In [238…

```
#Importación de las librerías a utilizar.
import pyodbc
import re
from telebot import TeleBot
import pandas as pd
import numpy
import json
import random
import pickle
import nltk
import tensorflow
import tflearn
from nltk.stem.lancaster import LancasterStemmer
from PIL import Image
import io
import binascii
import pymssql
import smtplib
import time
from datetime import date
```

CHATBOT creado por OSCAR DUVERNAY 2-18-0818

CHATBOT DE IBC TECHNY

IBC TechnY es una empresa dedicada a la venta y alquiler de equipos electrónicos.

In [ ]:

```
#pip uninstall telebot
#pip uninstall PyTelegramBotAPI
```

In [ ]:

```
#pip install tensorflow
#pip install telebot
#pip install PyTelegramBotAPI==2.2.3
#pip install pytelegrambotapi --upgrade
#pip install tflearn
```

DB Procedimientos

In [239…

```
connection_string="Driver={SQL Server};Server=(local);Database=Empresa2;Trusted_Connection=yes;"

def productos(cod=0):
    con = pyodbc.connect(connection_string)
    stmt = "EXEC [dbo].[productos_data] @id_producto = {id_p}".format(id_p=cod)
    cursor = con.cursor()
    cursor.execute(stmt)
    row = cursor.fetchall()
    products = []
    for f,i in enumerate(row):
        products.append([i[0],i[1],i[2],i[4],i[5]])
    con.close()
    return products
def mispedidos(email):
    con = pyodbc.connect(connection_string)
    stmt = "EXEC [dbo].[Clientes_Pedidos] @email = N'{Email}'".format(Email=email)
    cursor = con.cursor()
    cursor.execute(stmt)
    row = cursor.fetchall()
    pedidos = []
    if(len(row)==0):
        return 'No tiene pedidos activos'
    for f,i in enumerate(row):
        pedidos.append([i[0],i[1],i[2],i[3]])
    mensaje = "\n"

    for m in pedidos:
        mensaje += "Pedido: {ip_p} Tipo: {tipo} \n Fecha: {Fecha} Estado: {esta}\n\n".format(ip_p=m[0],tipo=m[2
    con.close()
    return mensaje

def ActualizarClientes(id_cliente,nombre,apellidos,email,sexo,direccion,fech_naci,telefono):
    con = pyodbc.connect(connection_string)
    stmt= "EXEC [dbo].[ActClientes] @id_cliente = {id_cliente}, @nombre = N'{nombre}', @apellido = N'{apellido}
    cursor = con.cursor()
    cursor.execute(stmt)
    row = cursor.fetchall()
    con.commit()
    con.close()
    if(len(row)==0):
```

```
        return 0
    else:
        return row[0][0]


def ConfirmClient(email):
    con = pyodbc.connect(connection_string)
    stmt= "EXEC [dbo].[Cliente_confirmacion] @email = N'{Email}'".format(Email=email)
    cursor = con.cursor()
    cursor.execute(stmt)
    row = cursor.fetchall()
    con.close()
    if(len(row)==0):
        return 0
    else:
        return row[0][0]

def ActualizarPedidos(id_pedido,id_cliente,total,id_estado,id_tipo):
    fecha=date.today()
    con = pyodbc.connect(connection_string)
    stmt= "DECLARE @tmp DATETIME SET @tmp = GETDATE() EXEC [dbo].[ActPedidos] @id_pedido = {id_pedido}, @id_cli
    cursor = con.cursor()
    cursor.execute(stmt)
    row = cursor.fetchall()
    con.commit()
    con.close()
    if(len(row)==0):
        return 0
    else:
        return row[0]

def ActualizarDetalle(id_pedido,productos,cantidad):
    con = pyodbc.connect(connection_string)
    cursor = con.cursor()
    stmt= "Exec [dbo].[Insert_Detalle_Pedido] @id_pedido = {id_pedido}, @id_producto = {productos}, @cantidad =
    cursor.execute(stmt)
    con.commit()
    con.close()


def ActualizarReservas(id_pedido,fecha):
    con = pyodbc.connect(connection_string)
    stmt= "EXEC [dbo].[ActPedidos_Reserva] @id_pedido = {id_pedido}, @Fecha = N'{Fecha}' ".format(id_pedido=id_
    cursor = con.cursor()
    cursor.execute(stmt)
    con.commit()
    con.close()
def ActualizarPedidosAdomicilio(id_pedido,direccion):
    con = pyodbc.connect(connection_string)
    stmt= "EXEC [dbo].[ActPedidos_Adomicilio] @id_pedido = {id_pedido}, @direccion = '{direccion}'".format(id_p
    cursor = con.cursor()
    cursor.execute(stmt)
    con.commit()
    con.close()



def CancelarPedido(id_pedido):
    con = pyodbc.connect(connection_string)
    stmt= "EXEC   [dbo].[Cancel_Pedido]   @id_pedido = {id_pedido}".format(id_pedido=id_pedido)
    cursor = con.cursor()
    cursor.execute(stmt)
    con.commit()
    con.close()
```

