LOVELY PROFESSIONAL UNIVERSITY

Academic Task-3

(Artificial Intelligence)

School of Computer Science and Engineering Faculty of Technology & Sciences

Name of the faculty member:- Mrs. Ankita Wadhawan

Course Code:- INT 404 Course

Title:- Artificial Intelligence

Student Name:- Kushagr Lohia, Kartik Prashar, Suraj Singh Mudila, Amandeep Singh Gautam

Student ID:- 11803966, 11803919, 11803922, 11804943

Roll Number:- 53, 54, 55, 64

Section:- K18UW

GitHub Link:- https://github.com/Kprashar/genuine

Abstract:-

A chatbot is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent.

Introduction:-

What is Chatbot?
A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human. Chatbots are used a lot in customer interaction, marketing on social network sites and instantly messaging the client. There are two basic types of chatbot models based on how they are built; Retrieval based and Generative based models.

1. Retrieval based Chatbots

A retrieval-based chatbot uses predefined input patterns and responses. It then uses some type of heuristic approach to select the appropriate response. It is widely used in the industry to make goal-oriented chatbots where we can customize the tone and flow of the chatbot to drive our customers with the best experience.

2. Generative based Chatbots

Generative models are not based on some predefined responses.
They are based on seq to seq neural networks. It is the same idea as machine translation. In machine translation, we translate the source code from one language to another language but here, we are going to transform input into an output. It needs a large amount of data and it is based on Deep Neural networks.

Literature Review:-

What is Natural Language Processing?

Natural language processing (NLP) is a field of artificial intelligence in which computers analysis, understand, and derive meaning from human language in a smart and useful way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation.

What is Natural Language Processing good for?

- Summarize blocks of text using Summarizer to extract the most important and central ideas while ignoring irrelevant information.
- Create a chat bot using Parsey McParseface, a language parsing deep learning model made by Google that uses Point-of-Speech tagging.
- **Automatically generate keyword tags** from content using AutoTag, which leverages LDA, a technique that discovers topics contained within a body of text.
- **Identify the type of entity extracted**, such as it being a person, place, or organization using Named Entity Recognition.
- Use Sentiment Analysis to **identify the sentiment of a string of text**, from very negative to neutral to very positive.
- **Reduce words to their root**, or stem, using Porter Stemmer, or **break up text into tokens** using Tokenizer.

What is Machine Learning?

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: *The ability to learn*. Machine learning is actively being used today, perhaps in many more places than one would expect.

Project Description:-

In this Python project with source code, we are going to build a chatbot using deep learning techniques. The chatbot will be trained on the dataset which contains categories (intents), pattern and responses. We use a special recurrent neural network (LSTM) to classify which category the user's message belongs to and then we will give a random response from the list of responses.

The Dataset

The dataset we will be using is 'intents.json'. This is a JSON file that contains the patterns we need to find and the responses we want to return to the user.

- Intents.json – The data file which has predefined patterns and responses.
- train_chatbot.py – In this Python file, we wrote a script to build the model and train our chatbot.
- Words.pkl – This is a pickle file in which we store the words Python object that contains a list of our vocabulary.
- Classes.pkl – The classes pickle file contains the list of categories.
- Chatbot_model.h5 – This is the trained model that contains information about the model and has weights of the neurons.
- Chatgui.py – This is the Python script in which we implemented GUI for our chatbot. Users can easily interact with the bot.

1. Import and load the data file

First, make a file name as train_chatbot.py. We import the necessary packages for our chatbot and initialize the variables we will use in our Python project.

2. Pre-process data

When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words.

3. Create training and testing data

Now, we will create the training data in which we will provide the input and the output. Our input will be the pattern and output will be the class our input pattern belongs to. But the computer doesn't understand text so we will convert text into numbers.

4. Build the model

We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as ‘chatbot_model.h5’.

5. Predict the response (Graphical User Interface)

Now to predict the sentences and get a response from the user to let us create a new file ‘chatgui.py’.

We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve us a random response from the list of responses.

Again we import the necessary packages and load the ‘words.pkl’ and ‘classes.pkl’ pickle files which we have created when we trained our model.

