

CHAT BOT

INTEGRANTES:

ORTIGOZA MARTÍNEZ HUITZIL

IBARRAZA RIVERA LIZETH

SÁNCHEZ SÁNCHEZ JOSÉ GABRIEL

ROSALES MARTÍNEZ AARON

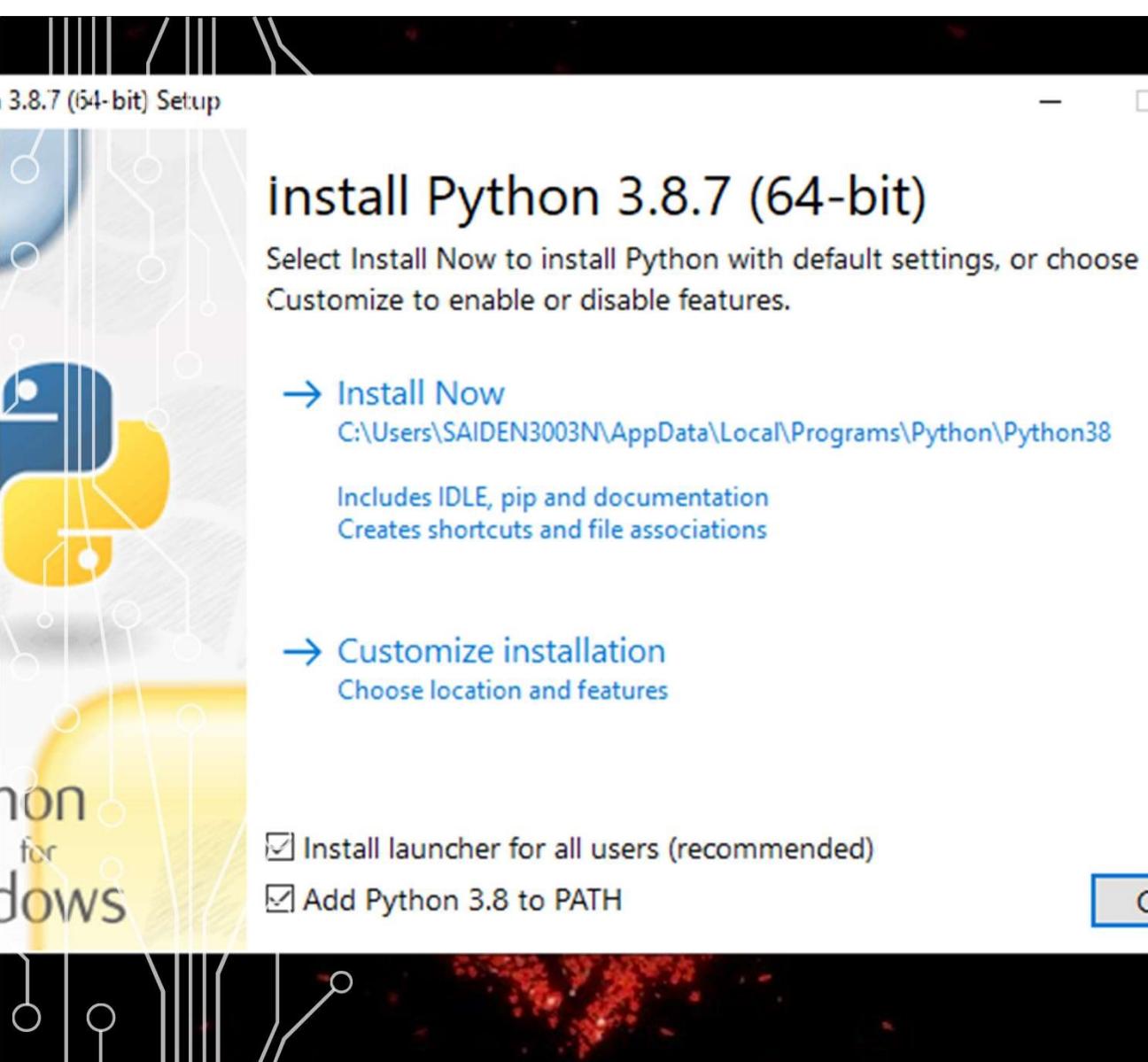


HERRAMIENTAS

Python 3.8

Anaconda 3

INSTALACIÓN DE LAS HERRAMIENTAS

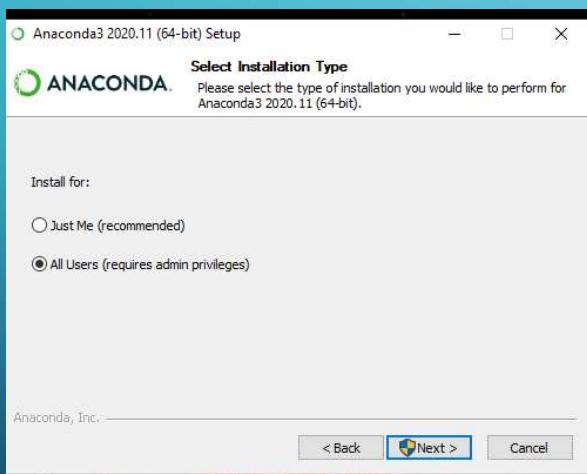


PYTHON

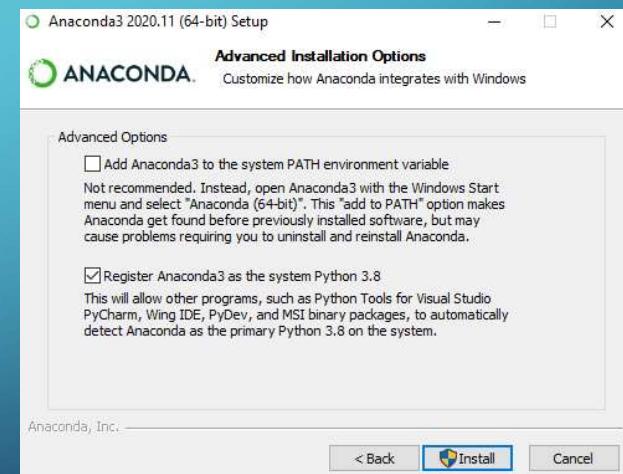
- Para instalar Python por el archivo ejecutable que se descarga de su pagina oficial.
- se seleccionaría las dos opciones agregando al path

ANACONDA

SE LE OTORGARA PERMISOS DE
USUARIOS



AL IGUAL QUE PYTHON SE
AGREGARA AL PATH



CREACIÓN DEL ENTORNO Y DESCARGA DE LIBRERÍAS

PARA LA CREACIÓN DEL ENTORNO DE DESARROLLO SE USA LA LÍNEA CONDA CRÉATE -N CHAT PYTHON 3.8

```
PS C:\WINDOWS\system32\cmd.exe - conda create -n chat python=3.8

C:\Users\SAIDEN3003N\conda create -n chat python=3.8
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\SAIDEN3003N\.conda\envs\chat

added / updated specs:
- python=3.8

The following NEW packages will be INSTALLED:

ca-certificates      pkgs/main/win-64::ca-certificates-2021.1.19-haa95532_0
certifi               pkgs/main/win-64::certifi-2020.12.5-py38haa95532_0
openssl              pkgs/main/win-64::openssl-1.1.1i-h2bbff1b_0
pip                  pkgs/main/win-64::pip-20.3.3-py38haa95532_0
python               pkgs/main/win-64::python-3.8.5-h5fd99cc_1
setuptools            pkgs/main/win-64::setuptools-51.3.3-py38haa95532_4
sqlite               pkgs/main/win-64::sqlite-3.33.0-h2a8f88b_0
vc                   pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime        pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel                pkgs/main/noarch::wheel-0.36.2-pyhd3eb1b0_0
wincertstore         pkgs/main/win-64::wincertstore-0.2-py38_0
zlib                 pkgs/main/win-64::zlib-1.2.11-h62cd97_4

Proceed ([y]/n)? y
```

SE ACEPTARA EL PROCESO DE INSTALACIÓN

```
Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate chat
#
# To deactivate an active environment, use
#
#     $ conda deactivate

C:\Users\SAIDEN3003N>
```

Se instalaran las librerías tensorflow, numpy, tflearn y nltk.
para hacerlo se necesita entrar al entorno con la línea:

