



INSTITUTO TECNOLÓGICO DE IZTAPALAPA

PROYECTO: ROBÓTICA /CHATBOT DE SALUD

Presentada por:

Luna Rojo Alexander

Sanchez Valdez Daniel

Valtierra Cervantes Joel

Velazquez Chavez Gloria Marial

Profesor:

Parra Hernandez Abiel Tomas

CIUDAD DE MÉXICO
Noviembre 2020

ÍNDICE

Introducción.....	4
Resumen.....	5
Análisis de riesgo.....	5
¿Qué es un ChatBot?.....	6
¿Cómo funciona un ChatBot?.....	7
Construyendo el Bot.....	8

Introducción

Para comunicarnos ha cambiado, significa que ***cada vez tenemos menos conversaciones telefónicas y más conversaciones escritas***, y sobretodo significa que las compañías tienen la obligación (y la oportunidad) de atender a sus consumidores mediante este nuevo canal de comunicación.

Por eso un chatbot es un sistema informático capaz de mantener un diálogo mediante un lenguaje natural con un humano (o con otro chatbot). Este sistema informático ha sido especialmente diseñado y programado para poder interpretar el motivo o intención de la conversación, entender las respuestas de un humano y en base a ellas decidir qué debe responder a continuación. Esta capacidad para entender y establecer de forma natural un diálogo y procesarlo forman el NLP (Natural Language Processing) y el AI (Artificial Intelligence).

RESUMEN:

La asistencia médica a distancia multiplica su presencia ante la dificultad de la atención convencional. La COVID-19 ha desatado el temor al colapso sanitario, ya no solo por la pandemia, sino porque mientras ésta dure, continúan produciéndose otras patologías que requieren de atención médica.

En este sentido, con el uso de “chatbots de salud”, las compañías e instituciones de atención médica pueden proporcionar soluciones instantáneas a las consultas y preocupaciones de los usuarios.

Si pensamos en el paciente, los healthbots (chatbots) resultan de gran utilidad para:

- aclarar dudas en relación a un diagnóstico médico
- asistencia en caso de urgencias
- adherencia a los tratamientos
- acceder a la información de un tratamiento concreto

Análisis de riesgo

RIESGOS	SOLUCIONES PROPUESTAS
<ul style="list-style-type: none">-Robos de información-usuarios utilizan el chatbot con otro propósito-suplantación de identidad.-manipulación indebida.-divulgación de información.-Inflexible: éste no permitirá lidiar con algunas preguntas porque no las tendremos en nuestra base de datos.-perdida de datos-No aceptación por los usuarios: A muchas personas no les gusta interactuar con chatbots, prefieren hablar con personas reales.-Emociones: Un chatbot no puede interpretar el sarcasmo o los estados de	<ul style="list-style-type: none">-Protocolos seguros-el mensaje con esta información se destruirá después de un período de tiempo establecido.-La autenticación de usuario se utiliza para identificar que un usuario se verifica con credenciales de inicio de sesión válidas y seguras.

emoción de los usuarios que interactúan con ellos. A veces pueden ser malinterpretados.	
---	--

¿Qué es un chatbot?

Un chatbot es una pieza de software con inteligencia artificial en un dispositivo (Siri, Alexa, Google Assistant, etc.), aplicación, sitio web u otras redes que intentan medir las necesidades de los consumidores y luego ayudarlos a realizar una tarea particular como una transacción comercial. Reservas de hoteles, envío de formularios, etc. Hoy en día, casi todas las empresas tienen un chatbot implementado para interactuar con los usuarios. Algunas de las formas en que las empresas están utilizando los chatbots son:

Para entregar información de vuelo.

·Para conectar a los clientes y sus finanzas.

Como soporte al cliente.

¿Cómo funcionan los ChatBots?

En términos generales, existen dos variantes de chatbots: basadas en reglas y autoaprendizaje.

1. En un enfoque basado en reglas, un bot responde preguntas basadas en algunas reglas que previamente han sido entrenadas. Las reglas definidas pueden ser muy simples o muy complejas. Los bots pueden manejar consultas simples pero fallan en administrar las complejas.

