Universidad Nacional Autónoma de México

Facultad de Ingeniería

Ingeniería en computación

Inteligencia Artificial (406)

Profesor: Ing. Jorge Alberto Hernández Nieto

Semestre 2025-1

Proyecto Final

Grupo: 5

Integrantes del Equipo:

Bravo Luna Ivonne Monserrat

Cruz Maldonado Armando

Diaz Gonzalez Rivas Angel Iñaqui

Olivos Jiménez Luis Mario

Siliano Haller Rodrigo

Zarco Romero José Alberto

Índice o contenido

1.*Objetivo*

Se desarrollará un sistema que a través del reconocimiento facial en imágenes faciales realice la clasificación de género, a partir del género detectado por el sistema ajustará un chatbot para la proporción de respuestas personalizadas, aplicando métodos como el entrenamiento de un modelo de reconocimiento facial, la detección y predicción en tiempo real mediante una cámara, y la interacción dinámica con el usuario mediante un chatbot adaptado a las características del individuo reconocido.

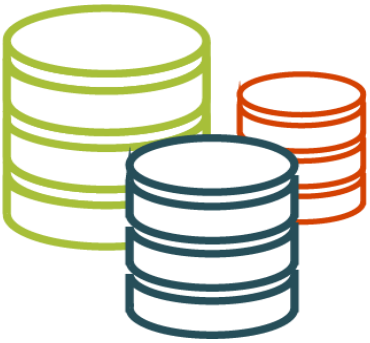
2.*Introducción*

En la actualidad, los sistemas inteligentes que combinan visión por computadora y procesamiento de lenguaje natural están transformando múltiples industrias al mejorar la interacción entre humanos y máquinas. Este proyecto propone el desarrollo de un sistema integrado capaz de reconocer el género de una persona mediante análisis de imágenes faciales y generar una respuesta adaptada a través de un chatbot. El sistema utiliza un enfoque basado en Eigenfaces para la clasificación de género y procesamiento en tiempo real, complementado con técnicas de procesamiento de lenguaje natural para personalizar la interacción según el género detectado.

Para que se logre el cumplimiento de nuestro sistema, usamos librerías importantes, entre ellas está OpenCV el cual su principal objetivo es facilitar el procesamiento de imágenes y videos en tiempo real, proporcionando un conjunto amplio de algoritmos que permiten realizar tareas como la detección de objetos, el reconocimiento facial, el análisis de movimiento y muchas otras.[1]

En nuestro sistema el uso de OpenCV nos permitirá hacer la función de reconocimiento facial, procesamiento de imágenes, captura de video en tiempo real y

la detección de rostros, con esto garantizamos que el programa identifique y diferencie entre un hombre y mujer para la configuración del sistema. Además, se aplicará el uso de dataset, que es un conjunto de datos, ordenado bajo un sistema de almacenamiento que otorga los lineamientos principales de búsqueda o directorio de la información que se quiere trabajar.[2]



Este conjunto de datos por supuesto que se puede utilizar para muchas cosas, dependiendo de la metodología, orientación o tratamiento que se le quiera dar a la información. Su finalidad es hacer mucho más fácil la vida a las personas, automatizar tareas o simplemente analizar información de una manera más ágil. [2]

Aplicando el uso de dataset, nos servirá para la organización y almacenamiento de imágenes para el entrenamiento del sistema, se busca que el programa pueda usar la información recopilada, para el reconocimiento de rostros.

En nuestro programa se configura un chatbot que adapta sus respuestas en función del género de la persona reconocida, creando un entorno de interacción dinámica y amigable. Este enfoque tiene aplicaciones potenciales en áreas como atención al cliente, educación, entretenimiento y desarrollo de interfaces accesibles y personalizadas.

3.Desarrollo

3.1 Proyecto. Ejercicio Propuesto

El programa realiza un chatbot el cual simula la conversación humana con el usuario, implementa una parte fundamental del sistema de reconocimiento facial, específicamente la fase de entrenamiento del modelo para la clasificación de género donde usa un dataset de 500 imágenes entre hombres y mujeres (El programa puede almacenar más cantidad de imágenes, solo trabajamos con 500 por el peso que puede generar dicho programa). Después del reconocimiento el

programa se adaptará hacia el tipo de usuario quien esté usando el chatbot, para que nuestro programa cumpla su funcionamiento se dividirá por 3 secciones:

1. Entrenamiento del modelo.

   Función principal: Carga y realiza el preprocesamiento de imágenes de diferentes clases (hombre y mujer).

