UNIVERSIDAD POLITÉCNICA SALESIANA SEDE GUAYAQUIL

CARRERA

INGENIERIA EN COMPUTACIÓN

APRENDIZAJE DE MAQUINA

TEMA

CHAT BOT + RECONOCIMIENTO DE EMOCIONES

AUTORES

JIMMY RUGEL BEJARANO

GABRIEL RECALDE

MILENA BELTRÁN

GRUPO

2

Contenido

Introducción

Este proyecto fue realizado en combinación de un Chat Bot implementado en Telegram y un código preexistente para la detección de emociones en rostros de personas, esto gracias a la integración de ambos casos a través de Python en Google Colaboratory.

No obstante, este documento da una breve descripción de la creación e implementación del proyecto en general por lo que se recomienda visitar las siguientes páginas web para un entendimiento más a fondo de las secciones próximas del documento, también para dar créditos a los creadores de ambos códigos, el Chat Bot con Telegram y el código utilizado para el Reconocimiento de Emociones.

Chat Bot Telegram:

https://robologs.net/2019/07/30/como-programar-un-chatbot-para-telegram-en-python/#:~:text=Pulsa%20el%20bot%C3%B3n%20%E2%80%9CStart%E2%80%9D%20o,chatbot%20escribiendo%20el%20comando%20%2Fnewbot.

Reconocimiento de Emociones:

https://www.youtube.com/watch?v=7AY_Xgehyks

Cualquier duda en cuanto al proyecto y para una mejor asistencia por parte de quienes lo implementamos, puede enviar un mensaje explicando su duda con los siguientes datos:

- Asunto: **“P.A.M – Soporte”**
- Correo: **Jirube@hotmail.com**

Creación de Chat Bot en Telegram

Telegram es una plataforma de mensajería instantánea que cuenta con muchas funciones, una de ellas es la creación de Chat Bots, estos son administrados por una cuenta oficial de Telegram denominada Bot Father.

Es necesario comunicarse con esta cuenta para la creación y administración del Chat Bot, al momento de crearlo se entregará un Token único de acceso, con el Token se puede crear una conexión al Chat Bot desde múltiples plataformas online o desde la consola de algún tipo de lenguaje de programación. (Para este proyecto se utilizó Python.)

Implementación del Chat Bot de Telegram con Python

En este proyecto se utilizó la plataforma de Google Colaboratory para la ejecución del código Python por lo que la explicación estará basada en esta misma.

Lo primero que necesitamos saber es que el Chat Bot otorgado se maneja a través de URLs a los cuales se accede con la dirección que brinda Telegram + el Token del Chat Bot, seguido de la sección a la que se quiere acceder (en este caso los mensajes).

Al acceder a la URL de mensajes, estos se almacenan en un archivo JSON y aquí es donde el código en Python (Archivo: ChatBotRE.ipynb) entra en acción, primero instalamos la librería Request y demás librerías necesarias para el funcionamiento del Chat Bot y Reconocimiento de Emociones.

Librería Request

```
[ ]  !pip install requests
```

```
#Importar librerias
import json
import requests
import random
import cv2 #Libreria para la detección de rostros. entre otras cosas
```

Esta librería nos permitirá hacer peticiones a internet. (Todas las peticiones deben incluir el Token otorgado por Telegram)

```
#Variables para el Token y la URL del chatbot
TOKEN = "##################################"
URL = "https://api.telegram.org/bot" + TOKEN + "/"
```

De esta forma podremos acceder al archivo de los mensajes que Telegram proporciona, y se almacenarán en un Diccionario de Datos que se utilizará para extraer los datos necesarios para el análisis y respuesta de los mensajes del Chat Bot.

```python
def update(offset):
    #Llamar al metodo getUpdates del bot, utilizando un offset
    respuesta = requests.get(URL + "getUpdates" + "?offset=" + str(offset) + "&timeout=" + str(100))

    #Decodificar la respuesta recibida a formato UTF8
    mensajes_js = respuesta.content.decode("utf8")

    #Convertir el string de JSON a un diccionario de Python
    mensajes_diccionario = json.loads(mensajes_js)

    #Devolver este diccionario
    return mensajes_diccionario
```

