

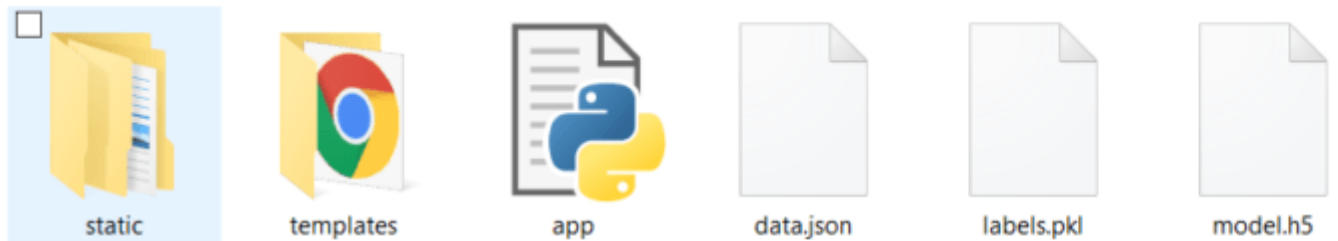
IBM PROJECT AI101

CHAT BOT USING PYTHON

CHAT BOT\_PHASE 3

## The Dataset

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



Now we are going to build the chatbot using Flask framework but first, let us see the file structure and the type of files we will be creating:

- **data.json** – The data file which has predefined patterns and responses.
- **training.py** – In this Python file, we wrote a script to build the model and train our chatbot.
- **Texts.pkl** – This is a pickle file in which we store the words Python object using Nltk that contains a list of our vocabulary.
- **Labels.pkl** – The classes pickle file contains the list of categories(Labels).
- **model.h5** – This is the trained model that contains information about the model and has weights of the neurons.
- **app.py** – This is the flask Python script in which we implemented web-based GUI for our chatbot. Users can easily interact with the bot.

## 1.Import and load the data file

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

The data file is in JSON format so we used the json package to parse the JSON file into Python. This is how our data.json file looks like.

```
{"intents": [  
  {"tag": "greeting",
```

```
"patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
"responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"],
"context": [""]
},
{"tag": "goodbye",
"patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
"responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
"context": [""]
},
{"tag": "thanks",
"patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
"responses": ["Happy to help!", "Any time!", "My pleasure"],
"context": [""]
},
{"tag": "noanswer",
"patterns": [],
"responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
"context": [""]
},
{"tag": "options",
"patterns": ["How you could help me?", "What you can do?", "What help you provide?", "How you can be helpful?", "What support is offered"],
"responses": ["I can guide you through Adverse drug reaction list, Blood pressure tracking, Hospitals and Pharmacies", "Offering support for Adverse drug reaction, Blood pressure, Hospitals and Pharmacies"],
"context": [""]
},
{"tag": "adverse_drug",
```

```
"patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a list of drugs causing adverse behavior", "List all drugs suitable for patient with adverse reaction", "Which drugs dont have adverse reaction?" ],
"responses": ["Navigating to Adverse drug reaction module"],
"context": [""]
},
{"tag": "blood_pressure",
"patterns": ["Open blood pressure module", "Task related to blood pressure", "Blood pressure data entry", "I want to log blood pressure results", "Blood pressure data management" ],
"responses": ["Navigating to Blood Pressure module"],
"context": [""]
},
{"tag": "blood_pressure_search",
"patterns": ["I want to search for blood pressure result history", "Blood pressure for patient", "Load patient blood pressure result", "Show blood pressure results for patient", "Find blood pressure results by ID" ],
"responses": ["Please provide Patient ID", "Patient ID?"],
"context": ["search_blood_pressure_by_patient_id"]
},
{"tag": "search_blood_pressure_by_patient_id",
"patterns": [],
"responses": ["Loading Blood pressure result for Patient"],
"context": [""]
},
{"tag": "pharmacy_search",
"patterns": ["Find me a pharmacy", "Find pharmacy", "List of pharmacies nearby", "Locate pharmacy", "Search pharmacy" ],
"responses": ["Please provide pharmacy name"],
"context": ["search_pharmacy_by_name"]
},
{"tag": "search_pharmacy_by_name",
"patterns": [],
"responses": ["Loading pharmacy details"],
```

```

"context": [""]
},
{"tag": "hospital_search",
"patterns": ["Lookup for hospital", "Searching for hospital to transfer patient", "I want to
search hospital data", "Hospital lookup for patient", "Looking up hospital details" ],
"responses": ["Please provide hospital name or location"],
"context": ["search_hospital_by_params"]
},
{"tag": "search_hospital_by_params",
"patterns": [],
"responses": ["Please provide hospital type"],
"context": ["search_hospital_by_type"]
},
{"tag": "search_hospital_by_type",
"patterns": [],
"responses": ["Loading hospital details"],
"context": [""]
}
]
}