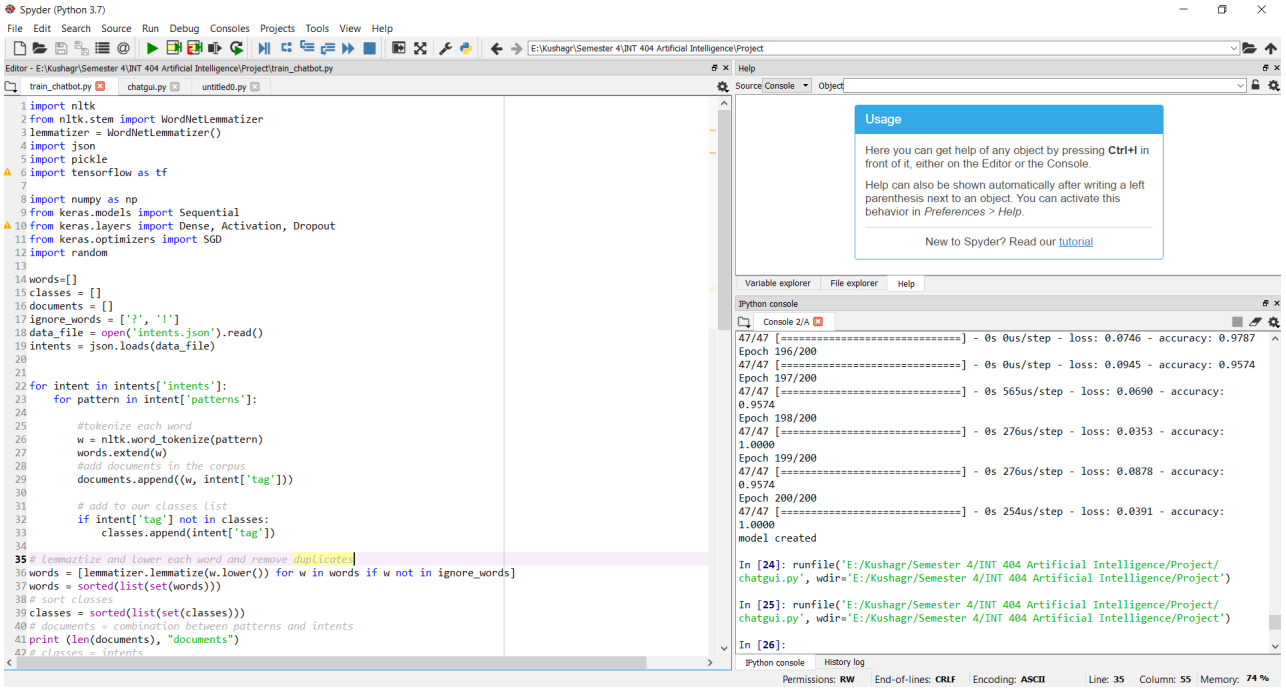
6. Run the chatbot

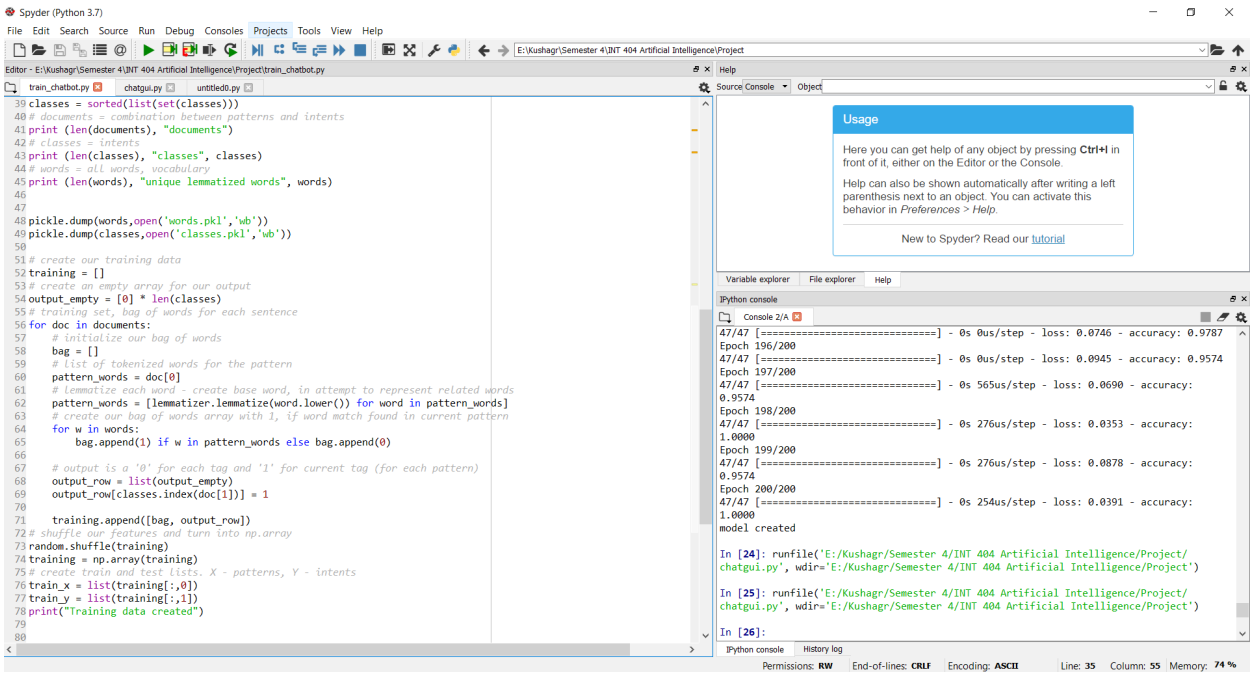
To run the chatbot, we have two main files; train_chatbot.py and chatgui.py.