Activate chat

Y se descargaran las librerías con: pip install tensorflow, o el nombre de la
librería que se desea descargar

```
(chat) C:\Users\SAIDEN3003N>pip install nltk
Collecting nltk
  Using cached nltk-3.5-py3-none-any.whl
Collecting click
  Using cached click-7.1.2-py2.py3-none-any.whl (82 kB)
Collecting joblib
  Using cached joblib-1.0.0-py3-none-any.whl (302 kB)
Collecting regex
  Using cached regex-2020.11.13-cp38-cp38-win_amd64.whl (270 kB)
Collecting tqdm
  Using cached tqdm-4.56.0-py2.py3-none-any.whl (72 kB)
Installing collected packages: tqdm, regex, joblib, click, nltk
Successfully installed click-7.1.2 joblib-1.0.0 nltk-3.5 regex-2020.11.13 tqdm-4.56.0

(chat) C:\Users\SAIDEN3003N>pip install tensorflow
Collecting tensorflow
  Using cached tensorflow-2.4.1-cp38-cp38-win_amd64.whl (370.7 kB)
Requirement already satisfied: wheel~=0.35 in c:\users\saiden3003n\.conda\envs\chat\lib\site-packages (from tensorflow)
(0.36.2)
Collecting gast==0.3.3
  Using cached gast-0.3.3-py2.py3-none-any.whl (9.7 kB)
Collecting absl-py~=0.10
  Using cached absl_py-0.11.0-py3-none-any.whl (127 kB)
Collecting astunparse==1.6.3
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
```

```
(chat) C:\Users\SAIDEN3003N>pip install numpy
Requirement already satisfied: numpy in c:\users\saiden3003n\.conda\envs\chat\lib\site-packages (from tensorflow)
(1.21.2)
Collecting tflearn
  Using cached tflearn-0.5.0-py3-none-any.whl
Requirement already satisfied: six in c:\users\saiden3003n\.conda\envs\chat\lib\site-packages (from tensorflow)
Requirement already satisfied: numpy in c:\users\saiden3003n\.conda\envs\chat\lib\site-packages (from tensorflow)
Collecting Pillow
  Using cached Pillow-8.1.0-cp38-cp38-win_amd64.whl (2.2 MB)
Installing collected packages: Pillow, tflearn
Successfully installed Pillow-8.1.0 tflearn-0.5.0

(chat) C:\Users\SAIDEN3003N>
```

ARCHIVO .JSON

```
3     "patrones":["hola","un saludo","hello","buenos dias"],  
4     "respuestas":["hola que tal, en que puedo ayudarte","Comó te va","un gusto de verte","buenos dias, en que puedo ayudarte"]  
5 },  
6 {"tag":"postre",  
7     "patrones":["postre","que postre tienes","que hay de postre","que postres hay"],  
8     "respuestas":["pastel de chocolate, panseillos, flan, helado, arroz con leche"]  
9 },  
10  
11 {"tag":"despedida",  
12     "patrones":["adios","hasta luego","nos vemos","hasta la proxima"],  
13     "respuestas":["cuidate","adios", "nos vemos pronto", "te estare esperando"]  
14 }  
15 },  
16 {"tag":"abierto",  
17     "patrones":["abierto","abren","a que hora abren","desde a que hora estan abiertos","a que hora aceptan clientes"],  
18     "respuestas":["El restaurante abre a las 10 am","aceptamos clientes desde las 10am"]  
19 },  
20 },  
21 {"tag":"sin",  
22     "patrones":["sin reservacion","aceptan clientes sin reservacion"],  
23     "respuestas":["solo se acepta clientes con reservacion","no se aceptan clientes sin reservacion, deberias hacer una reservacion"]  
24 }  
25 }, en este archivo se alojara el  
26 {"tag": "contenido" } que consta de:  
27     "patrones":["cerrado","cierran","a que hora cierran"],  
28     "respuestas":["El restaurante cierra a las 12 pm, pero dejamos de recibir clientes con reservacion a las 12pm"]  
29 },  
30 },  
31 {"tag":"personas",  
32     "patrones":["cuantas personas son en la reservacion","cuantas personas por reservacion","cuantos comensales pueden ir en una reservacion"],  
33     "respuestas":["son 5 personas maximo","5 personas por reservacion","en una reservacion pueden ir 5 comensales"]  
34 },  
35 },  
36 {"tag":"reservacion",  
37     "patrones":["cuantas veces puedo reservar","puedo hacer mas de una reservacion","puedo hacer mas reservaciones"],  
38     "respuestas":["solo puedes hacer una reservacion por persona","si ya hiciste una reservacion ya no puedes hacer otra"]  
39 }.
```