In [240…

```
#print(mispedidos('lineess70@gmail.com'))
#print(productos())
#print(ActualizarPedidos(0,3,304,1,1)[0])
#datetime.now().strftime("%d/%m/%Y %H:%M:%S")
```

Telegram bot

In [241…

```
def Enviar_Correo(name,correo,cedula,producto,cantidad):
    with smtplib.SMTP_SSL('smtp.gmail.com',465) as smtp:

        smtp.login('@gmail.com','Plomo1500')

        subject = 'Reservacion!'
        body = f'''Su reservacion se realizo con exito con los siguientes datos:\n\n
        Nombre:{name}\nCedula:{cedula}\nProducto reservado:{producto}\nCantidad:{cantidad}
        \n\nTiene un plazo de 5 dias para pasar por su producto, de lo contrario la reservacion sera cancelada.
        P.S:Debe presentar su cedula a la hora de recoger su producto, tambien si desea cancelar su reservacion
        un plazo de 4 horas, sino no se podra cancelar.'''
        msg = f'Subject: {subject}\n\n{body}'
        smtp.sendmail('@gmail.com',correo,msg)
```

In [242…

```
def contenido():
    dat = {
        "contenido": [
             {
                "tag": "diferente",
                "patrones": [
                   " "," "
                ],
                "respuestas": [
                    "Disculpe, no le pude entender.",
                    "No se le entiende porfavor repita.",
                    "Porfavor repitame."
                ]
            },
            {
                "tag": "despedida",
                "patrones": [
                    "adios",
                    "hasta la proxima",
                    "nos vemos",
                    "Cuidate"

                ],
                "respuestas": [
                    "Cuidate mucho",
                    "Hasta luego",
                    "Te estaré esperando",
                    "Suerte"
                ]
            },
             {
                "tag": "saludo",
                "patrones": [
                    "hola",
                    "hello",
                    "saludos",
                    "que tal",
                    "como estas","klk","buenas"
                ],
                "respuestas": ["Saludos en que le puedo ayudar","Buenas, como lo puedo ayudar","Bienvenido, en 

            },
             {"tag":"productos",
              "patrones":[
                  "prodCuuctos",
                  "articulos",
                  "cuales productos venden",
                  "muestreme el catalogo",
                  "que productos venden",
                  "venden"
                ],

               "respuestas":["Esta son las articulos que tenemos por el momento: {res}".format(res=productos())
           },

            {
                "tag": "direccion",
                "patrones": [
                    "donde estan ubicados",
                    "como puedo contactarlos",
                    "cual es su direccion","donde se encuentran"
                ],
                "respuestas": ["Actualmente estamos ubicados en la avenida estrella sadahala #20, puedes contac
            },
          {
                "tag": "gracias",
                "patrones": [
                    "gracias ",
                    "Muchas gracias",
                    "Gracias por todo", "gracia"
                ],
                "respuestas": ["Estamos para servirle","A la orden","De nada","No hay de que"]
            },
             {
                "tag": "reservar",
                "patrones": [
                    "quisiera realizar una reservacion",
                    "quiero hacer una reserva",
                    "quiero apartar",
                ],
                "respuestas": ["Su reservacion se ha procesado exitosamente: {res}"]
            },
            {
                "tag": "cancelar",
                "patrones": [
                    "quiero cancelar"
                    ,"cancelar"
                    ,"cancelame mi reservacion"
```

```
                    ,"ya no quiero",
                    "quiero cancelar un pedido"
                ],
                "respuestas": [""]
            },
            {
                "tag": "estado",
                "patrones": [
                    "cual es el estado",
                    "como esta mi reservacion","digame de mi",
                    "como va",
                    "mis reservas",
                    "mis pedidos"

                ],
                "respuestas": ["Este es el estado de sus pedidos:\n {res}"]
            },
             {
                "tag": "Tomas",
                "patrones": [
                    "El profe",
                    "El profesor",
                    "El teacher"
                ],
                "respuestas": ["Es duro en COD y nos va a pasar en A (:-D)!"]
            }
        ]
    }
    return dat
```