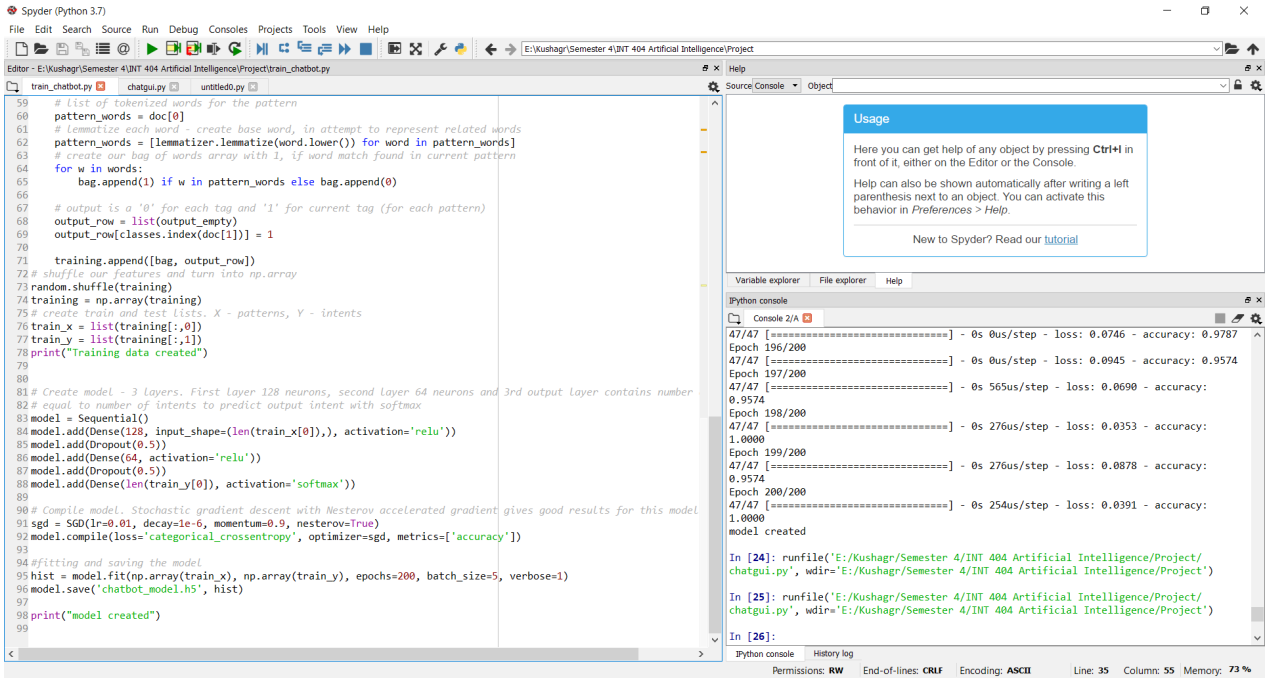
Conclusion:-

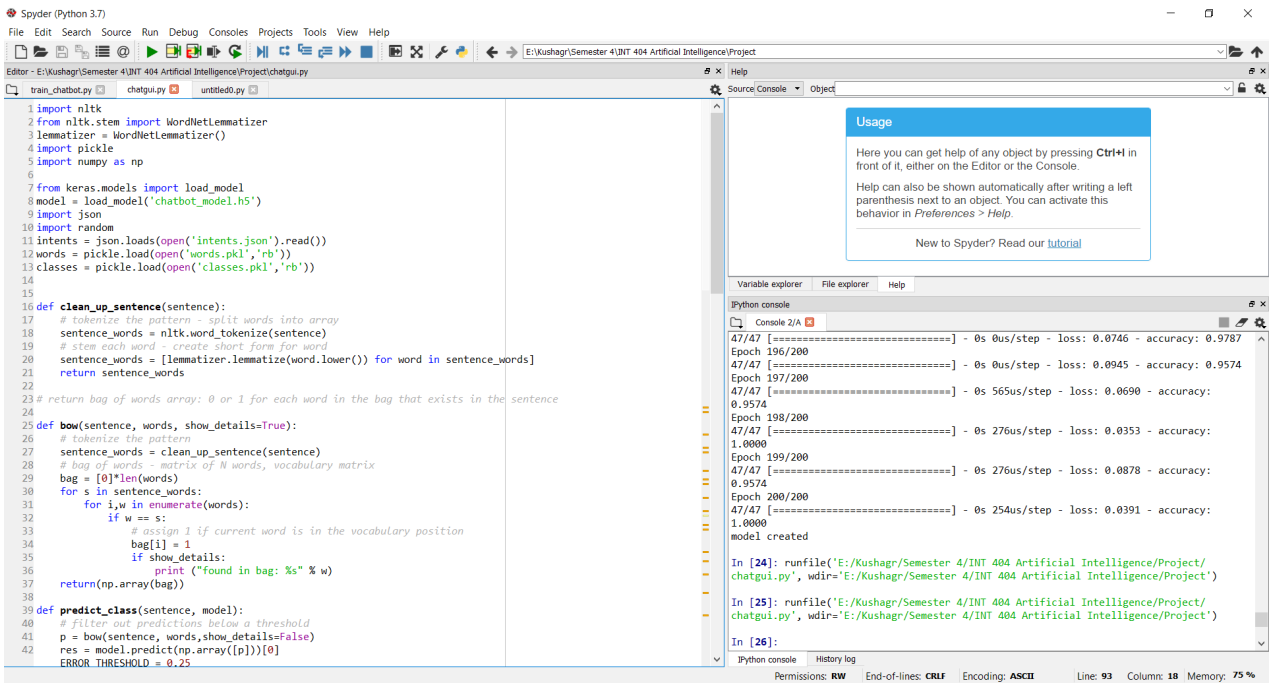
In this Python data science project, we understood about chatbots and implemented a deep learning version of a chatbot in Python which is accurate. You can customize the data according to business requirements and train the chatbot with great accuracy. Chatbots are used everywhere and all businesses is looking forward to implementing bot in their workflow.