IMPORTAR LAS LIBRERIAS

- La librería nltk permitirá el procesamiento de lenguaje natural.
- Nltk.stem.Lancaster esto permitirá transformar las palabras, para que sean mas entendibles para el chatbot
- Json que ahí se tendrá la información
- Randon para las respuestas aleatorias
- Pickle para guardar el proyecto

```
📁 mainbot.py > ...
1 import nltk
2 from nltk.stem.lancaster import LancasterStemmer
3 stemmer = LancasterStemmer()
4 import numpy
5 import tflearn
6 import tensorflow
7 import json
8 import random
9 import pickle
10
```

SE MANDA A LLAMAR AL ARCHIVO .JSON

```
mainbot.py > ...
1  import nltk
2  from nltk.stem.lancaster import LancasterStemmer
3  stemmer = LancasterStemmer()
4  import numpy
5  import tflearn
6  import tensorflow
7  import json
8  import random
9  import pickle
10
11 #nltk.download('punkt')
12 with open("contenido.json") as archivo:
13     datos = json.load(archivo)
14     print(datos)
```

Y EL RESULTADO ES TODA LA INFORMACIÓN DE EL

```
{'contenido': [{tag: 'saludo', 'patrones': ['hola', 'un saludo', 'hello', 'buenos dias'], 'respuestas': ['hola que tal', 'en que puedo ayudarte', 'ComAº te va', 'un gusto de verte', 'buenos dias, en que puedo ayudarte']}, {tag: 'postre', 'patrones': ['postre', 'que postre tienes', 'que hay de postre', 'que pontrres hay'], 'respuestas': ['pastel de chocolate', 'panseillos, flan, helado,arroz con leche']}, {tag: 'despedida', 'patrones': ['adios', 'hasta luego', 'nos vemos', 'hasta la proxima'], 'respuestas': ['cuidate', 'adios', 'nos vemos pronto', 'te estare esperando']}, {tag: 'abrierto', 'patrones': ['abierto', 'abren', 'a que hora abren', 'desde a que hora estan abiertos', 'a que hora aceptan clientes'], 'respuestas': ['El restaurante abre a las 10 am', 'aceptamos clientes desde las 10am']}, {tag: 'sin', 'patrones': ['sin reservacion', 'aceptan clientes sin reservacion'], 'respuestas': ['solo se acepta clientes con reservacion', 'no se aceptan clientes sin reservacion, deberias hacer una reservacion']}, {tag: 'cerrado', 'patrones': ['cerrado', 'cierran', 'a que hora cierran'], 'respuestas': ['El restaurante cierra a las 12 pm, pero dejamos de recibir clientes con reservacion a las 11pm']}, {tag: 'personas', 'patrones': ['cuantas personas son en la reservacion', 'cuantas personas por reservacion', 'cuantos comensales pueden ir en una reservacion'], 'respuestas': ['son 5 personas maximo', '5 personas por reservacion', 'en una reservacion pueden ir 5 comensales']}, {tag: 'reservacion', 'patrones': ['cuantas veces puedo reservar', 'puedo hacer mas de una reservacion', 'puedo hacer mas reservaciones'], 'respuestas': ['solo puedes hacer una reservacion por persona', 'si ya hiciste una reservacion ya no puedes hacer otra']}, {tag: 'niAtos', 'patrones': ['pueden entrar niAtos', 'pueden entrar bebes', 'cuantos niAtos pueden ir por reservacion'], 'respuestas': ['se admiten niAtos, PAGAN DOBLE, solo toma encuenta el menu del restaurante', 'acuerdate que solo son 5 personas maximo, contanto a los niAtos']}, {tag: 'estas', 'patrones': ['como estas', 'hola como estas', 'que tal estas'], 'respuestas': ['no lo se, tu dime']}]}
```

(chat) C:\Users\SAIDEN3003N\chat>

se creara un for para entrar al contenido en los tags para los patrones y acceder a ellos, el contenido de ello se almacenara en auxpalabra, junto con nltk.word_tokenize para reconocer los signos especiales.

En auxY se guardaran los tags repetido para pasarlo a individual se usa un if.