2. Los robots de autoaprendizaje son los que utilizan algunos enfoques basados en el aprendizaje automático y son definitivamente más eficientes que los robots basados en reglas. Estos bots pueden ser de otros dos tipos: basados en recuperación o generativos.

i) En los modelos basados en recuperación , un chatbot utiliza cierta heurística para seleccionar una respuesta de una biblioteca de respuestas predefinidas. El chatbot utiliza el mensaje y el contexto de conversación para seleccionar la mejor respuesta de una lista predefinida de mensajes a dar. El contexto puede incluir una posición actual en el árbol de diálogo, todos los mensajes anteriores en la conversación, variables guardadas previamente (por ejemplo, nombre de usuario). Las heurísticas para seleccionar una respuesta se pueden diseñar de muchas maneras diferentes, desde la lógica condicional en el que se basan las reglas o los clasificadores de aprendizaje automático.

ii) Los bots generativos pueden generar las respuestas y no siempre responden con una de las respuestas de un conjunto de respuestas. Esto los hace más inteligentes a medida que toman palabra por palabra de la consulta y generan las respuestas.

Construyendo el Bot

Un hábil en el conocimiento de scikit biblioteca y NLTK se asume. Sin embargo, si es nuevo en PNL, aún puede leer el artículo y luego consultar los recursos. PNL.

El campo de estudio que se centra en las interacciones entre el lenguaje humano y las computadoras se llama Procesamiento del lenguaje natural, o PNL para abreviar. Se encuentra en la intersección de la informática, la inteligencia artificial y la lingüística computacional. La PNL es una forma para que las computadoras analicen, comprendan y obtengan significado del lenguaje humano de una manera inteligente y útil.

NLTK: una breve introducción

NLTK (Natural Language Toolkit) es una plataforma líder para crear programas Python que funcionen con datos de lenguaje humano. NLTK ha sido llamado "una herramienta maravillosa para enseñar y trabajar en lingüística computacional usando Python" y "una biblioteca increíble para jugar con el lenguaje natural".
Instalación de paquetes NLTK

Importar NLTK y ejecutar. `nltk.download()`. Esto abrirá el descargador de NLTK desde donde puede elegir los corpus y modelos para descargar. También puede descargar todos los paquetes a la vez.

Bag of Words

Después de la fase de preprocesamiento inicial, necesitamos transformar el texto en un vector significativo (o matriz) de números. La bolsa de palabras es una representación de texto que describe la aparición de palabras dentro de un documento. Se trata de dos cosas:

- Un vocabulario de palabras conocidas.

- Una medida de la presencia de palabras conocidas.

¿Por qué se llama una "bag" of words? Esto se debe a que cualquier información sobre el orden o la estructura de las palabras en el documento se descarta y al modelo solo le preocupa si las palabras conocidas aparecen en el documento, no donde aparecen en el documento.

Preprocesamiento de texto con NLTK

El principal problema con los datos de texto es que están todos en formato de texto (cadenas). Sin embargo, los algoritmos de aprendizaje automático necesitan algún tipo de vector de características numéricas para realizar la tarea. Entonces, antes de comenzar con cualquier proyecto de PNL,

definiremos una función para un saludo del bot, es decir, si la entrada de un usuario es un saludo, el bot devolverá una respuesta de saludo. ELIZA utiliza una simple concordancia de palabras clave para los saludos

```
GREETING_INPUTS = ("hola", "hola", "saludos", "sup", "qué pasa",  
"hey",)  
  
GREETING_RESPONSES = ("hola", "oye", "** asiente **", "hola", "hola",  
";Me alegro! Me estás hablando")  
  
def saludo (oración):  
    para palabra en  
        oración.split () : si palabra.bajo () en ENTRADAS_SALUDO:  
            devuelve elección aleatoria (RESPUESTAS_SALUDO)
```

Para generar una respuesta de nuestro bot para las preguntas de entrada, se utilizará el concepto de similitud de documentos. Entonces comenzamos importando los módulos necesarios.

- Desde la biblioteca scikit learn, importe el vectorizador TFidf para convertir una colección de documentos en bruto en una matriz de funciones TF-IDF.

Además, importe el módulo de similitud de coseno de la biblioteca scikit learn. Esto se utilizará para encontrar la similitud entre las palabras ingresadas por el usuario y las palabras en el corpus. Esta es la implementación más simple posible de un chatbot.