```

## 2. Preprocess data

When working with text data, we need to perform various preprocessing on the data before design an ANN 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.

Here we iterate through the patterns and tokenize the sentence using `nltk.word_tokenize()` function and append each word in the words list. We also create a list of classes for our tags.

Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

## 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 'model.h5'.

**Training.py is shown below**

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('data.json').read()
intents = json.loads(data_file)
for intent in intents['intents']:
    for pattern in intent['patterns']:
        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))
```

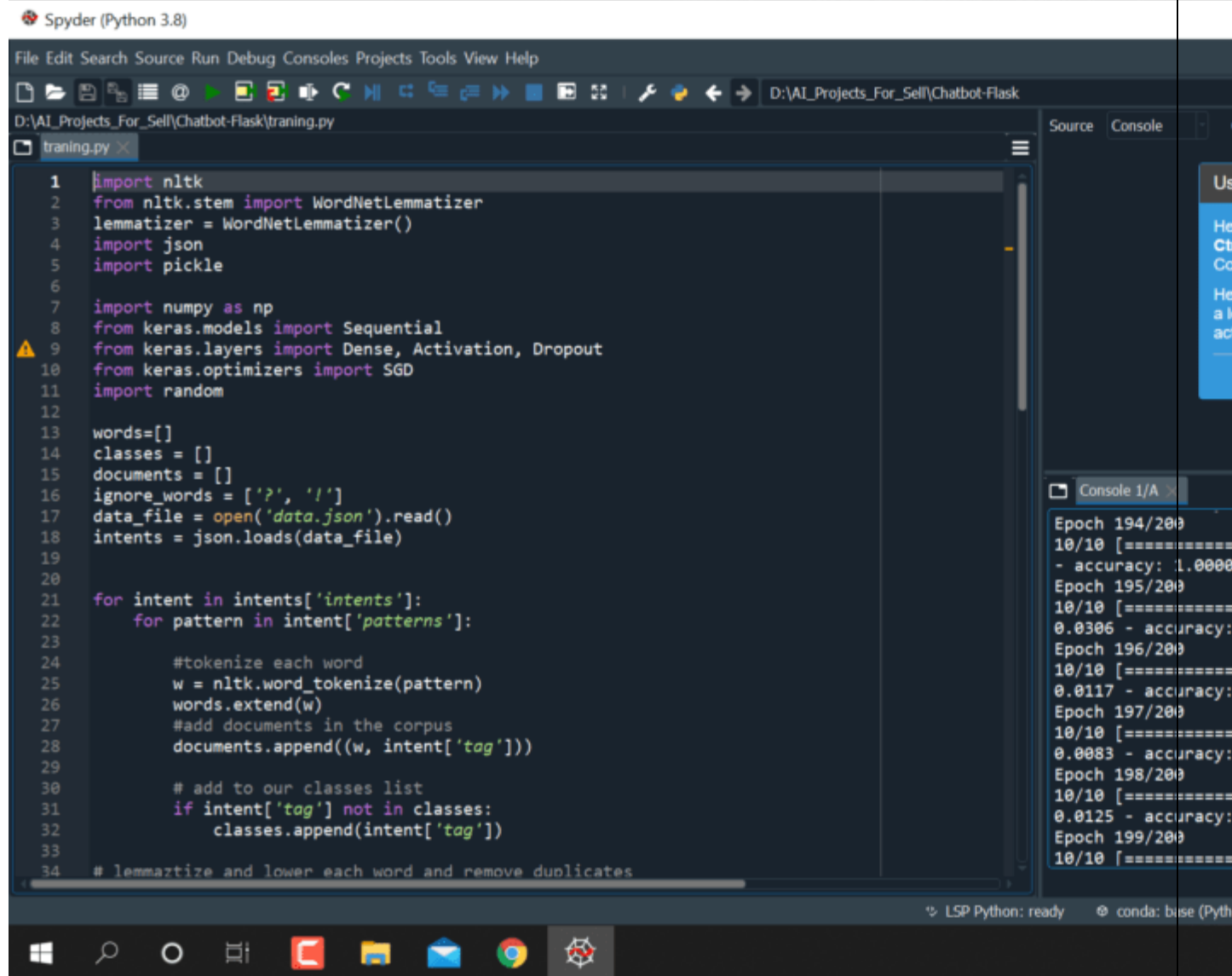
```