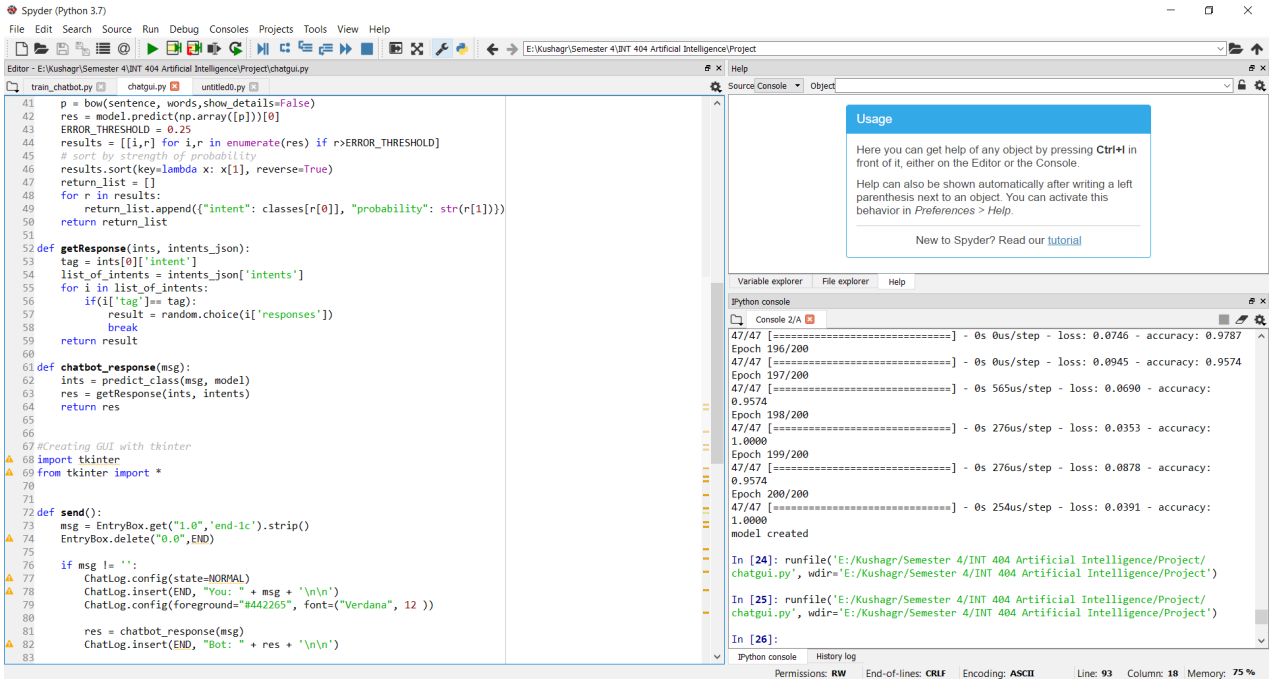
Output:-

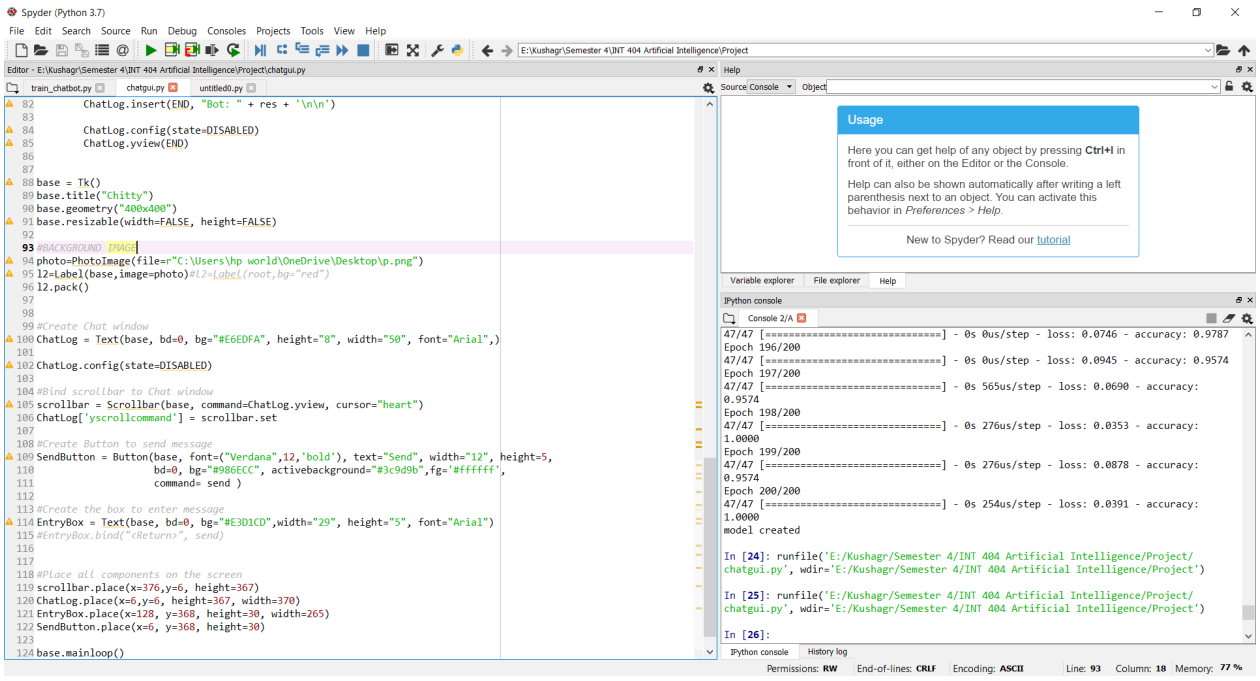

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
train_chatbot.py
chatgui.py
untitled0.py
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer
File explorer
Help
IPython console
Console 2/A
47/47 [==============================] - 0s 0us/step - loss: 0.0746 - accuracy: 0.9787
Epoch 196/200
47/47 [==============================] - 0s 0us/step - loss: 0.0945 - accuracy: 0.9574
Epoch 197/200
47/47 [==============================] - 0s 565us/step - loss: 0.0690 - accuracy: 0.9574
Epoch 198/200
47/47 [==============================] - 0s 276us/step - loss: 0.0353 - accuracy: 1.0000
Epoch 199/200
47/47 [==============================] - 0s 276us/step - loss: 0.0878 - accuracy: 0.9574
Epoch 200/200
47/47 [==============================] - 0s 254us/step - loss: 0.0391 - accuracy: 1.0000
model created
In [24]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [25]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [26]:
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 35
Column: 55
Memory: 74 %