A su vez se establecen algunos métodos que se utilizaran para determinar el tipo de mensaje y extraer los datos del usuario que envió el mensaje y poder contestarle correctamente.

```python
def info_mensaje(mensaje):

    #Comprobar el tipo de mensaje
    if "text" in mensaje["message"]:
        tipo = "texto"
    elif "sticker" in mensaje["message"]:
        tipo = "sticker"
    elif "animation" in mensaje["message"]:
        tipo = "animacion" #Nota: GIF son animaciones
    elif "photo" in mensaje["message"]:
        tipo = "foto"
        id_file= mensaje["message"]["photo"][0]["file_id"]
        mensajesF_diccionario= updateFoto(id_file)
    else:
        tipo = "otro" # el resto de tipos entra en la categoria "otro"
```

En este proyecto se incluyeron varios detalles, como el recibir una notificación especifica en un grupo de Telegram cada que un usuario pidiera asistencia (Soporte) con el Chat Bot. (en la flecha roja se coloca el id del grupo)

```python
    #Recoger la info del mensaje (remitente, id del chat e id del mensaje)
    nombre = mensaje["message"]["from"]["first_name"]
    id_chat = mensaje["message"]["chat"]["id"]
    id_update = mensaje["update_id"]
    if "username" in mensaje["message"]["from"]:
      usuario= mensaje["message"]["from"]["username"]
      msg="Ir al chat: t.me/"+usuario
    else:
      usuario=nombre
      msg="----------------------------------------------\nNueva Alerta\n\nUsuario: "+usuario+"\n\nHa intentado realizar
      requests.get(URL + "sendMessage?text=" + msg + "&chat_id=##############")

    #Devolver toda la informacion
    return tipo, id_chat, nombre, id_update, usuario, msg
```

```python
def leer_mensaje(mensaje):

    #Extraer el texto, nombre de la persona e id del último mensaje recibido
    texto = mensaje["message"]["text"]

    #Devolver las dos id, el nombre y el texto del mensaje
    return texto

def enviar_mensaje(idchat, texto):
    #Llamar el metodo sendMessage del bot, passando el texto y la id del chat
    requests.get(URL + "sendMessage?text=" + texto + "&chat_id=" + str(idchat))

def enviar_foto(idchat, texto, imagen):
    #Llamar el metodo sendMessage del bot, passando el texto y la id del chat
    requests.get(URL + "sendPhoto?chat_id=" + str(idchat) + "&caption="+ str(texto) + "&photo=" + str(imagen) )

def notificar(notificacion):
    #Llamar el metodo sendMessage del bot, passando el texto y la id del chat
    requests.get(URL + "sendMessage?chat_id=###############&text="+notificacion)
```

El Chat Bot se puede adaptar de muchas formas, cubriendo así una gran área de objetivos con los que se lo quiera usar manteniendo la misma estructura, el usuario podrá cambiar a su gusto las opciones de la memoria del Chat Bot al igual que las respuestas.

```
#Memoria del Chatbot 3.0
Memoria.update(Memoria.fromkeys(['/menu','menu','menú','/menú'],'\n¡Vamos a empezar!\nEnviame una foto de tu rostro (o la de alguien más) y determin
Memoria.update(Memoria.fromkeys(['/start','start'],'¡Bienvenid@!\nSoy P.A.M Tu Asistente Virtual de Detección de Emociones.'+Memoria['menu']))
Memoria.update(Memoria.fromkeys(['ayuda','/ayuda'],'\n¿Algún problema con la detección de emociones? Digita SOPORTE y una agente se pondra en contac
Memoria.update(Memoria.fromkeys(['/hola','hola','ola'],"Hola, "+nombre+"!"+Memoria['menu']))
Memoria.update(Memoria.fromkeys(['/adios','adios','adiós'],"Hasta pronto!, "+nombre))
Memoria.update(Memoria.fromkeys(['/soporte','soporte'],"Se ha generado el Ticket de Soporte\n(Ticket N° "+str(id_update)+")\n\n¡La ayuda está en cam
MCodigo.update(MCodigo.fromkeys(['/soporte','/comprar','comprar','soporte'],"https://qrcode.tec-it.com/API/QRCode?data=Proceso%20N°:%20"+str(id_upda
Memoria.update(Memoria.fromkeys(['gracias','entendido','muchas gracias','gracias por todo'],"Un placer haberte asistido, hasta otra."))
```