# add to our classes list
if intent['tag'] not in classes:
    classes.append(intent['tag'])
# lemmatize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)
pickle.dump(words,open('texts.pkl','wb'))
pickle.dump(classes,open('labels.pkl','wb'))
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # 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 of neurons
# 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('model.h5', hist)
print("model created")
```





```
1 import nltk
2 from nltk.stem import WordNetLemmatizer
3 lemmatizer = WordNetLemmatizer()
4 import json
5 import pickle
6
7 import numpy as np
8 from keras.models import Sequential
9 from keras.layers import Dense, Activation, Dropout
10 from keras.optimizers import SGD
11 import random
12
13 words=[]
14 classes = []
15 documents = []
16 ignore_words = ['?', '!']
17 data_file = open('data.json').read()
18 intents = json.loads(data_file)
19
20
21 for intent in intents['intents']:
22     for pattern in intent['patterns']:
23
24         #tokenize each word
25         w = nltk.word_tokenize(pattern)
26         words.extend(w)
27         #add documents in the corpus
28         documents.append((w, intent['tag']))
29
30         # add to our classes list
31         if intent['tag'] not in classes:
32             classes.append(intent['tag'])
33
34 # lemmatize and lower each word and remove duplicates
```

Console 1/A

```
Epoch 194/200
10/10 [=====
- accuracy: 1.0000
Epoch 195/200
10/10 [=====
0.0306 - accuracy:
Epoch 196/200
10/10 [=====
0.0117 - accuracy:
Epoch 197/200
10/10 [=====
0.0083 - accuracy:
Epoch 198/200
10/10 [=====
0.0125 - accuracy:
Epoch 199/200
10/10 [=====
```

Training-ANN Model

## 5. Predict the response (Flask web-based GUI)

Now to predict the sentences and get a response from the user to let us create a new file 'app.py' using flask web-based framework.

**Note for making flask app we need to make to folders name as static and templates and app.py files.**

- **static folder contains a subfolder with name styles. The styles folder contain css file with the name style.css**
- ***Templates folder HTML file with the name of index.html***
- **app.py file for run the flask app using IDE.**

```
:root {
--body-bg: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
--msger-bg: #fff;
--border: 2px solid #ddd;
--left-msg-bg: #ececec;
--right-msg-bg: #579ffb;
}
html {
box-sizing: border-box;
}
*,
*:before,
*:after {
margin: 0;
padding: 0;
box-sizing: inherit;
}
body {
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
background-image: var(--body-bg);
font-family: Helvetica, sans-serif;
}
.msger {
display: flex;
flex-flow: column wrap;
justify-content: space-between;
width: 100%;
max-width: 867px;
margin: 25px 10px;
```