In [243…

```
#proceso de token a telegram
stemmer = LancasterStemmer()
bot = TeleBot("5056785184:AAH2G9Ha8tiaqww93GtQc0tZ9U_EQAoi98o")
```

Objets

In [244…

```
class Pedido:
    def __init__(self,name):
        self.name = name
        self.email = None
        self.indice = None
        self.producto = None
        self.cantidad = None
        self.total=None
        self.tipo=None
        self.direccion = None
        self.fech_reserv = None
class Cliente:
    def __init__(self,name):
        self.name=name
        self.id_cliente=None
        self.nombre=None
        self.apellidos=None
        self.email=None
        self.sexo=None
        self.direccion=None
        self.fech_naci=None
        self.telefono=None
class cancelar:
    def __init__(self,cedula):
        self.id_pedido=None
class Monitorear:
    def __init__(self,email):
        self.correo=email
```

Mini Objeto usado en las funciones para escribir las respuestas

In [245…

```
datos=contenido()
```

Monitoreo de pedidos del cliente

In [246…

```
def monitoreo(message):
    msg=bot.reply_to(message,"Digame su email")
    bot.register_next_step_handler(msg,monitorear)
def monitorear(message):
    correo=message.text
    bot.reply_to(message,tagsearch_res("estado").format(res=mispedidos(correo)))
```

Registrar cliente en el sistema

In [247…

```
def regis_email(message):
    if(ConfirmClient(message.text)==0):
        Cliente.email=message.text
        bot.reply_to(message,"Cual es su nombre")
        bot.register_next_step_handler(message,regis_name)
    else:
        bot.reply_to(message,"Este email ya esta registrado, favor digitar un email valido")
        bot.register_next_step_handler(message,regis_email)

def regis_name(message):
    Cliente.nombre=message.text
    bot.reply_to(message,"Digame sus apellidos")
    bot.register_next_step_handler(message,regis_last_name)
def regis_last_name(message):
    Cliente.apellidos=message.text
    bot.reply_to(message,"Usted es M/F?")
    bot.register_next_step_handler(message,regis_sex)
def regis_sex(message):
    Cliente.sexo=message.text
    bot.reply_to(message,"Denos su direccion")
    bot.register_next_step_handler(message,regis_direccion)
def regis_direccion(message):
    Cliente.direccion=message.text
    bot.reply_to(message,"Fecha de nacimiento")
    bot.register_next_step_handler(message,regis_fech_naci)
def regis_fech_naci(message):
    Cliente.fech_naci=message.text
    bot.reply_to(message,"Digite su numero de contacto")
    bot.register_next_step_handler(message,registro)
def registro(message):
    Cliente.telefono=message.text
    Cliente.id_cliente=0
    bot.reply_to(message,"Un momento por favor")
    Cliente.id_cliente=ActualizarClientes(Cliente.id_cliente,Cliente.nombre,Cliente.apellidos,Cliente.email,Cli
    bot.reply_to(message,"Registro completo")
    bot.reply_to(message,"Que desea hacer?")
    return
```

Registro conditionals

Estas se lanzan o a la hora de hacer una reservacion, o a la hora de hacer un pedido adomicilio antes de proceder a realizar el pedido, su objetivo es saber si el cliente esta registrado, de no estarlo podra registrarse, de estarlo, entonces podra proseguir.