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

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"])

print(palabras)
print(auxX)
print(auxY)
print(tags)
```

```
[]
[['hola'], ['un', 'saludo'], ['hello'], ['buenos', 'dias'], ['postre'], ['que', 'postre', 'tiempos'], ['que', 'hay', 'de', 'postre'], ['que', 'pontres', 'hay'], ['adios'], ['hasta', 'luego'], ['nos', 'vemos'], ['hasta', 'la', 'proxima'], ['abierto'], ['abren'], ['a', 'que', 'hora', 'abren'], ['desde', 'a', 'que', 'hora', 'estan', 'abiertos'], ['a', 'que', 'hora', 'aceptan', 'clientes'], ['sin', 'reservacion'], ['aceptan', 'clientes', 'sin', 'reservacion'], ['cerrado'], ['cierran'], ['a', 'que', 'hora', 'cierran'], ['cuantas', 'personas', 'son', 'en', 'la', 'reservacion'], ['cuantas', 'personas', 'por', 'reservacion'], ['cuantos', 'comensales', 'pueden', 'ir', 'en', 'una', 'reservacion'], ['cuantas', 'veces', 'puedo', 'reservar'], ['puedo', 'hacer', 'mas', 'de', 'una', 'reservacion'], ['puedo', 'hacer', 'mas', 'reservaciones'], ['pueden', 'entrar', 'niños'], ['pueden', 'entrar', 'bebés'], ['cuantos', 'niños', 'pueden', 'ir', 'por', 'reservacion'], ['como', 'estas'], ['hola', 'como', 'estas'], ['que', 'tal', 'estas']]
['saludo', 'saludo', 'saludo', 'postre', 'postre', 'postre', 'despedida', 'despedida', 'despedida', 'despedida', 'abierto', 'abierto', 'abierto', 'abierto', 'abierto', 'sin', 'sin', 'cerrado', 'cerrado', 'cerrado', 'personas', 'personas', 'personas', 'reservacion', 'reservacion', 'reservacion', 'reservacion', 'niños', 'niños', 'niños', 'estas', 'estas']
['saludo', 'postre', 'despedida', 'abierto', 'sin', 'cerrado', 'personas', 'reservacion', 'niños', 'estas']
```

Se usara la línea:

```
palabras = [stemmer.stem(w.lower())  
for w in palabras if w!="?"]
```

Para pasar la palabra en minúscula
siempre y cuando la palabra sea
diferente a ?

Se ordenara las palabras y los tags

```
palabras = sorted(list(set(palabras)))
```

```
tags = sorted(tags)
```

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

Para entrenar al bot, se hace uso de una cubeta, par esto se usara un for, el cual enumerara las palabras por un índice guardado en x

En el for w, si esta en palabras se añadirá en cubeta, 1 si esta 0 si no esta.

La fila salida ayudara para obtener el índice de y para asignarle el valor de 1

En entrenamiento se agregara la cubeta

```
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)
print(entrenamiento)
print(salida)
```

\chat>

El resultado es una lista de listas, se muestra los 1 donde se encontró la palabra encontrada

Se pasa las listas en arreglos con numpy

Con tensorflow creamos que se rince todo

En la red se creara una red, con forma shape y la longitud,

Se crea los hidden layers (5), los cuales tendrán las neuronas que tendrán, al igual se creara una red para la longitud de la salida que será 0, esta sera la salida

Con tflearn.regression se obtendrán probabilidades para saber que tag nos referimos

```
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,80)
red = tflearn.fully_connected(red,80,activation="softmax")
red = tflearn.regression(red)
```

Se creara un modelo.fit que ayudara a entrenar a la red, con n_epoch será las veces que vea la información, batch_size nos ayudara para saber cuantas entradas tenemos que va a las palabras de los patrones.

Se guardara el modelo en un archivo.tflearn

```
modelo = tflearn.DNN(red)
modelo .fit(entrenamiento,salida,n_epoch=10000,batch_size=84,show_metric=True)
modelo.save("modelo.tflearn")
```

CÓDIGO

```
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, 80)
red = tflearn.fully_connected(red, len(salida[0]), activation="softmax")
red = tflearn.regression(red)

modelo = tflearn.DNN(red)
modelo .fit(entrenamiento,salida,n_epoch=10000,batch_size=84,show_metric=True)
modelo.save("modelo.tflearn")
```

SE MUESTRAN COMO HACE EL ENTRENAMIENTO