```
height: calc(100% - 50px);
border: var(--border);
border-radius: 5px;
background: var(--msger-bg);
box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);
}

.msger-header {
/* display: flex; */
font-size: medium;
justify-content: space-between;
padding: 10px;
text-align: center;
border-bottom: var(--border);
background: #eee;
color: #666;
}

.msger-chat {
flex: 1;
overflow-y: auto;
padding: 10px;
}

.msger-chat::-webkit-scrollbar {
width: 6px;
}

.msger-chat::-webkit-scrollbar-track {
background: #ddd;
}

.msger-chat::-webkit-scrollbar-thumb {
background: #bdbdbd;
}

.msg {
display: flex;
align-items: flex-end;
```

```
margin-bottom: 10px;
}

.msg-img {
width: 50px;
height: 50px;
margin-right: 10px;
background: #ddd;
background-repeat: no-repeat;
background-position: center;
background-size: cover;
border-radius: 50%;
}

.msg-bubble {
max-width: 450px;
padding: 15px;
border-radius: 15px;
background: var(--left-msg-bg);
}

.msg-info {
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 10px;
}

.msg-info-name {
margin-right: 10px;
font-weight: bold;
}

.msg-info-time {
font-size: 0.85em;
}

.left-msg .msg-bubble {
border-bottom-left-radius: 0;
```

```
}  
  
.right-msg {  
flex-direction: row-reverse;  
}  
  
.right-msg .msg-bubble {  
background: var(--right-msg-bg);  
color: #fff;  
border-bottom-right-radius: 0;  
}  
  
.right-msg .msg-img {  
margin: 0 0 0 10px;  
}  
  
.msger-inputarea {  
display: flex;  
padding: 10px;  
border-top: var(--border);  
background: #eee;  
}  
  
.msger-inputarea * {  
padding: 10px;  
border: none;  
border-radius: 3px;  
font-size: 1em;  
}  
  
.msger-input {  
flex: 1;  
background: #ddd;  
}  
  
.msger-send-btn {  
margin-left: 10px;  
background: rgb(0, 196, 65);  
color: #fff;  
font-weight: bold;
```