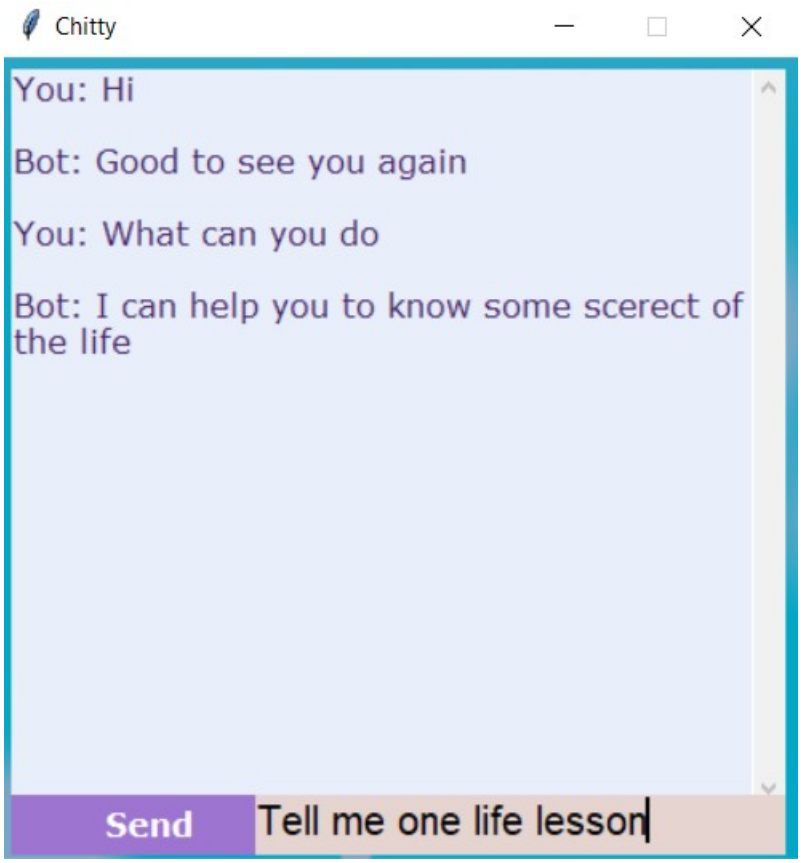

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
train_chatbot.py
chatgui.py
untitled0.py
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer
File explorer
Help
IPython console
Console 2/A
47/47 [==============================] - 0s 0us/step - loss: 0.0746 - accuracy: 0.9787
Epoch 196/200
47/47 [==============================] - 0s 0us/step - loss: 0.0945 - accuracy: 0.9574
Epoch 197/200
47/47 [==============================] - 0s 565us/step - loss: 0.0690 - accuracy: 0.9574
Epoch 198/200
47/47 [==============================] - 0s 276us/step - loss: 0.0353 - accuracy: 1.0000
Epoch 199/200
47/47 [==============================] - 0s 276us/step - loss: 0.0878 - accuracy: 0.9574
Epoch 200/200
47/47 [==============================] - 0s 254us/step - loss: 0.0391 - accuracy: 1.0000
model created
In [24]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [25]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [26]:
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 35
Column: 55
Memory: 74 %

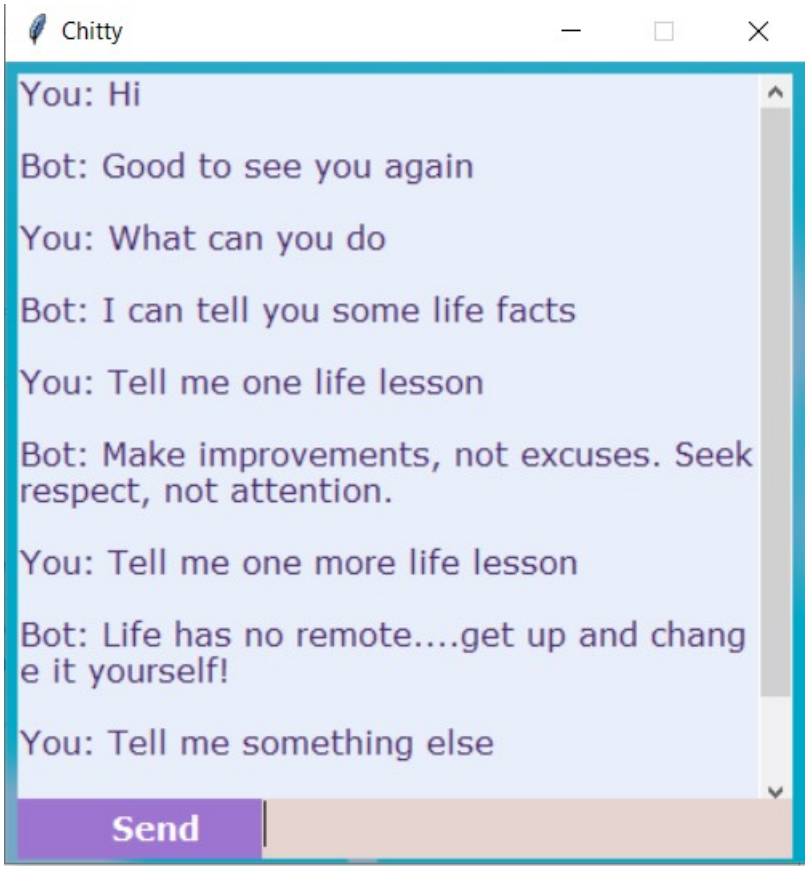

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project
Editor - E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project\train_chatbot.py
train_chatbot.py
chatgui.py
untitled0.py
# list of tokenized words for the pattern
pattern_words = doc[0]
# lemmatize each word - create base word, in attempt to represent related words
pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
# create our bag of words array with 1, if word match found in current pattern
for w in words:
bag.append(1) if w in pattern_words else bag.append(0)
# output is a '0' for each tag and '1' for current tag (for each pattern)
output_row = list(output_empty)
output_row[classes.index(doc[1])] = 1
training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))
# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbot_model.h5', hist)
print("model created")
Help
Source Console
Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer
File explorer
Help
IPython console
Console 2/A
47/47 [==============================] - 0s 0us/step - loss: 0.0746 - accuracy: 0.9787
Epoch 196/200
47/47 [==============================] - 0s 0us/step - loss: 0.0945 - accuracy: 0.9574
Epoch 197/200
47/47 [==============================] - 0s 565us/step - loss: 0.0690 - accuracy:
0.9574
Epoch 198/200
47/47 [==============================] - 0s 276us/step - loss: 0.0353 - accuracy:
1.0000
Epoch 199/200
47/47 [==============================] - 0s 276us/step - loss: 0.0878 - accuracy:
0.9574
Epoch 200/200
47/47 [==============================] - 0s 254us/step - loss: 0.0391 - accuracy:
1.0000
model created
In [24]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [25]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [26]:
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 35
Column: 55
Memory: 73 %