In [248…

```
def pedir_condition(message):
    cond=message.text
    if(cond.lower()=='si'or cond.lower()=='s'):
        bot.reply_to(message,"Digame su email")
        bot.register_next_step_handler(message,regis_email)
    elif(cond.lower()=='no'or cond.lower()=='n'):
        Pedido.id_pedido=0
        Pedido.indice=0
        Pedido.total=0
        Pedido.tipo=0
        Pedido.producto=[]
        Pedido.cantidad=[]
        bot.reply_to(message, "Digame su email")
        bot.register_next_step_handler(message,pedir_email)
    else:
        bot.reply_to(message,"Por favor, escriba correctamente")
        bot.register_next_step_handler(msg,pedir)
def pedir_email(message):
    if(ConfirmClient(message.text)!=0):
        Cliente.id_cliente=ConfirmClient(message.text)
        bot.reply_to(message,tagsearch_res('productos'))
        bot.reply_to(message,"Escriba el numero del producto que desea comprar")
        bot.register_next_step_handler(message,pedir_producto)
    else:
        msg=bot.reply_to(message,"Este email no esta registrado, favor digitar un email valido")
        bot.register_next_step_handler(msg,regis_email)
```

Realizar reservacion

In [249…

```

def pedir(message):
    msg=bot.reply_to(message,"Es nuevo en este sitio? \n Si/No")
    bot.register_next_step_handler(message,pedir_condition)

def pedir_producto(message):
    if(message.text.isnumeric()):
        if(product_exist(message.text)):
            Pedido.producto.append(int(message.text))
```

```
            bot.reply_to(message,"Que cantidad?")
            bot.register_next_step_handler(message,pedir_cantidad)
        else:
            bot.reply_to(message, "Este valor no corresponde a ningun producto")
            bot.register_next_step_handler(message,pedir_producto)
    else:
        bot.reply_to(message,"Ingrese un numero")
        bot.register_next_step_handler(message,pedir_producto)
def pedir_cantidad(message):
    if(message.text.isnumeric()):
        if(inrange(message.text,Pedido.producto[Pedido.indice])):
            Pedido.cantidad.append(int(message.text))
            Pedido.total=total_resize(Pedido.producto,Pedido.cantidad,Pedido.indice)
            bot.reply_to(message,"Su total es de: ${num} \n Desea pedir otro producto? Si/No".format(num=Pedido
            bot.register_next_step_handler(message,pedir_inter)
        else:
            bot.reply_to(message, "Ha excedido la cantidad de producto.\n Debe ser menor a {res}".format(res=pr
            bot.register_next_step_handler(message,pedir_producto)
    else:
        bot.reply_to(message,"Ingrese un numero")
def pedir_inter(message):
    cond=message.text
    if(cond.lower()=='si'or cond.lower()=='s'):
        Pedido.indice+=1
        bot.reply_to(message,tagsearch_res('productos'))
        bot.reply_to(message,"Escriba el numero del producto que desea comprar")
        bot.register_next_step_handler(message,pedir_producto)
    elif(cond.lower()=='no'or cond.lower()=='n'):
        msg=bot.reply_to(message,"Quiere hacer una reserva o entrega a domicilio?")
        bot.register_next_step_handler(msg,pedir_tipo)
    else:
        bot.reply_to(message,"Ingrese una opcion valida")
        bot.register_next_step_handler(message,pedir_inter)

def pedir_tipo(message):
    pedir_tipo_decide(message)
    if(Pedido.tipo==1):
        bot.reply_to(message,"Para cuando desea el pedido?")
        bot.register_next_step_handler(message,reservar)
    elif(Pedido.tipo==2):
        bot.reply_to(message,"Donde desea que llegue la entrega?")
        bot.register_next_step_handler(message,adomicilio)
    else:
        bot.reply_to(message,"Por favor ingrese un valor valido")
        bot.register_next_step_handler(message,pedir_tipo_decide)

def pedir_tipo_decide(message):
    cond=message.text
    if(cond.lower()=='reserva' or cond.lower()=='una reserva' or cond.lower()=='una reservacion' or cond.lower(
        Pedido.tipo=1
    elif(cond.lower()=='entrega' or cond.lower()=='a domicilio' or cond.lower()=='entrega a domicilio'or cond.l
        Pedido.tipo=2
    else:
        bot.reply_to(message,"Por favor, ingrese un valor valido")
        bot.register_next_step_handler(message,pedir_tipo_decide)
    return

def reservar(message):
    Pedido.fech_reserv = message.text
    id_pedido=ActualizarPedidos(Pedido.id_pedido,Cliente.id_cliente,Pedido.total,1,Pedido.tipo)[0]
    ActualizarReservas(id_pedido,Pedido.fech_reserv)

    for cont in range(0,Pedido.indice):
        ActualizarDetalle(id_pedido,Pedido.producto[cont],Pedido.cantidad[cont])
    bot.reply_to(message, "Su pedido se proceso con exito")

def adomicilio(message):
    Pedido.direccion = message.text
    id_pedido=ActualizarPedidos(Pedido.id_pedido,Cliente.id_cliente,Pedido.total,1,Pedido.tipo)[0]
    ActualizarPedidosAdomicilio(id_pedido,Pedido.direccion)
    for cont in range(0,Pedido.indice):
        ActualizarDetalle(id_pedido,Pedido.producto[cont],Pedido.cantidad[cont])
    bot.reply_to(message, "Su pedido se proceso con exito")
```