   Acciones que realiza:

   - Lectura de imágenes en escala de grises.
   - Redimensionamiento de las imágenes a un tamaño estándar (150x150 píxeles).
   - Asignación de etiquetas a las imágenes para su posterior uso en un modelo de clasificación.
   - Entrenamiento de un modelo EigenFaceRecognizer (de OpenCV) con las imágenes procesadas.
   - Guarda el modelo entrenado en un archivo .xml

2. Clasificación de imágenes

   Función principal: Realiza el procesamiento y la organización de un dataset para la clasificación de género.

   Acciones que realiza:

   - Carga un dataset con imágenes y un archivo. mat (con metadatos de las imágenes).
   - Clasifica las imágenes en carpetas separadas (hombre y mujer) basándose en los metadatos de género.
   - Controla el número máximo de imágenes procesadas por género.
   - Ajusta las rutas de las imágenes y valida los datos.

3. Reconocimiento de género.

   Función principal: realizará el reconocimiento del género tomando como referencia el modelo entrenado para el uso del chatbot

   Acciones que realiza:

   - Carga del modelo de reconocimiento de género (basado en el modelo realizado en el entrenamiento).
   - Utiliza un clasificador de rostros para detectar caras en imágenes capturadas en tiempo real con la cámara.
   - Clasifica el rostro como hombre o mujer y muestra la predicción junto con el nivel de confianza.
   - Activa un chatbot específico según el género detectado, utilizando respuestas predeterminadas.

3.1.1 Código

- Código entrenar_modelo.py

```
import cv2
import os
import numpy as np
import time

# Ruta al dataset procesado
processedPath = '/Users/angelinaquidiazgonzalezrivas/Documents/Proyecto/Python/CV/Datasets/Procesado'  # Carpeta donde están "Hombre" y "Mujer"
clases = os.listdir(processedPath)  # Listará las carpetas "Hombre" y "Mujer"
print("Clases encontradas: ", clases)

labels = []
facesData = []
label = 0

# Leer las imágenes de cada clase
for clase in clases:
    clasePath = os.path.join(processedPath, clase)

    print(f"Leyendo imágenes de la clase: {clase}")
    time.sleep(2)
    flag = 0
    for fileName in os.listdir(clasePath):
        if flag <= 450:
            filePath = os.path.join(clasePath, fileName)
            print(f"Procesando archivo: {filePath}")

            # Cargar imagen en escala de grises
            imagen = cv2.imread(filePath, 0)
            if imagen is None:
                print(f"Error al cargar imagen: {filePath}")
                continue
```

```
            # Redimensionar la imagen a 150x150
            imagen = cv2.resize(imagen, (150, 150))
            facesData.append(imagen)
            labels.append(label)
            flag += 1

    label += 1

# Verificar equilibrio de las clases
print(f"Número de imágenes por clase: {np.bincount(labels)}")

cv2.destroyAllWindows()  # Cerrar todas las ventanas de OpenCV

# Entrenar el modelo EigenFaces
print("Entrenando modelo...")
reconocimientoGenero = cv2.face.EigenFaceRecognizer_create()
reconocimientoGenero.train(facesData, np.array(labels))

# Guardar el modelo entrenado
modeloPath = 'D:/Python/CV/modeloGenero.xml'
reconocimientoGenero.write(modeloPath)
print(f"Modelo almacenado en: {modeloPath}")
```

El componente central del código es la implementación del algoritmo EigenFaces, una técnica probada en el reconocimiento facial. Este algoritmo se entrena con las imágenes procesadas y sus etiquetas correspondientes, aprendiendo a distinguir las características faciales asociadas a cada género.

