo de audio original

os.remove(audio\_file\_path)

return improved\_text

@app.route('/summarize', methods=['POST'])

def summarize():

if request.method == 'POST':

transcription = request.data.decode("utf-8")

summary = summarize\_text(transcription)

return summary

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**“INDEX.HTML”**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>ARIEL</title>

<link rel="stylesheet" href="static/css/styles.css">

<script>

async function submitForm(event) {

event.preventDefault();

showProcessingMessage();

const form = document.getElementById("transcription-form");

const formData = new FormData(form);

const response = await fetch("/upload", {

method: "POST",

body: formData,

});

const transcription = await response.text();

document.getElementById("transcription").value = transcription;

hideProcessingMessage();

}

function showProcessingMessage() {

document.getElementById("status-message").innerText = "Procesando...";

}

function hideProcessingMessage() {

document.getElementById("status-message").innerText = "";

}

function downloadTranscription() {

const transcription = document.getElementById("transcription").value;

if (transcription === "") {

alert("No hay transcripción para descargar.");

return;

}

const blob = new Blob([transcription], { type: "text/plain;charset=utf-8" });

const url = URL.createObjectURL(blob);

const link = document.createElement("a");

link.href = url;

link.download = "transcripcion.txt";

link.click();

URL.revokeObjectURL(url);

}

async function downloadSummary() {

const transcription = document.getElementById("transcription").value;

if (transcription === "") {

alert("No hay transcripción para resumir.");

return;

}

showProcessingMessage();

const response = await fetch("/summarize", {

method: "POST",

body: transcription,

});

const summary = await response.text();

hideProcessingMessage();

const blob = new Blob([summary], { type: "text/plain;charset=utf-8" });

const url = URL.createObjectURL(blob);

const link = document.createElement("a");

link.href = url;

link.download = "resumen.txt";

link.click();

URL.revokeObjectURL(url);

}

function showFilename() {

const fileInput = document.getElementById("file-input");

const filename = fileInput.files[0].name;

document.getElementById("file-name").innerText = filename;

}

</script>

</head>

<body>

<p> </p>

<h1>TRANSCRIPCIÓN</h1>

<p>

Soy ARIEL®, tu asistente para la redacción inteligente de escritos legales. En este módulo, utilizo inteligencia artificial con tecnología de procesamiento natural del lenguaje para transcribir tus archivos de audio a texto. Reconozco la mayoría de los formatos usuales (.mp4, .mp4, .m4a, etc.).

</p>

<p>

Intentaré ser lo más fiel posible al contenido original. Sin embargo, si no entiendo lo que se dice, usaré el contexto para dar sentido a la transcripción.

</p>

<p>

Recuerda: estoy en fase de entrenamiento, así que siempre revisa el producto final y contrástalo con el audio que has cargado.

</p>

<p> </p>

<form id="transcription-form" onsubmit="submitForm(event)">

<input type="file" id="file-input" name="file" accept="audio/\*" onchange="showFilename()" required>

<label for="file-input" class="file-upload-label">Seleccionar archivo</label>

<span id="file-name"></span>

<button type="submit" class="download-btn">Transcribir</button>

</form>

<div id="status-message"></div>

<textarea id="transcription" rows="10" cols="50" readonly></textarea>

<br>

<button id="download-btn" class="download-btn" onclick="downloadTranscription()">Descargar transcripción</button>

<button id="download-summary-btn" class="download-btn" onclick="downloadSummary()">Descargar resumen</button>

</body>

</html>

**“STYLES.CSS”**

\* {

font-family: Arial, sans-serif;

}

body {

margin: 0;

padding: 0;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

min-height: 100vh;

background-color: #FFFFFF;

}

h1 {

font-size: 2rem;

margin-bottom: 1.5rem;

text-align: center;

}

p {

font-size: 1rem;

line-height: 1.5;

text-align: justify;

max-width: 600px;

}

form {

background-color: #fff;

padding: 2rem;

margin-bottom: 1.5rem;

border-radius: 4px;

box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);

display: flex;

flex-direction: column;

align-items: center;

}

.file-upload input[type="file"] {

display: none;

}

.file-upload-label {

background-color: #007bff;

color: #fff;

border: none;

padding: 0.5rem 0.5rem;

border-radius: 4px;

cursor: pointer;

font-size: 1.2rem;

text-align: center;

display: inline-block;

overflow: hidden;

white-space: nowrap;

text-overflow: ellipsis;

width: 100%;

max-width: 200px;

margin-bottom: 1.5rem;

}

.file-upload-label:hover {

background-color: #0056b3;

}

#file-name {

display: block;

margin-top: 0.2rem;

margin-bottom: 2rem;

font-size: 0.9rem;

color: #333;

}

.download-btn {

background-color: #007bff;

color: #fff;

border: none;

padding: 0.5rem 1rem;

border-radius: 4px;

cursor: pointer;

font-size: 1.2rem;

margin-bottom: 1.5rem;

