

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

Install Python 3.8.7 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now

C:\Users\SAIDEN3003N\AppData\Local\Programs\Python\Python38

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation

Choose location and features

☒ Install launcher for all users (recommended)

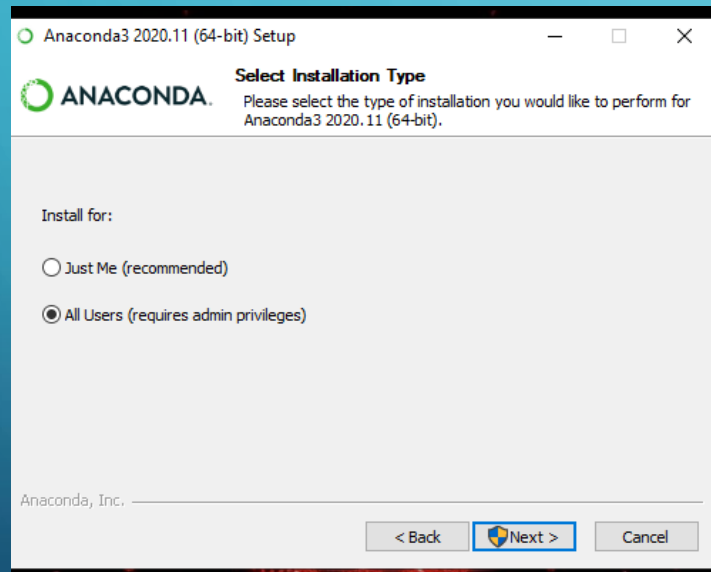
☒ Add Python 3.8 to PATH

PYTHON

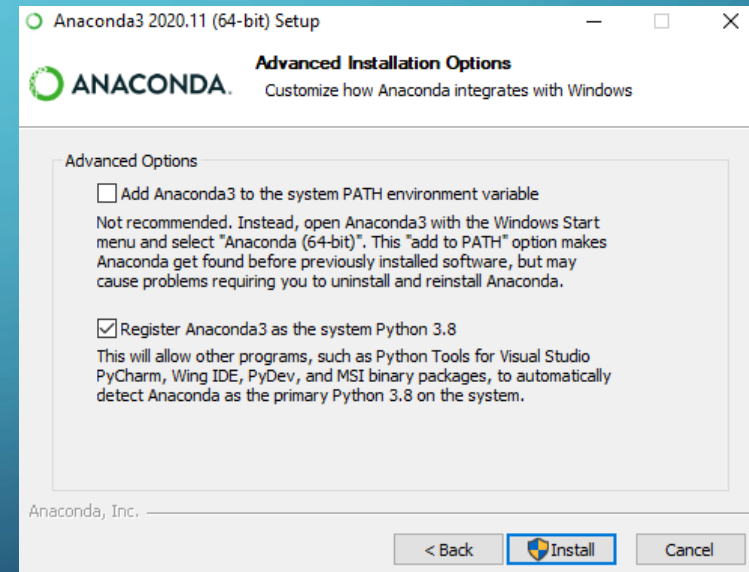
- Para instalar Python por el archivo ejecutable que se descarga de su pagina oficial.
- se seleccionaría las dos opciones agregando al path

ANACONDA

SE LE OTORGARA PERMISOS DE
USUARIOS



AL IGUAL QUE PYTHON SE
AGREGARA AL PATH



The background is a blue gradient. In the corners, there are white line-art illustrations of circuit boards or neural networks, with lines and small circles representing nodes.

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

SE ACEPTARA EL PROCESO DE
INSTALACIÓN

```
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-h62dcd97_4

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

```
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, nampy, tflearn y nltk.

para hacerlo se necesita entrar al entorno con la línea:

Activate chat

Y se decargarán 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 MB)
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-py3-none-any.whl (13 kB)
```

```
(chat) C:\Users\SAIDEN3003N>pip install numpy
Requirement already satisfied: numpy in c:\users\saiden3003n\.conda\envs\chat\lib\site-packages (from tensorflow) (1.21.0)

(chat) C:\Users\SAIDEN3003N>pip install tflearn
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 tflearn) (1.16.0)
Requirement already satisfied: numpy in c:\users\saiden3003n\.conda\envs\chat\lib\site-packages (from tflearn) (1.21.0)
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>
```

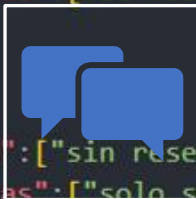