Lo que resta del código en cuanto a lo principal del Chat Bot es, el responder mensajes, esto se basa en condicionales para otorgar respuestas al usuario según lo que este escriba y una vez sea analizado/consultado en el Diccionario de Datos de Memoria, se enviara la respuesta.

```
#Generar una respuesta dependiendo del tipo de mensaje
if tipo == "texto":
    texto = leer_mensaje(i)
    if Memoria.get(texto.lower())!=None:
      if texto.lower() in MCodigo:
        texto_respuesta=Memoria.get(texto.lower())
        imagen=MCodigo.get(texto.lower())
        if idchat!=-1001146542990 and "username" in i["message"]["from"]:
          if texto.lower()=="soporte" or texto.lower()=="/soporte":
            notificacion="----------------------------------------\nNuevo Ticket\n\nUsuario: "+nombre+"\n("+msg+")\n\nHa abierto un Ticket de Soport
          notificar(notificacion)
      else:
        texto_respuesta=Memoria.get(texto.lower())
    else:
      texto_respuesta="Lo siento no conozco el termino:\n" + str(texto) + "\n\n¿Necesitas algo?\nDigita AYUDA para ser asistido."

elif tipo == "sticker":
    texto_respuesta = "Su sticker es "+ str(r1) + "% Bonito"
elif tipo == "animacion":
    texto_respuesta = "Me gusta este GIF!"
elif tipo == "foto":
    texto_respuesta = "¿Buscas algo relacionado a la imagen?\nDigita AYUDA para ser asistido."
elif tipo == "otro":
    texto_respuesta = "¿Necesitas algo?\nDigita AYUDA para ser asistido."
```

Todo lo mencionado anteriormente serían las bases del Chat Bot simple, ahora procedemos a la implementación del Reconocimiento de Emociones, por lo que es necesario explicar algunos detalles primero de su funcionamiento.

Capturar Rostros

Antes que nada, aconsejar que este código debe ser ejecutado de manera local en Python o algún otro gestor del lenguaje, ya que Google Colaboratory no permite la creación de ventanas anexas a cv2.

Para poder empezar con la parte que sirve para Detectar emociones, necesitaremos como primer punto, el archivo que contiene el código fuente CapturaRostros.py. Este código lo que hace es activar la cámara, empezar a capturar “n” fotos que nosotros determinamos y las almacena en la carpeta de emoción que corresponde a través de la ruta establecida en la siguiente línea de código. (Se aconseja crear una carpeta “Reconocimiento de Emociones” y dentro otra carpeta llamada “Data”)

```
dataPath = '.../Reconocer Emociones/Data' #Cambia a la ruta donde hayas almacenado Data
emotionsPath = dataPath + '/' + emotionName
```

En el código se encuentran 4 emociones a utilizar y ese es el número de veces que hay que ejecutar el código, pero hay que comentar 3 de las 4 emociones, y dejar 1 para que el código trabaje y almacene las fotos de acuerdo con esa emoción. Luego se comenta la que se utilizó y se le quita el comentario a la siguiente, por lo que se tendría que volver a ejecutar el código para que esta almacene las fotos a esa siguiente emoción.

```
emotionName = 'Enojo'
#emotionName = 'Felicidad'
#emotionName = 'Sorpresa'
#emotionName = 'Tristeza'
```