- Código procesar_dataset.py

```
import os
import scipy.io
import shutil
import numpy as np

# Ruta al dataset y archivo .mat
datasetPath = '/Users/angelinaquidiazgonzalezrivas/Documents/Proyecto/Python/CV/Datasets/IMDB-WIKI/wiki_crop/'
labelsFile = '/Users/angelinaquidiazgonzalezrivas/Documents/Proyecto/Python/CV/Datasets/IMDB-WIKI/wiki_crop/wiki.mat'

# Verificar que el archivo .mat exista
if not os.path.exists(labelsFile):
    raise FileNotFoundError(f"El archivo wiki.mat no se encuentra en: {labelsFile}")

# Carpetas de destino
processedPath = 'D:/Python/CV/Datasets/Procesado'
hombrePath = os.path.join(processedPath, 'Hombre')
mujerPath = os.path.join(processedPath, 'Mujer')
os.makedirs(hombrePath, exist_ok=True)
os.makedirs(mujerPath, exist_ok=True)

# Cargar el archivo .mat
mat = scipy.io.loadmat(labelsFile)

# Ajustar las claves
gender = mat['wiki']['gender'][0][0][0]  # Género (1 = hombre, 0 = mujer)
imagePaths = mat['wiki']['full_path'][0][0][0]  # Rutas relativas

# Depurar rutas
print("Primeras 10 rutas relativas del archivo .mat:")
print([str(path[0]) for path in imagePaths[:10]])

# Contadores
male_count = 0
female_count = 0
MAX_IMAGES_PER_GENDER = 600
```

En esta sección se configuran las carpetas de destino para el procesamiento, creando directorios separados para Hombre y Mujer, dentro de la ruta 'processed'. Utiliza os.makedirs() con el parámetro exist_ok=True para evitar errores si los directorios ya existen.

```
# Procesar imágenes
for i, imgPath in enumerate(imagePaths):
    try:
        if male_count >= MAX_IMAGES_PER_GENDER and female_count >= MAX_IMAGES_PER_GENDER:
            break

        # Convertir la ruta relativa a una cadena estándar de Python
        relativePath = str(imgPath[0])  # Limpiar el formato de NumPy
        imgFullPath = os.path.join(datasetPath, relativePath.replace('\\', '/'))  # Normalizar separadores

        if not os.path.exists(imgFullPath):
            print(f"Imagen encontrada: {imgFullPath}")
            continue

        if i >= len(gender) or np.isnan(gender[i]):
            print(f"Género inválido en índice {i}")
            continue

        genderLabel = int(gender[i])  # Género
        if genderLabel == 1 and male_count < MAX_IMAGES_PER_GENDER:
            shutil.copy(imgFullPath, os.path.join(hombrePath, os.path.basename(imgFullPath)))
            male_count += 1
        elif genderLabel == 0 and female_count < MAX_IMAGES_PER_GENDER:
            shutil.copy(imgFullPath, os.path.join(mujerPath, os.path.basename(imgFullPath)))
            female_count += 1

    except Exception as e:
        print(f"Error procesando la imagen {imgPath}: {e}")

print("Clasificación completada.")
print(f"Imágenes procesadas: {male_count} hombres, {female_count} mujeres.")
print("Imágenes guardadas en:", processedPath)
```

Finalmente, se colocan los contadores male_count y female_count, los cuales establecen un límite máximo de 600 imágenes por género (MAX_IMAGES_PER_GENDER), lo que ayuda a mantener un conjunto de datos balanceado.

- Código reconocer_genero.py

```
import cv2
import os
from nltk.chat.util import Chat, reflections
import sys  # Para salir del programa completamente

# Ruta al modelo entrenado
modeloPath = '/Users/angelinaquidiazgonzalezrivas/Documents/Proyecto/ReconocimientoFa/modeloGenero.xml'
processedPath = '/Users/angelinaquidiazgonzalezrivas/Documents/Proyecto/Python/CV/Datasets/Procesado'
clases = os.listdir(processedPath)  # Lee las clases ("Hombre", "Mujer")

# Cargar el modelo entrenado
reconocimientoGenero = cv2.face.EigenFaceRecognizer_create()
reconocimientoGenero.read(modeloPath)

# Inicializar la cámara
cap = cv2.VideoCapture(0, cv2.CAP_AVFOUNDATION)
clasificadorRostro = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Configuración del chatbot
pares_hombre = [
    [r"hola|hey|buenas", ["Hola, caballero. ¿Cómo estás?"]],
    [r"como estas?", ["Estoy bien, ¿y tú?"]],
    [r"que haces?", ["Converso contigo."]],
    [r"finalizar", ["Adiós, fue un gusto hablar contigo."]]
]

pares_mujer = [
    [r"hola|hey|buenas", ["Hola, señora/señorita. ¿Cómo te va?"]],
    [r"como estas?", ["Estoy bien, ¿y tú?"]],
    [r"que haces?", ["Estoy aquí para conversar contigo."]],
    [r"finalizar", ["Adiós, espero volver a hablar contigo pronto."]]
]
```

En este punto se observa la ruta al modelo entrenado, haciendo el siguiente punto, la carga de este modelo entrenando, para pasar ahora sí a la cámara del equipo, o incluso de un video, para poder así, comenzar con la detección de rostros.