Definimos una respuesta de función que busca en el enunciado del usuario una o más palabras clave conocidas y devuelve una de varias respuestas posibles. Si no encuentra la entrada que coincida con ninguna de las palabras clave, devuelve una respuesta: "¡Lo siento! No te entiendo "

```
respuesta def (user_response):
    robo_response = ""

    TfIdfVec = TfIdfVectorizer (tokenizer = LemNormalize, stop_words =
    'english')
    tfidf = TfIdfVec.fit_transform (sent_tokens)
    vals = cosine_similarity (tfidf [-1], tfidf)
    idx = vals.argsort ()
    plans = [0] [- 2 vals.flatten ()]
    flat.sort ()
    req_tfidf = flat [-2]

    if (req_tfidf == 0):
        robo_response = robo_response + "¡Lo siento! No te entiendo"
        return robo_response
    else:
        robo_response = robo_response + sent_tokens [idx]
        return robo_response
```

Creación del ChatBot

Para la creación del entorno de desarrollo se usa la línea conda create -n chat python 3.8

También la importación de librerías.

la librería nltk permitirá el procesamiento de lenguaje natural.

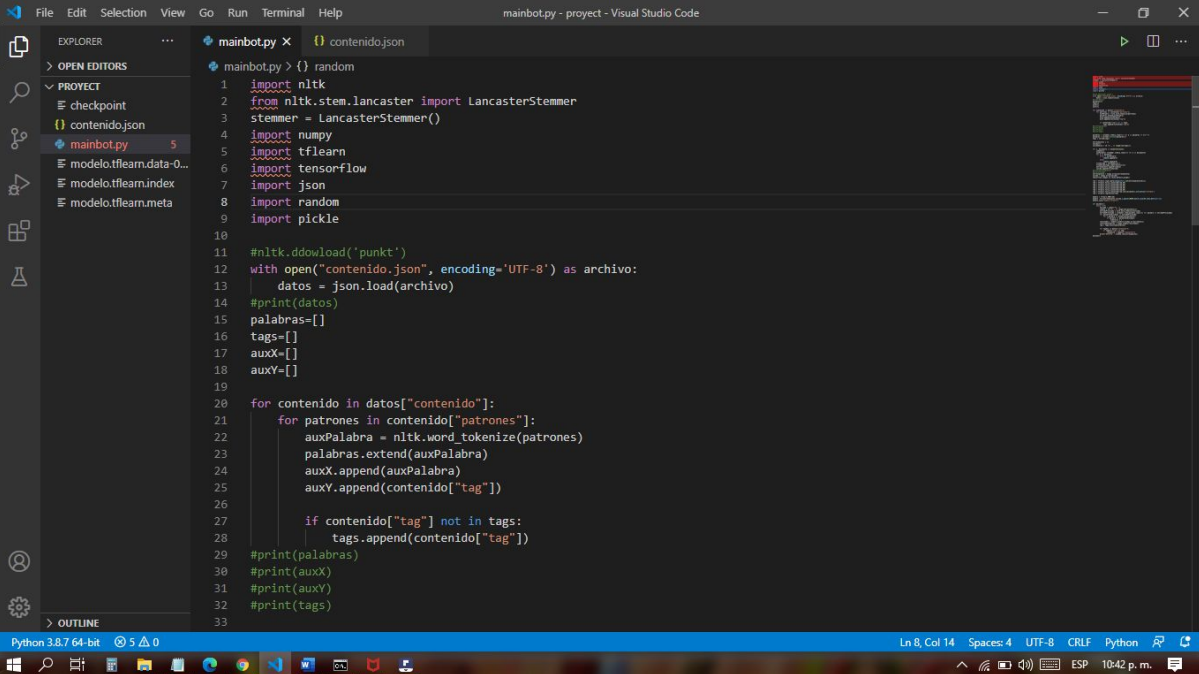
nltk.stem.lancaster

esto permitirá transformar las palabras, para que sean más entendibles para el chatbot