En la siguiente imagen, se señaló en que sección se define la cantidad de fotos por emoción (en este caso con 200 fotos por emoción). El contador en 0 es el punto inicial, el contador en 200 es el punto final de dicha emoción, por lo que al querer capturar ya sean más fotos por persona o fotos de otra persona se tendría que aumentar el contador final o cambiar el inicial a 200 y el final a 400 respectivamente, con el objetivo de no sobrescribir las fotos tomadas previamente.

```
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
count = 0

while True:

    ret, frame = cap.read()
    if ret == False: break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()

    faces = faceClassif.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
        cv2.imwrite(emotionsPath + '/rotro_{}.jpg'.format(count),rostro)
        count = count + 1
    cv2.imshow('frame',frame)

    k =  cv2.waitKey(1)
    if k == 27 or count >= 200:
        break
```

Modelo Entrenado

Usando el archivo con el código fuente de Entrenando.py. Se va a realizar todo el proceso de entrenamiento del modelo de detección de emociones, primero cargando la data (usando la misma ruta que en la captura de rostros) que se va a utilizar y cargará las emociones que encontró en dicho directorio.

```
dataPath = '/content/drive/My Drive/Data1' #Cambia a la ruta donde hayas almacenado Data
emotionsList = os.listdir(dataPath)
print('Lista de Emociones: ', emotionsList)
```

Luego de que el código se encargue de procesar todos los datos, empezará a entrenar el modelo con lo previo que ya fue procesado, luego de que el modelo haya entrenado con la data cargada se almacenará en la ruta que uno establezca.

```
# Almacenando el modelo obtenido
emotion_recognizer.write("/content/drive/My Drive/ModelosColab/modelo1"+method+".xml")
```

Reconocimiento de Emociones

Una vez realizadas las secciones anteriores significa que tendríamos que tener listo el archivo del Modelo pre-entrenado por lo que se recomienda subir ese archivo a Google Drive para tener ya una ubicación fija en la nube a la cual acceder a través de Google Colaboratory para que funciones sin ningún problema nuestro Chat Bot con Reconocimiento de Emociones.

Para empezar con esta parte, se implementó en el código del Chat Bot la selección del método que se utilizara para el reconocimiento de rostros y emociones.

```
#Deteccion de Emociones (Son los metodos por los cuales se puede identificar las emociones, el mas eficaz a mi parecer el FisherFaces)
#method = 'EigenFaces'
method = 'FisherFaces'
#method = 'LBPH'

if method == 'EigenFaces': emotion_recognizer = cv2.face.EigenFaceRecognizer_create()
if method == 'FisherFaces': emotion_recognizer = cv2.face.FisherFaceRecognizer_create() #Parte del codigo que sirve con el metodo para la deteccion de rostros
if method == 'LBPH': emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()

emotion_recognizer.read('/content/drive/My Drive/Modelos2/modelo'+method+'.xml') #Lectura del modelo pre-entrenado
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
```

Además es necesario crear algunas funciones para la extracción de la imagen que el Chat Bot recibe para posteriormente analizarla con otra función denominada “ReconocerEmoción”.

La función a continuación se encarga de extraer la imagen del servidor de Telegram, esto también se maneja a nivel de Diccionario de Datos, pero para acceder al directorio donde se encuentra la imagen que el Chat Bot recibió por parte de un usuario, la imagen es almacenada en tiempo real en la memoria del Google Colaboratory y con cada imagen nueva se sobrescribe ya que no buscamos almacenar las imágenes recibidas por el Chat Bot.

```
def updateFoto(id):
    #Llamar al metodo getFile del bot
    respuestaF = requests.get(URL + "getFile?file_id=" + str(id))

    #Decodificar la respuesta recibida a formato UTF8
    mensajesF_js = respuestaF.content.decode("utf8")

    #Convertir el string de JSON a un diccionario de Python
    mensajesF_diccionario = json.loads(mensajesF_js)

    #Descargar imagen
    file_path= mensajesF_diccionario["result"]["file_path"]
    imagen = requests.get("https://api.telegram.org/file/bot1086093807:AAH43GWmUoJe5BV6deaS49mEpZUIIi079sE/"+str(file_path)).content
    with open("imagen.jpg",'wb') as handler:
      handler.write(imagen)
```