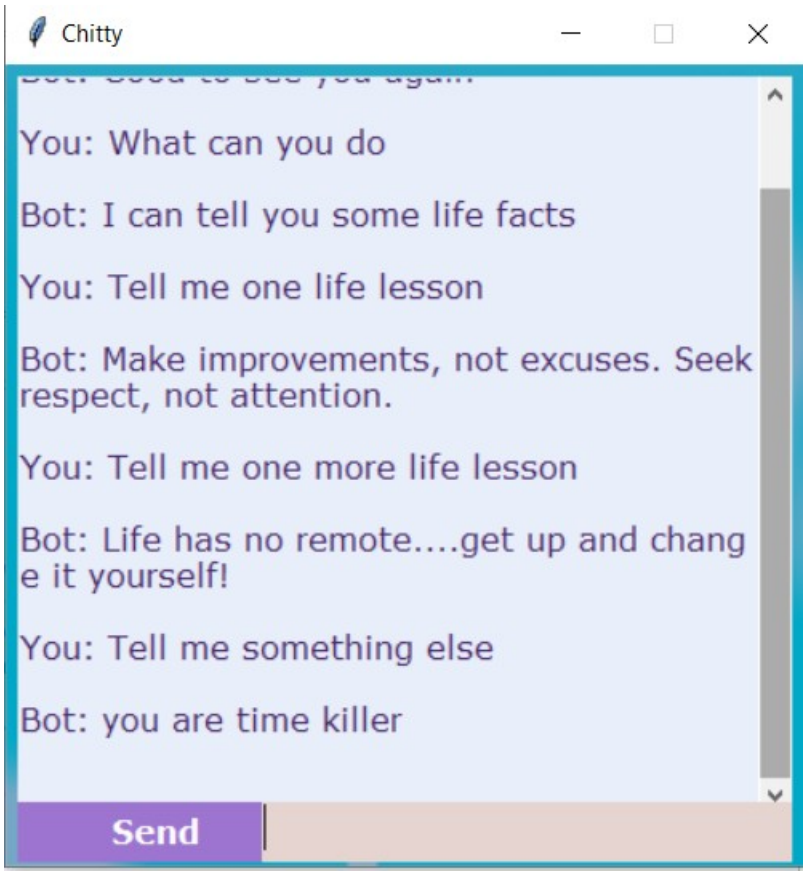

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project
Editor - E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project\chatgui.py
train_chatbot.py
chatgui.py
untitled0.py
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
from keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
def clean_up_sentence(sentence):
# tokenize the pattern - split words into array
sentence_words = nltk.word_tokenize(sentence)
# stem each word - create short form for word
sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
def bow(sentence, words, show_details=True):
# tokenize the pattern
sentence_words = clean_up_sentence(sentence)
# bag of words - matrix of N words, vocabulary matrix
bag = [0]*len(words)
for s in sentence_words:
for i,w in enumerate(words):
if w == s:
# assign 1 if current word is in the vocabulary position
bag[i] = 1
if show_details:
print ("found in bag: %s" % w)
return(np.array(bag))
def predict_class(sentence, model):
# filter out predictions below a threshold
p = bow(sentence, words,show_details=False)
res = model.predict(np.array([p]))[0]
ERROR_THRESHOLD = 0.25
Help
Source Console
Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer
File explorer
Help
IPython console
Console 2/A
47/47 [==============================] - 0s 0us/step - loss: 0.0746 - accuracy: 0.9787
Epoch 196/200
47/47 [==============================] - 0s 0us/step - loss: 0.0945 - accuracy: 0.9574
Epoch 197/200
47/47 [==============================] - 0s 565us/step - loss: 0.0690 - accuracy:
0.9574
Epoch 198/200
47/47 [==============================] - 0s 276us/step - loss: 0.0353 - accuracy:
1.0000
Epoch 199/200
47/47 [==============================] - 0s 276us/step - loss: 0.0878 - accuracy:
0.9574
Epoch 200/200
47/47 [==============================] - 0s 254us/step - loss: 0.0391 - accuracy:
1.0000
model created
In [24]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [25]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [26]:
IPython console
History log
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 93
Column: 18
Memory: 75 %