Cancelacion

In [250…

```
def can(message):
    monitoreo(message)
    bot.register_next_step_handler(message,cancelar)

def cancelar(message):
    bot.reply_to(message,"Cual pedido desea cancelar?")
    bot.register_next_step_handler(message,cancel)
def cancel(message):
    if(message.text.isnumeric()):
```

```
        Pedido.id_pedido=int(message.text)
        bot.reply_to(message,"Estas seguro? Si/No")
        bot.register_next_step_handler(message,cancelacion)
    else:
        bot.reply_to(message, "Introduce un valor real")
        bot.register_next_step_handler(message,cancelar)
    return
def cancelacion(message):
    cond=message.text
    if(cond.lower()=='si'or cond.lower()=='s'):
        CancelarPedido(Pedido.id_pedido)
        bot.reply_to(message,"Se ha cancelado el pedido {res}".format(res=Pedido.id_pedido))
        return
    elif(cond.lower()=='no'or cond.lower()=='n'):
        msg=bot.reply_to(message,"Gracias por su tiempo")
        return
    else:
        bot.reply_to(message,"Ingrese una opcion valida")
        bot.register_next_step_handler(message,cancelacion)
```

Funciones auxiliares

Funciones utilizadas para tareas especificas en los bots

In [251…

```
def product_exist(i):
    a=int(i)
    products=productos()
    for x in products:
        if(x[0]==a):
            return True
    return False
def inrange(c,i):
    a=int(c)
    product=productos(i)
    if(a<=product[0][3]):
        return True
    else:
        return False
def total_resize(product,cantidad,indice):
    total=0
    for num in range(0,indice+1):
        produc=productos(product[num])
        total+=produc[0][4]*cantidad[num]
    return total
```