```
C:\WINDOWS\system32\cmd.exe
| Adam | epoch: 9991 | loss: 0.34426 - acc: 0.9819 -- iter: 34/34
Training Step: 9992 | total loss: +[1m-[32m0.30996-[0m-[0m
| Adam | epoch: 9992 | loss: 0.30996 - acc: 0.9837 -- iter: 34/34
Training Step: 9993 | total loss: +[1m-[32m0.27926-[0m-[0m | time: 0.016s
| Adam | epoch: 9993 | loss: 0.27926 - acc: 0.9853 -- iter: 34/34
Training Step: 9994 | total loss: +[1m-[32m0.25147-[0m-[0m
| Adam | epoch: 9994 | loss: 0.25147 - acc: 0.9868 -- iter: 34/34
Training Step: 9995 | total loss: +[1m-[32m0.22638-[0m-[0m
| Adam | epoch: 9995 | loss: 0.22638 - acc: 0.9881 -- iter: 34/34
Training Step: 9996 | total loss: +[1m-[32m0.20378-[0m-[0m
| Adam | epoch: 9996 | loss: 0.20378 - acc: 0.9893 -- iter: 34/34
Training Step: 9997 | total loss: +[1m-[32m0.18345-[0m-[0m
| Adam | epoch: 9997 | loss: 0.18345 - acc: 0.9904 -- iter: 34/34
Training Step: 9998 | total loss: +[1m-[32m0.16518-[0m-[0m
| Adam | epoch: 9998 | loss: 0.16518 - acc: 0.9913 -- iter: 34/34
Training Step: 9999 | total loss: +[1m-[32m0.14876-[0m-[0m
| Adam | epoch: 9999 | loss: 0.14876 - acc: 0.9922 -- iter: 34/34
Training Step: 10000 | total loss: +[1m-[32m0.13399-[0m-[0m
| Adam | epoch: 10000 | loss: 0.13399 - acc: 0.9930 -- iter: 34/34
--
```

(chat) C:\Users\SAIDEN3003\chat>

Se crea la función principal, la cual se encargara de tomar al entrada del usuario.

Para esto se usa while, junto con una variable llamada entrada.

Se crea una cubeta para saber que variables esta usando el usuario y verificar cuales están en los patrones.

Se procesa la entrada para que el bot, entienda los puntos, signos etc.

Se creara un nuevo for para identificar palabra por palabra

Se creara un if para saber si se a usado una de las palabras que están en el patron y estas se ayudara con la cubeta para saber si se esta usando

Para saber a que tag nos referimos, se ocupara modelo.predict junto con la cubeta

```
def mainbot():
    while True:
        entrada = input("Tu: ")
        cubeta = [0 for _ in range(len(palabras))]
        entradaProcesada = nltk.word_tokenize(entrada)
        entradaProcesada = [stemmer.stem(palabra.lower()) for palabra in entradaProcesada]
        for palabraIndividual in entradaProcesada:
            for i,palabra in enumerate(palabras):
                if palabra == palabraIndividual:
                    cubeta[i] = 1
        resultados = modelo.predict([numpy.array(cubeta)])
        print(resultados)

mainbot()
```

Se vera el entrenamiento
del bot y las
probabilidades que
arroja las cercana a 1 es
donde es el tag que se
usara

```
ON C:\WINDOWS\system32\cmd.exe - python mainbot.py
-- 
Training Step: 993 | total loss: <[1m<[32m0.00877<[0m<[0m | time: 0.002s
| Adam | epoch: 993 | loss: 0.00877 - acc: 1.0000 -- iter: 45/45
-- 
Training Step: 994 | total loss: <[1m<[32m0.00873<[0m<[0m | time: 0.001s
| Adam | epoch: 994 | loss: 0.00873 - acc: 1.0000 -- iter: 45/45
-- 
Training Step: 995 | total loss: <[1m<[32m0.00869<[0m<[0m | time: 0.002s
| Adam | epoch: 995 | loss: 0.00869 - acc: 1.0000 -- iter: 45/45
-- 
Training Step: 996 | total loss: <[1m<[32m0.00865<[0m<[0m | time: 0.001s
| Adam | epoch: 996 | loss: 0.00865 - acc: 1.0000 -- iter: 45/45
-- 
Training Step: 997 | total loss: <[1m<[32m0.00860<[0m<[0m | time: 0.001s
| Adam | epoch: 997 | loss: 0.00860 - acc: 1.0000 -- iter: 45/45
-- 
Training Step: 998 | total loss: <[1m<[32m0.00856<[0m<[0m | time: 0.002s
| Adam | epoch: 998 | loss: 0.00856 - acc: 1.0000 -- iter: 45/45
-- 
Training Step: 999 | total loss: <[1m<[32m0.00852<[0m<[0m | time: 0.001s
| Adam | epoch: 999 | loss: 0.00852 - acc: 1.0000 -- iter: 45/45
-- 
Training Step: 1000 | total loss: <[1m<[32m0.00848<[0m<[0m | time: 0.001s
| Adam | epoch: 1000 | loss: 0.00848 - acc: 1.0000 -- iter: 45/45
-- 
Tu: horario
[[4.6220092e-07 6.5878145e-03 6.9901130e-06 6.6254544e-04 9.7993726e-01
 5.9117783e-06 4.0723116e-04 3.1064264e-04 3.0651118e-03 6.8411035e-03
 2.1390521e-03 3.5991750e-05]]
```