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project
Editor - E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project\chatgui.py
train_chatbot.py
chatgui.py
untitled0.py
41 p = bow(sentence, words,show_details=False)
42 res = model.predict(np.array([p]))[0]
43 ERROR_THRESHOLD = 0.25
44 results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
45 # sort by strength of probability
46 results.sort(key=lambda x: x[1], reverse=True)
47 return_list = []
48 for r in results:
49 return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
50 return return_list
51
52 def getResponse(ints, intents_json):
53 tag = ints[0]['intent']
54 list_of_intents = intents_json['intents']
55 for i in list_of_intents:
56 if(i['tag']== tag):
57 result = random.choice(i['responses'])
58 break
59 return result
60
61 def chatbot_response(msg):
62 ints = predict_class(msg, model)
63 res = getResponse(ints, intents)
64 return res
65
66
67 #Creating GUI with tkinter
68 import tkinter
69 from tkinter import *
70
71
72 def send():
73 msg = EntryBox.get("1.0",'end-1c').strip()
74 EntryBox.delete("0.0",END)
75
76 if msg != '':
77 ChatLog.config(state=NORMAL)
78 ChatLog.insert(END, "You: " + msg + '\n\n')
79 ChatLog.config(foreground="#442265", font=("Verdana", 12))
80
81 res = chatbot_response(msg)
82 ChatLog.insert(END, "Bot: " + res + '\n\n')
83
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer File explorer Help
IPython console
Console 2/A
47/47 [==============================] - 0s 0us/step - loss: 0.0746 - accuracy: 0.9787
Epoch 196/200
47/47 [==============================] - 0s 0us/step - loss: 0.0945 - accuracy: 0.9574
Epoch 197/200
47/47 [==============================] - 0s 565us/step - loss: 0.0690 - accuracy: 0.9574
Epoch 198/200
47/47 [==============================] - 0s 276us/step - loss: 0.0353 - accuracy: 1.0000
Epoch 199/200
47/47 [==============================] - 0s 276us/step - loss: 0.0878 - accuracy: 0.9574
Epoch 200/200
47/47 [==============================] - 0s 254us/step - loss: 0.0391 - accuracy: 1.0000
model created
In [24]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [25]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [26]:
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 93 Column: 18 Memory: 75 %


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project
Editor - E:\Kushagr\Semester 4\INT 404 Artificial Intelligence\Project\chatgui.py
train_chatbot.py
chatgui.py
untitled0.py
82 ChatLog.insert(END, "Bot: " + res + '\n\n')
83
84 ChatLog.config(state=DISABLED)
85 ChatLog.yview(END)
86
87
88 base = Tk()
89 base.title("Chitty")
90 base.geometry("400x400")
91 base.resizable(width=FALSE, height=FALSE)
92
93 #BACKGROUND IMAGE
94 photo=PhotoImage(file="C:\Users\hp world\OneDrive\Desktop\p.png")
95 l2=Label(base,image=photo)#l2=Label(root,bg="red")
96 l2.pack()
97
98
99 #Create Chat window
100 ChatLog = Text(base, bd=0, bg="#E6EDFA", height="8", width="50", font="Arial",)
101
102 ChatLog.config(state=DISABLED)
103
104 #Bind scrollbar to Chat window
105 scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
106 ChatLog['yscrollcommand'] = scrollbar.set
107
108 #Create Button to send message
109 SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
110 bd=0, bg="#986ECC", activebackground="#3c9d9b",fg='#ffffff',
111 command= send)
112
113 #Create the box to enter message
114 EntryBox = Text(base, bd=0, bg="#E3D1CD",width="29", height="5", font="Arial")
115 #EntryBox.bind("<Return>", send)
116
117
118 #Place all components on the screen
119 scrollbar.place(x=376,y=6, height=367)
120 ChatLog.place(x=6,y=6, height=367, width=370)
121 EntryBox.place(x=128, y=368, height=30, width=265)
122 SendButton.place(x=6, y=368, height=30)
123
124 base.mainloop()
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Variable explorer File explorer Help
IPython console
Console 2/A
47/47 [==============================] - 0s 0us/step - loss: 0.0746 - accuracy: 0.9787
Epoch 196/200
47/47 [==============================] - 0s 0us/step - loss: 0.0945 - accuracy: 0.9574
Epoch 197/200
47/47 [==============================] - 0s 565us/step - loss: 0.0690 - accuracy: 0.9574
Epoch 198/200
47/47 [==============================] - 0s 276us/step - loss: 0.0353 - accuracy: 1.0000
Epoch 199/200
47/47 [==============================] - 0s 276us/step - loss: 0.0878 - accuracy: 0.9574
Epoch 200/200
47/47 [==============================] - 0s 254us/step - loss: 0.0391 - accuracy: 1.0000
model created
In [24]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [25]: runfile('E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project/chatgui.py', wdir='E:/Kushagr/Semester 4/INT 404 Artificial Intelligence/Project')
In [26]:
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 93 Column: 18 Memory: 77 %


Chitty
You: Hi
Bot: Good to see you again
You: What can you do
Bot: I can help you to know some scerect of the life
Send
Tell me one life lesson


Chitty
You: Hi
Bot: Good to see you again
You: What can you do
Bot: I can tell you some life facts
You: Tell me one life lesson
Bot: Make improvements, not excuses. Seek respect, not attention.
You: Tell me one more life lesson
Bot: Life has no remote....get up and chang e it yourself!
You: Tell me something else
Send


Chitty
You: What can you do
Bot: I can tell you some life facts
You: Tell me one life lesson
Bot: Make improvements, not excuses. Seek respect, not attention.
You: Tell me one more life lesson
Bot: Life has no remote....get up and chang e it yourself!
You: Tell me something else
Bot: you are time killer
Send