Bot

In [ ]:

```
palabras=[]
tags=[]
auxX=[]
auxY=[]

#Aqui llamamos a nuestro diccionario de palabras para que responda al usuario
for contenido in datos["contenido"]:
    for patrones in contenido["patrones"]:
        auxPalabra =nltk.word_tokenize(patrones)
        palabras.extend(auxPalabra)
        auxX.append(auxPalabra)
        auxY.append(contenido["tag"])

        if contenido["tag"] not in tags:
            tags.append(contenido["tag"])


palabras =[stemmer.stem(w.lower()) for w in palabras if w!="?"]
palabras=sorted(list(set(palabras)))
tags=sorted(tags)

entrenamiento=[]
salida=[]

salidaVacia=[0 for _ in range(len(tags))]



for x,documento in enumerate(auxX):
    cubeta=[]
    auxPalabra=[stemmer.stem(w.lower()) for w in documento]
    for w in palabras:
        if w in auxPalabra:
            cubeta.append(1)
        else:
            cubeta.append(0)
```

```
    filaSalida= salidaVacia[:]
    filaSalida[tags.index(auxY[x])]=1
    entrenamiento.append(cubeta)
    salida.append(filaSalida)

entrenamiento=numpy.array(entrenamiento)
salida=numpy.array(salida)


tensorflow.compat.v1.reset_default_graph()

red= tflearn.input_data(shape=[None,len(entrenamiento[0])])
red= tflearn.fully_connected(red,10)
red= tflearn.fully_connected(red,10)

red= tflearn.fully_connected(red,len(salida[0]), activation="softmax")
red = tflearn.regression(red)

modelo = tflearn.DNN(red)
modelo.fit(entrenamiento,salida,n_epoch=1000, batch_size=11, show_metric=True)
modelo.save("modelo.tflearn")



res_dict = {}
can_dict = {}


@bot.message_handler(func=lambda m: True)
def echo_all(message):
    entrada = message.text
    cubeta=[0 for _ in range(len(palabras))]
    entradaPro=nltk.word_tokenize(entrada)
    entradaPro=[stemmer.stem(palabra.lower()) for palabra in entradaPro]
    for palabraInd in entradaPro:
        for i, palabra in enumerate(palabras):
            if palabra == palabraInd:
                cubeta[i]=1
    resultados = modelo.predict([numpy.array(cubeta)])
    resultadosIndi=numpy.argmax(resultados)
    tag=tags[resultadosIndi]

    res=[]
    if tag=='productos':
        bot.reply_to(message,tagsearch_res(tag))
    elif tag=='cancelar':
        can(message)
    elif tag=='estado':
        res = monitoreo(message)
    elif tag=='reservar':
        res = pedir(message)
    else:
        bot.reply_to(message,tagsearch_res(tag))

def tagsearch_res(tag):
    for tagaux in datos["contenido"]:
        if tagaux["tag"]==tag:
            print(tagaux)
            return random.choice(tagaux["respuestas"])
bot.polling()
```

```
Training Step: 3999  | total loss: 0.02912 | time: 0.010s
| Adam | epoch: 1000 | loss: 0.02912 - acc: 0.9995 -- iter: 33/44
Training Step: 4000  | total loss: 0.02845 | time: 0.012s
| Adam | epoch: 1000 | loss: 0.02845 - acc: 0.9996 -- iter: 44/44
--
INFO:tensorflow:C:\Users\santo\Bots\modelo.tflearn is not in all_model_checkpoint_paths. Manually adding it.
{'tag': 'saludo', 'patrones': ['hola', 'hello', 'saludos', 'que tal', 'como estas', 'klk', 'buenas'], 'respuest
as': ['Saludos en que le puedo ayudar', 'Buenas, como lo puedo ayudar', 'Bienvenido, en que le puedo servir']}
{'tag': 'estado', 'patrones': ['cual es el estado', 'como esta mi reservacion', 'digame de mi', 'como va', 'mis
reservas', 'mis pedidos'], 'respuestas': ['Este es el estado de sus pedidos:\n {res}']}
{'tag': 'estado', 'patrones': ['cual es el estado', 'como esta mi reservacion', 'digame de mi', 'como va', 'mis
reservas', 'mis pedidos'], 'respuestas': ['Este es el estado de sus pedidos:\n {res}']}
{'tag': 'despedida', 'patrones': ['adios', 'hasta la proxima', 'nos vemos', 'Cuidate'], 'respuestas': ['Cuidate
mucho', 'Hasta luego', 'Te estaré esperando', 'Suerte']}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js