```

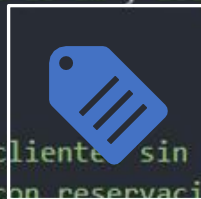
3     "patrones":["hola","un saludo","hello","buenos dias"],
4     "respuestas":["hola que tal, en que puedo ayudarte","Comó te va","un gusto de verte","buenos dias, en que puedo ayudarte"]
5 },
6 {"tag":"postre",
7     "patrones":["postre","que postre tienes","que hay de postre","que pontres hay"],
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 {"tag":"sin",
21     "patrones":["sin reservacion","aceptan cliente sin reservacion"],
22     "respuestas":["solo se acepta clientes con reservacion","no se aceptan clientes sin reservacion, deberias hacer una reservacion"]
23 },
24 },
25 {"tag":"cerrado",
26     "patrones":["cerrado","cierran","a que hora cierran"],
27     "respuestas":["El restaurante cierra a las 12 pm, pero dejamos de recibir clientes con reservacion a las 10pm"]
28 },
29 },
30 {"tag":"personas",
31     "patrones":["cuantas personas son en la reservacion","cuantas personas por reservacion","cuantos comensales pueden ir en una reservacion"],
32     "respuestas":["son 5 personas maximo","5 personas por reservacion","en una reservacio pueden ir 5 comensales"]
33 },
34 },
35 {"tag":"reservacion",
36     "patrones":["cuantas veces puedo reservar","puedo hacer mas de una reservacion","puedo hacer mas reservaciones"],
37     "respuestas":["solo puedes hacer una reservacion por persona","si ya hiciste una reservacion ya no puedes hacer otra"]
38 },

```

ARCHIVO .JSON



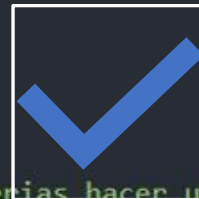
en este archivo se alojara el contenido que consta de:



Tags: la acción



Patrones: que son las palabras que están formada una oración



Respuestas: aquí se encontraran todas las respuestas que se pudieran dar a los patrones

IMPORTAR LAS LIBRERIAS

```
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
```

- 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

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

```
{
  'tag': 'saludo',
  'patrones': ['hola', 'un saludo', 'hello', 'buenos dias'],
  'respuestas': ['hola que tal',
    'en que puedo ayudarte', 'ComÃ³ te va', 'un gusto de verte', 'buenos dias, en que puedo ayudarte']},
  {'tag': 'postre',
    'patrones': ['postre', 'que postre tienes', 'que hay de postre', 'que pontres 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': 'abierto',
    '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': 'niños',
    'patrones': ['puede n entrar niños', 'pueden entrar bebes', 'cuantos niños pueden ir por reservacion'],
    'respuestas': ['se admiten niños, PAGAN DOBLE, solo toma en cuenta el menu del restaurante', 'acuerdate que solo son 5 personas maximo, contanto a los niños']},
  {'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 pasarlos 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', 'tienes'], ['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', 'bebes'], ['cuantos', 'niños', 'pueden', 'ir', 'por', 'reservacion'], ['como', 'estas'], ['hola', 'como', 'estas'], ['que', 'tal', 'estas']]
['saludo', 'saludo', 'saludo', 'saludo', 'postre', 'postre', 'postre', 'postre', 'despedida', 'despedida', 'despedida', 'despedida', 'abierto', 'abierto', 'abierto', 'abierto', 'abierto', 'abierto', 'sin', 'sin', 'cerrado', 'cerrado', 'cerrado', 'personas', 'personas', 'personas', 'reservacion', 'reservacion', 'reservacion', 'niños', 'niños', 'niños', 'estas', 'estas', 'estas']
['saludo', 'postre', 'despedida', 'abierto', 'sin', 'cerrado', 'personas', 'reservacion', 'niños', 'estas']
]

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


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, por esto se usará un for, el cual enumerará las palabras por un índice guardado en x

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

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

En entrenamiento se agregará 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 = salidaVacía[:]
    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 aremos que se reinicie 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 crea 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)
red = tflearn.fully_connected(red, 80)
red = tflearn.fully_connected(red, 80)
red = tflearn.fully_connected(red, 80)
red = tflearn.fully_connected(red, len(salida[0]), activation="softmax")
red = tflearn.regression(red)
```

Se creara un modelo.fity 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,80)
red = tflearn.fully_connected(red,80)
red = tflearn.fully_connected(red,80)
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\SAIDEN3003N\chat>
```