}

.download-btn:hover {

background-color: #0056b3;

}

textarea {

width: 100%;

max-width: 600px;

border: 1px solid #ccc;

border-radius: 4px;

padding: 0.5rem;

}

#file-input {

position: absolute;

width: 1px;

height: 1px;

padding: 0;

margin: -1px;

overflow: hidden;

clip: rect(0, 0, 0, 0);

border: 0;

}

MODEL: “LARGE”  
CHAT-GPT: “GPT-4”

Temperatura: 1

+

-TEXTBOX-

+

RESUMEN

+

TRANSCRIPCIÓN LITERAL

Terminal:

cd /Users/andresdiaz/Library/Mobile\ Documents/com~apple~TextEdit/Documents/Whisper/

python3 app.py

PLAYGROUND

System:

Eres mi asistente de programación. Yo no tengo ninguna experiencia en programación ni desarrollo de aplicaciones. Deberás guiarme paso a paso y ser muy clara en tus indicaciones. Muy importante: espera a que te confirme que he hecho un paso antes de avanzar con el siguiente. Tengo una MacOS, Python3 y uso Safari. Cuando deba hacer una modificación de un código, transcríbelo completamente para que yo solamente deba copiar y pegarlo en su integridad.

Prompt:

Vamos a trabajar en la apariencia de la interfaz de mi programa para la transcripción de audio a texto. Actualmente, tengo tres archivos ("index.html"; "app.py" y "styles.css"), cuyo contenido te transcribiré en dos mensajes a continuación.

1. "index.html"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/templates/index.html

Contenido:

2. "app.py"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/app.py

Contenido:

3. "styles.css"

Ruta: /Users/andresdiaz/Library/Mobile Documents/com~apple~TextEdit/Documents/Whisper/templates/static/css/styles.css

Contenido:

**“APP.PY”**

from flask import Flask, render\_template, request, send\_from\_directory

import os

import certifi

import whisper

import openai

import json

import requests

# Establecer la variable de entorno SSL\_CERT\_FILE

os.environ["SSL\_CERT\_FILE"] = certifi.where()

app = Flask(\_\_name\_\_, static\_folder="templates/static")

# Configurar el directorio de carga

UPLOAD\_FOLDER = 'uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Cargar el modelo de Whisper

model = whisper.load\_model("large")

# Configurar la API Key de OpenAI

API\_KEY = "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"

API\_ENDPOINT = "https://api.openai.com/v1/chat/completions"

def generate\_chat\_completion(messages, model="gpt-4", temperature=0.5, max\_tokens=None):

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {API\_KEY}",

}

data = {

"model": model,

"messages": messages,

"temperature": temperature,

}

if max\_tokens is not None:

data["max\_tokens"] = max\_tokens

response = requests.post(API\_ENDPOINT, headers=headers, data=json.dumps(data))

if response.status\_code == 200:

return response.json()["choices"][0]["message"]["content"]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def improve\_transcription(text):

messages = [

{"role": "system", "content": "Eres mi asistente para la redacción de textos."},

{"role": "user", "content": f"Corrige la redacción de este texto. Cada vez que cambie el interlocutor, inicia un nuevo párrafo con un guión: {text}"}

]

improved\_text = generate\_chat\_completion(messages, model="gpt-4", temperature=1)

return improved\_text.strip()

def summarize\_text(text):

messages = [

{"role": "system", "content": "Eres mi asistente para resumir textos."},

{"role": "user", "content": f"Por favor, resume este texto: {text}"}

]

summary = generate\_chat\_completion(messages, model="gpt-4", temperature=0.5)

return summary.strip()

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/upload', methods=['POST'])

def upload():

return process\_upload(False)

@app.route('/upload\_literal', methods=['POST'])

def upload\_literal():

return process\_upload(True)

def process\_upload(literal\_transcription):

if request.method == 'POST':

# Obtener el archivo de audio de la solicitud

audio\_file = request.files['file']

# Guardar el archivo en el directorio de carga

audio\_file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], audio\_file.filename)

audio\_file.save(audio\_file\_path)

# Transcribir el archivo y guardar la transcripción en un archivo .txt

result = model.transcribe(audio\_file\_path, language="Spanish")

transcription\_file = os.path.splitext(audio\_file.filename)[0] + ".txt"

transcription\_path = os.path.join(app.config['UPLOAD\_FOLDER'], transcription\_file)

with open(transcription\_path, "w") as f:

if literal\_transcription:

f.write(result["text"])

output\_text = result["text"]

else:

improved\_text = improve\_transcription(result["text"])

f.write(improved\_text)

output\_text = improved\_text

# Eliminar el archivo de audio original

os.remove(audio\_file\_path)

return output\_text

@app.route('/summarize', methods=['POST'])

def summarize():

if request.method == 'POST':

transcription = request.data.decode("utf-8")

summary = summarize\_text(transcription)

return summary

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)