La siguiente función se encarga de cargar la imagen previamente almacenada para analizarla con el modelo pre-entrenado y así reconocer a que emoción pertenece. (la predicción arroja 2 valores, el primero es para identificar a que emoción pertenece con números enteros del 0 al 3 en este caso y el segundo una medida de precisión que varía según el método que se utilice.)

```
def ReconocerEmocion(idchat):
    image = cv2.imread('/content/imagen.jpg')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    auxFrame=gray.copy()
    faces = faceClassif.detectMultiScale(gray,1.3,5)
```

En esta función también se ponen condicionales que evalúan la cantidad de rostros reconocidos en una foto, para el proyecto se implemento que se reconozca las emociones de un solo rostro por lo que el Reconocimiento de Emociones solo funciona si se ha detectado un solo rostro, si existe una detección múltiple de rostros el algoritmo enviara un mensaje de

advertencia al usuario indicando la cantidad de rostros que se reconocieron. (este mensaje es editable o incluso desechable si así se deseara.)

```
if len(faces) == 0:
  texto="¡Error!\nNo se ha podido reconocer tu rostro, esto podría deberse a:\n1. Condiciones de poca luz.\n2. Baja calidad/resolución.\n3. Uso de al
  requests.get(URL + "sendMessage?chat_id="+ str(idchat) +"&text="+texto)
elif len(faces) == 1:
  for (x,y,w,h) in faces:
      rostro = auxFrame[y:y+h,x:x+w]
      rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
      result = emotion_recognizer.predict(rostro)
      print(str(result))
      if method == 'EigenFaces': P=5700
      if method == 'FisherFaces': P=500
      if method == 'LBPH': P=60
      #if result[0] == 0 and result[1] < P:
      #  texto="Dudoso:\n¿Pensando mucho el día de hoy?"
      if result[0] == 0 and result[1] < P:
        texto="Enojo:\n¿Mucha ira? Respira y tranquilizate un poco."
      elif result[0] == 1 and result[1] < P:
        texto="Felicidad:\nDebe ser un buen día para andar con esa alegría. :D"
      elif result[0] == 2 and result[1] < P:
        texto="Sorpresa:\n¿Sorprendido? Debe ser por algo impactante!"
      elif result[0] == 3 and result[1] < P:
        texto="Tristeza:\n¿Por qué esa cara? Sonríe que en la vida no hay tiempo para estar triste."
      else:
        texto="¡Error!\nAlgunos factores pueden estar impidiendo reconocer tu emoción:\n1. Condiciones de poca luz.\n2. Baja calidad/resolución.\n3.
      requests.get(URL + "sendMessage?chat_id="+ str(idchat) +"&text="+texto)

else:
  texto="¡Advertencia! La detección de emociones no funciona con multiples rostros. (Se detectaron "+str(len(faces))+" para ser exactos)"
  requests.get(URL + "sendMessage?chat_id="+ str(idchat) +"&text="+texto)
```

Y esto seria todo en cuanto a este proyecto, tal como se indica los códigos estarán respectivamente nombrados en el repositorio de GitHub listos para ser ejecutados una vez se cumplan pasos previos como la inserción del Token del Chat Bot, el ID del grupo para recibir las notificaciones, la ruta para las imágenes, el entrenamiento del modelo, la ruta de almacenamiento y demás puntos explicados en este documento.