```
resultadosIndices = numpy.argmax(resultados)
tag = tags[resultadosIndices]

for tagAux in datos["contenido"]:
    if tagAux["tag"] == tag:
        respuesta = tagAux["respuestas"]
print("BOTITO: ",random.choice(respuesta))

mainbot()
```

- Nuestro resultado lo modificaremos para saber el índice del tag con el resultadoIndices
- Para tener el tag igualamos a tag
- Para que nos arroje una respuesta usaremos un for con tagaux que recorrerá todo el archivo contenido
- Cuando lo tenga se parasa al if que si encuentra el tag y la respuesta lo pasara a respuesta

```
Training Step: 999 | total loss: 0.00703
| Adam | epoch: 999 | loss: 0.00703
-
Training Step: 1000 | total loss: 0.00700
| Adam | epoch: 1000 | loss: 0.00700
-
Tu: hola
BOT SITO: buenos dias, en que puedo a
Tu: me podrías dar el horario del res
BOT SITO: el horario del restaurante
Tu: que hay de comer
BOT SITO: cochinita pibill, pozole, o
Tu: que postres tienen
BOT SITO: pastel de chocolate, pansei
Tu: pueden ingresar bebés}
BOT SITO: acuerdate que solo son 5 pe
Tu:
```

- El bot ya no arroja numeros ya nos da una respuesta concreta entre todas las que ya tiene definidas

SE GUARDA LAS VARIABLES PARA UNA CARGA MAS RÁPIDA

```
#nltk.download('punkt')
with open("contenido.json", encoding='utf-8') as archivo:
    datos = json.load(archivo)
try:
    with open("variables.pickle","rb") as archivoPickle:
        palabras,tags, entrenamiento,salida= pickle.load(archivoPickle)
except:
    #print(datos)
    palabras=[]
    tags=[]
    auxX=[]
    auxY=[ ]
```

```
    #print(datos)
    entrenamiento = numpy.array(entrenamiento)
    salida = numpy.array(salida)
    with open("variables.pickle","wb") as archivoPickle:
        pickle.dump([(palabras,tags,entrenamiento,salida)],archivoPickle)
```

SE GUARDA LA RED

```
✓ try:  
    modelo.load("modelo.tflearn")  
✓ except:  
    modelo .fit(entrenamiento,salida,n_epoch=1000,batch_size=84,show_metric=True)  
    modelo.save("modelo.tflearn")
```

EL RESULTADO ES QUE NO APARECERÁ EL ENTRENAMIENTO Y EL BOT TRABAJARA NORMALMENTE

```
2021-01-24 15:57:19.688806: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect StreamExecutor with strength 1 edge matrix:  
2021-01-24 15:57:19.688806: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]  
2021-01-24 15:57:19.689732: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set  
2021-01-24 15:57:19.756039: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1261] Device interconnect StreamExecutor with strength 1 edge matrix:  
2021-01-24 15:57:19.756193: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1267]  
2021-01-24 15:57:19.756753: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set  
Tu: hola  
BOT SITO: buenos dias, en que puedo ayudarte  
Tu: me das el horario del restaurante  
BOT SITO: el horario del restaurante es de lunes a domingo de 10am a 12pm, pero dejamos de admitir clientes a las 10:30  
Tu: a que hora abren  
BOT SITO: aceptamos clientes desde las 10am  
Tu:
```