Además de eso, se muestra la configuración de los comentarios que se le pueden realizar a nuestro Chatbot, así como las respuestas que este puede mostrar, tanto del lado de las mujeres, como del lado de los hombres, mostradas dentro de las ejecuciones del ejercicio, que se encuentran una sección más adelante.

```
# Función para iniciar el chatbot
def iniciar_chat(pares):
    chat = Chat(pares, reflections)
    print("Escribe algo para comenzar. Escribe 'finalizar' para salir.")

    while True:
        user_input = input("Tú: ")
        if user_input.lower() == "finalizar":
            print("Chatbot: Adiós.")
            sys.exit()  # Termina el programa completamente

        response = chat.respond(user_input)
        if response:
            print(f"Chatbot: {response}")
        else:
            print("Chatbot: Interesante, cuéntame más.")  # Respuesta genérica para entradas no reconocidas

# Bucle principal
while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = gray.copy()

    # Detectar rostros en la imagen
    caras = clasificadorRostro.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in caras:
        rostro = auxFrame[y:y + h, x:x + w]
        rostro = cv2.resize(rostro, (150, 150), interpolation=cv2.INTER_CUBIC)
```

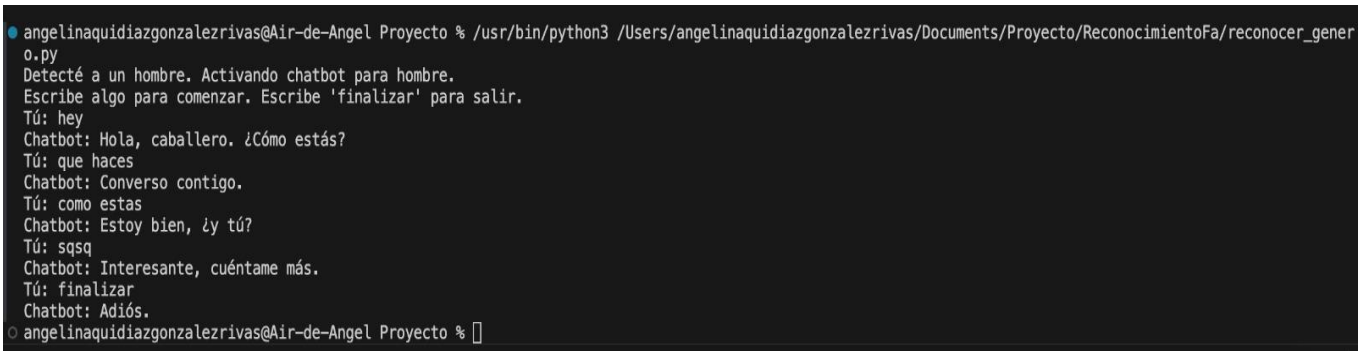
En esta sección se analizan las características faciales extraídas y determina el género de la persona, junto con un nivel de confianza asociado a la predicción. El sistema utiliza esta información para adaptar dinámicamente su comportamiento, activando diferentes conjuntos de respuestas del chatbot según el género detectado. Además de contener las posibles respuestas que el chatbot mostrará una vez encuentre una respuesta o el usuario coloque uno de los comentarios que se le plantean.

```
        # Predecir género
        resultado = reconocimientoGenero.predict(rostro)
        genero = clases[resultado[0]]
        confianza = resultado[1]

        # Mostrar resultado en pantalla
        if confianza < 8000:  # Ajusta este umbral según el modelo
            cv2.putText(frame, f"{genero} ({confianza:.2f})", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

            # Activar chatbot según el género
            if genero == "Hombre":
                print("Detecté a un hombre. Activando chatbot para hombre.")
                iniciar_chat(pares_hombre)
            elif genero == "Mujer":
                print("Detecté a una mujer. Activando chatbot para mujer.")
                iniciar_chat(pares_mujer)
        else:
            cv2.putText(frame, "Desconocido", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

    cv2.imshow('Reconocimiento de Género', frame)

    if cv2.waitKey(1) == 27:  # Presiona 'Esc' para salir
        break

cap.release()
cv2.destroyAllWindows()
```