json que ahí se tendrá la información random 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
```

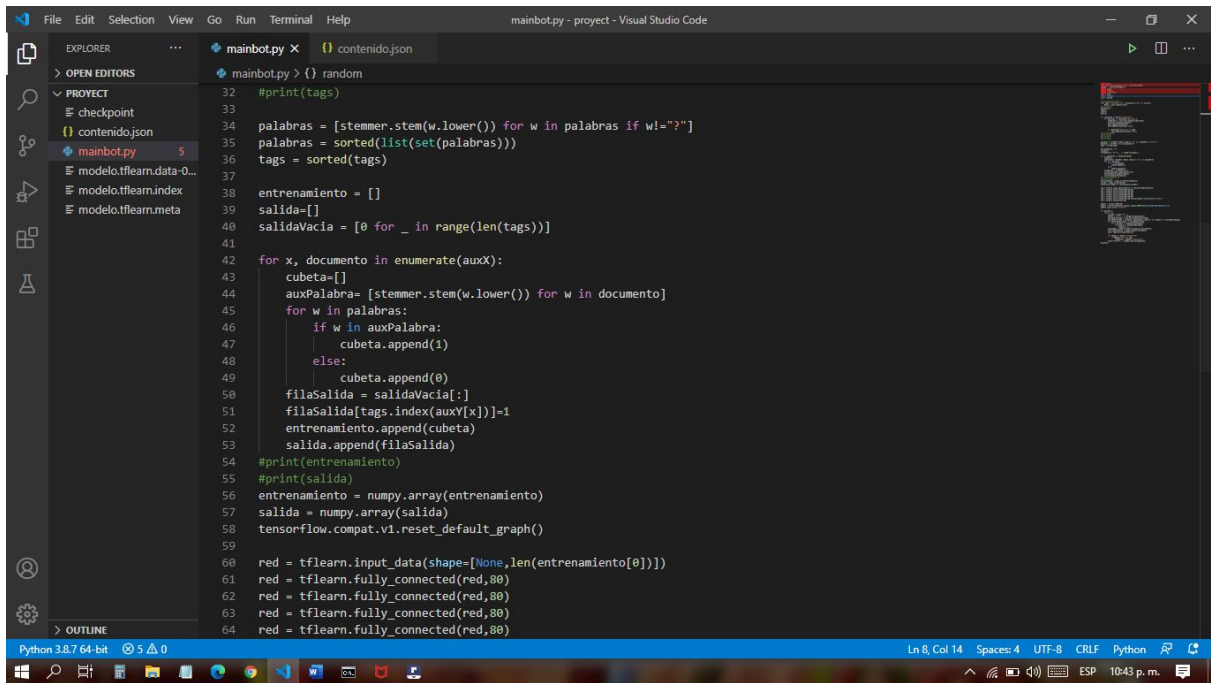
El código en Python para la elaboración de este chatbot



The image shows a Visual Studio Code editor window with a Python script named `mainbot.py` open. The script is located in a project named `mainbot.py - proyect - Visual Studio Code`. The Explorer panel on the left shows the project structure, including files like `checkpoint`, `contenido.json`, `mainbot.py`, `modelo.tflearn.data-0...`, `modelo.tflearn.index`, and `modelo.tflearn.meta`. The main editor area displays the following Python code:

```
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", encoding='UTF-8') as archivo:
13     datos = json.load(archivo)
14     #print(datos)
15     palabras=[]
16     tags=[]
17     auxX=[]
18     auxY=[]
19
20     for contenido in datos["contenido"]:
21         for patrones in contenido["patrones"]:
22             auxPalabra = nltk.word_tokenize(patrones)
23             palabras.extend(auxPalabra)
24             auxX.append(auxPalabra)
25             auxY.append(contenido["tag"])
26
27             if contenido["tag"] not in tags:
28                 tags.append(contenido["tag"])
29     #print(palabras)
30     #print(auxX)
31     #print(auxY)
32     #print(tags)
33
```

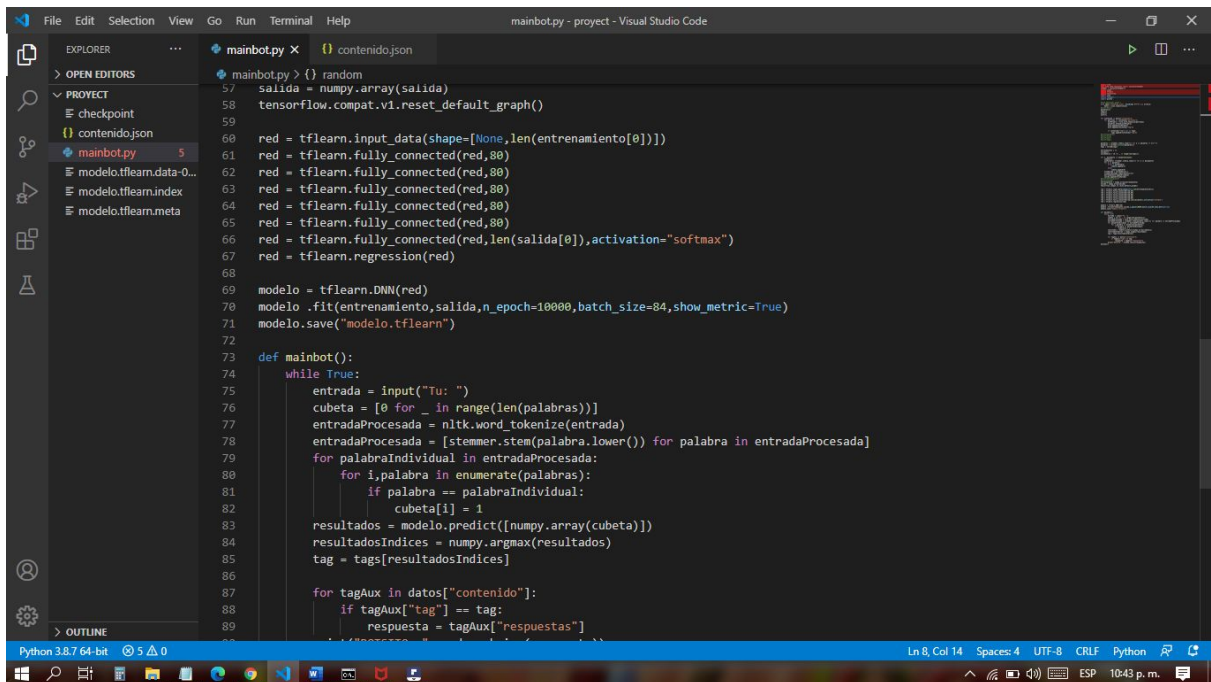
The status bar at the bottom indicates the editor is using Python 3.8.7 64-bit, with 5 lines and 0 columns selected. The current position is Line 8, Column 14. The encoding is UTF-8, and the line ending is CRLF. The system clock shows 10:42 p. m.



The screenshot shows the Visual Studio Code editor with a Python file named `mainbot.py` open. The Explorer sidebar on the left shows a project structure with files like `contenido.json`, `mainbot.py`, `modelo.tflern.data-0...`, `modelo.tflern.index`, and `modelo.tflern.meta`. The main editor area displays the following code:

```
32 #print(tags)
33
34 palabras = [stemmer.stem(w.lower()) for w in palabras if w!="?"]
35 palabras = sorted(list(set(palabras)))
36 tags = sorted(tags)
37
38 entrenamiento = []
39 salida=[]
40 salidaVacia = [0 for _ in range(len(tags))]
41
42 for x, documento in enumerate(auxX):
43     cubeta=[]
44     auxPalabra= [stemmer.stem(w.lower()) for w in documento]
45     for w in palabras:
46         if w in auxPalabra:
47             cubeta.append(1)
48         else:
49             cubeta.append(0)
50     filaSalida = salidaVacia[:]
51     filaSalida[tags.index(auxX[x])] = 1
52     entrenamiento.append(cubeta)
53     salida.append(filaSalida)
54 #print(entrenamiento)
55 #print(salida)
56 entrenamiento = numpy.array(entrenamiento)
57 salida = numpy.array(salida)
58 tensorflow.compat.v1.reset_default_graph()
59
60 red = tflearn.input_data(shape=[None,len(entrenamiento[0])])
61 red = tflearn.fully_connected(red,80)
62 red = tflearn.fully_connected(red,80)
63 red = tflearn.fully_connected(red,80)
64 red = tflearn.fully_connected(red,80)
```

The status bar at the bottom indicates the file is at line 8, column 14, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python syntax highlighting.



This screenshot shows the continuation of the `mainbot.py` file. The code continues from where the previous screenshot left off:

```
65 red = tflearn.fully_connected(red,80)
66 red = tflearn.fully_connected(red,len(salida[0]),activation="softmax")
67 red = tflearn.regression(red)
68
69 modelo = tflearn.DNN(red)
70 modelo.fit(entrenamiento,salida,n_epoch=10000,batch_size=84,show_metric=True)
71 modelo.save("modelo.tflern")
72
73 def mainbot():
74     while True:
75         entrada = input("Tu: ")
76         cubeta = [0 for _ in range(len(palabras))]
77         entradaProcesada = nltk.word_tokenize(entrada)
78         entradaProcesada = [stemmer.stem(palabra.lower()) for palabra in entradaProcesada]
79         for palabraIndividual in entradaProcesada:
80             for i,palabra in enumerate(palabras):
81                 if palabra == palabraIndividual:
82                     cubeta[i] = 1
83             resultados = modelo.predict([numpy.array(cubeta)])
84             resultadosIndice = numpy.argmax(resultados)
85             tag = tags[resultadosIndice]
86
87         for tagAux in datos["contenido"]:
88             if tagAux["tag"] == tag:
89                 respuesta = tagAux["respuestas"]
90                 print(respuesta)
```

The status bar at the bottom shows the cursor is now at line 8, column 14, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python syntax highlighting. The time is 10:43 p.m.

objetivo tener el inicio de la plática y lo declaramos como “saludo” , la siguiente variable en este caso “patrones” , declaramos las posibles palabras que dan comienzo a una plática y la variable “respuesta” como respuesta de la variable “patrones”.

Lo que nosotros hicimos fueron las posibles palabras y oraciones que tendrá nuestro chatbot desde el saludo hasta la despedida.

Podemos concluir entonces en que JSON es un formato común para ‘serializar’ y ‘deserializar’ objetos en la mayoría de los lenguajes.

Abrimos CMD para ejecutar a nuestro Chatbot e interactuar con nuestro asistente virtual

```
CAWINDOWS\system32\cmd.exe - python mainbot.py
Training Step: 9980 | total loss: +[1m-[32m0.07725+[0m-[0m
| Adam | epoch: 9990 | loss: 0.07725 - acc: 0.9531 -- iter: 43/43
...
Training Step: 9991 | total loss: +[1m-[32m0.07665+[0m-[0m
| Adam | epoch: 9991 | loss: 0.07665 - acc: 0.9532 -- iter: 43/43
...
Training Step: 9992 | total loss: +[1m-[32m0.07610+[0m-[0m
| Adam | epoch: 9992 | loss: 0.07610 - acc: 0.9532 -- iter: 43/43
...
Training Step: 9993 | total loss: +[1m-[32m0.07557+[0m-[0m
| Adam | epoch: 9993 | loss: 0.07557 - acc: 0.9532 -- iter: 43/43
...
Training Step: 9994 | total loss: +[1m-[32m0.07508+[0m-[0m
| Adam | epoch: 9994 | loss: 0.07508 - acc: 0.9533 -- iter: 43/43
...
Training Step: 9995 | total loss: +[1m-[32m0.07462+[0m-[0m
| Adam | epoch: 9995 | loss: 0.07462 - acc: 0.9533 -- iter: 43/43
...
Training Step: 9996 | total loss: +[1m-[32m0.07418+[0m-[0m
| Adam | epoch: 9996 | loss: 0.07418 - acc: 0.9533 -- iter: 43/43
...
Training Step: 9997 | total loss: +[1m-[32m0.07377+[0m-[0m | time: 0.016s
| Adam | epoch: 9997 | loss: 0.07377 - acc: 0.9533 -- iter: 43/43
...
Training Step: 9998 | total loss: +[1m-[32m0.07339+[0m-[0m
| Adam | epoch: 9998 | loss: 0.07339 - acc: 0.9533 -- iter: 43/43
...
Training Step: 9999 | total loss: +[1m-[32m0.07302+[0m-[0m
| Adam | epoch: 9999 | loss: 0.07302 - acc: 0.9534 -- iter: 43/43
...
Training Step: 10000 | total loss: +[1m-[32m0.07268+[0m-[0m
| Adam | epoch: 10000 | loss: 0.07268 - acc: 0.9534 -- iter: 43/43
...
Tu: Hola
BOTSITO: Hola, un gusto de verte, en que puedo ayudarte?
Tu: tengo malestar
BOTSITO: ¿aparte de la tos, tienes otro sintoma?
Tu: solo tengo malestar
BOTSITO: reposa una hora
Tu: muchas gracias
BOTSITO:
Tu: gracias
BOTSITO: cuidate y recuerda 'sana distancia'
Tu:
```