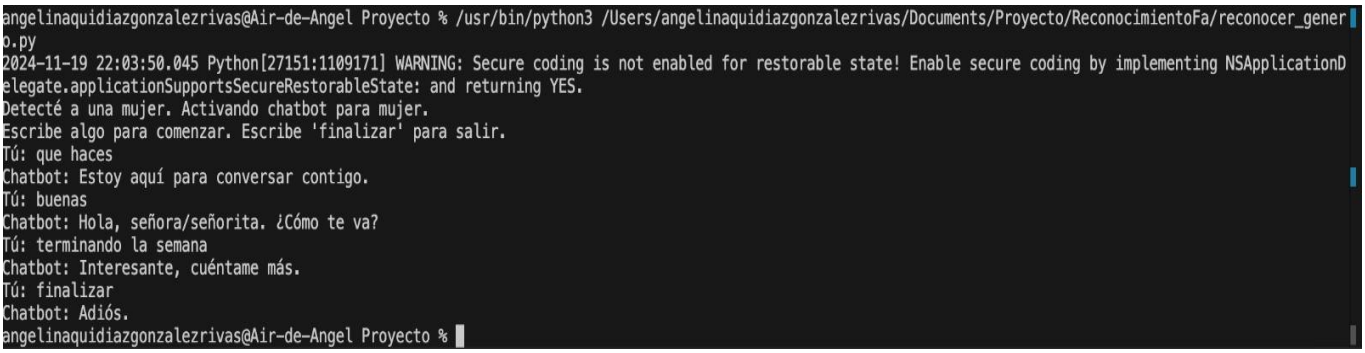
En esta sección y después del resultado obtenido con el análisis del rostro de las personas, se muestra lo que el chatbot realizará para comenzar la conversación de acuerdo con el género detectado, incluso colocando una condición en la que se almacena un mensaje por no haber reconocido a un Hombre o a una Mujer. Una vez que se detecte a una de las dos opciones existentes, el chateo comenzará, logrando así tener la secuencia que se mostrará dentro de la ejecución del programa que se encuentra en la siguiente sección.

3.1.2 Ejecución del programa

- En caso de detección de un hombre:



- En caso de detección de una mujer:



4.Conclusiones

El desarrollo del proyecto nos permitió un flujo de trabajo bien estructurado que abarca desde la captura y preprocesamiento de datos, el entrenamiento de un modelo predictivo, hasta la implementación de un sistema en tiempo real capaz de reconocer rostros de manera precisa y eficiente, con la implementación de librerías de código abierto como lo son OpenCV y NumPy nos permitieron hacer el manejo el de estas estructuras de manera eficiente.

El sistema establece una base confiable al capturar imágenes faciales en tiempo real, junto con un proceso automatizado que crea carpetas específicas para cada persona registrada. Todas las imágenes se procesan de manera uniforme a un tamaño estándar de 150 x 150 píxeles, lo que nos asegura que todos los datos mantengan la misma calidad y formato. Esta organización estructurada mejora notablemente el desempeño de nuestro modelo y hace que todo el sistema funcione de manera más eficiente.

A partir de esta base logramos el entrenamiento de un modelo predictivo basado en EigenFaces,la lectura de imágenes y la asociación de etiquetas facilitaron la organización de los datos, aplicando el procesamiento de dataset con el cual es un elemento para la optimización de nuestro sistema, se trabajó con un rango de 500 imágenes para el manejo de datos y almacenamiento.

El sistema implementa un módulo de reconocimiento facial en tiempo real que se integra con nuestra base de datos biométrica previamente entrenada. Este realiza un análisis dinámico de las características faciales del usuario para determinar su género, lo cual permite una personalización automatizada de las respuestas del chatbot, optimizando así la experiencia de interacción según los parámetros demográficos identificados.

5.Bibliografía

[1] C. G. Fernández. “OpenCV: Introducción y su rol en la visión por computadora | OpenWebinars”. OpenWebinars.net. Accedido el 21 de noviembre de 2024. [En línea]. Disponible: https://openwebinars.net/blog/opencv-introduccion-y-su-rol-en-la-vision-por-computadora/

[2] D. C. Solis. “Datasets: Qué son y cómo acceder a ellos | OpenWebinars”. OpenWebinars.net. Accedido el 22 de noviembre de 2024. [En línea]. Disponible: https://openwebinars.net/blog/datasets-que-son-y-como-acceder-a-ellos/