```
cursor: pointer;
transition: background 0.23s;
}
.msger-send-btn:hover {
background: rgb(0, 180, 50);
}
.msger-chat {
background-color: #fcfcfe;
background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg'
width='260' height='260' viewBox='0 0 260 260'%3E%3Cg fill-rule='evenodd'%3E%3Cg
fill='%23dddddd' fill-opacity='0.4'%3E%3Cpath d='M24.37 16c.2.65.39 1.32.54 2H21.17l1.17
2.34.45.9-.24.11V28a5 5 0 0 1-2.23 8.94l-.02.06a8 8 0 0 1-7.75 6h-20a8 8 0 0 1-7.74-6l-.02-
.06A5 5 0 0 1-17.45 28v-6.76l-.79-1.58-.44-.9.9-.44.63-.32H-20a23.01 23.01 0 0 1 44.37-2zm-
36.82 2a1 1 0 0 0-.44.11-3.1 1.56.89 1.79 1.31-.66a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-
1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0
.86.02l2.88-1.27a3 3 0 0 1 2.43 0l2.88 1.27a1 1 0 0 0 .85-.02l3.1-1.55-.89-1.79-1.42.71a3 3 0 0
1-2.56.06l-2.77-1.23a1 1 0 0 0-.4-.09h-.01a1 1 0 0 0-.4.09l-2.78 1.23a3 3 0 0 1-2.56-.06l-2.3-
1.15a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.11L.9 19.22a3 3 0 0 1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-
.11h-.01a1 1 0 0 0-.44.11-2.21 1.11a3 3 0 0 1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-.11h-.01zm0-2h-
4.9a21.01 21.01 0 0 1 39.61 0h-2.09l-.06-.13-.26.13h-32.31zm30.35 7.68l1.36-.68h1.3v2h-36v-
1.15l.34-.17 1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69 0l1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69 0l2.26
23h2.59l1.36.68a3 3 0 0 0 2.56.06l1.67-.74h3.23l1.67.74a3 3 0 0 0 2.56-.06zM-13.82 27l16.37
4.91L18.93 27h-32.75zm-.63 2h.34l16.66 5 16.67-5h.33a3 3 0 1 1 0 6h-34a3 3 0 1 1 0-6zm1.35
8a6 6 0 0 0 5.65 4h20a6 6 0 0 0 5.66-4H-13.1z'/%3E%3Cpath id='path6_fill-copy' d='M284.37
16c.2.65.39 1.32.54 2H281.17l1.17 2.34.45.9-.24.11V28a5 5 0 0 1-2.23 8.94l-.02.06a8 8 0 0 1-
7.75 6h-20a8 8 0 0 1-7.74-6l-.02-.06a5 5 0 0 1-2.24-8.94v-6.76l-.79-1.58-.44-.9.9-.44.63-
.32H240a23.01 23.01 0 0 1 44.37-2zm-36.82 2a1 1 0 0 0-.44.11-3.1 1.56.89 1.79 1.31-.66a3 3 0 0
1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-1.1a3 3 0 0 1
2.69 0l2.2 1.1a1 1 0 0 0 .86.02l2.88-1.27a3 3 0 0 1 2.43 0l2.88 1.27a1 1 0 0 0 .85-.02l3.1-1.55-
.89-1.79-1.42.71a3 3 0 0 1-2.56.06l-2.77-1.23a1 1 0 0 0-.4-.09h-.01a1 1 0 0 0-.4.09l-2.78 1.23a3
3 0 0 1-2.56-.06l-2.3-1.15a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.11-2.21 1.11a3 3 0 0 1-2.69 0l-
2.2-1.1a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.11-2.21 1.11a3 3 0 0 1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-
.11h-.01zm0-2h-4.9a21.01 21.01 0 0 1 39.61 0h-2.09l-.06-.13-.26.13h-32.31zm30.35 7.68l1.36-
```

.68h1.3v2h-36v-1.15l.34-.17 1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69 0l1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69 0l1.36-.68h2.59l1.36.68a3 3 0 0 0 2.56.06l1.67-.74h3.23l1.67.74a3 3 0 0 0 2.56-.06zM246.18 27l16.37 4.9lL278.93 27h-32.75zm-.63 2h.34l16.66 5 16.67-5h.33a3 3 0 1 1 0 6h-34a3 3 0 1 1 0-6zm1.35 8a6 6 0 0 0 5.65 4h20a6 6 0 0 0 5.66-4H246.9z'/%3E%3Cpath d='M159.5 21.02A9 9 0 0 0 151 15h-42a9 9 0 0 0-8.5 6.02 6 6 0 0 0 .02 11.96A8.99 8.99 0 0 0 109 45h42a9 9 0 0 0 8.48-12.02 6 6 0 0 0 .02-11.96zM151 17h-42a7 7 0 0 0-6.33 4h54.66a7 7 0 0 0-6.33-4zm-9.34 26a8.98 8.98 0 0 0 3.34-7h-2a7 7 0 0 1-7 7h-4.34a8.98 8.98 0 0 0 3.34-7h-2a7 7 0 0 1-7 7h-4.34a8.98 8.98 0 0 0 3.34-7h-2a7 7 0 0 1-7 7h-7a7 7 0 1 1 0-14h42a7 7 0 1 1 0 14h-9.34zM109 27a9 9 0 0 0-7.48 4H101a4 4 0 1 1 0-8h58a4 4 0 0 1 0 8h-.52a9 9 0 0 0-7.48-4h-42z'/%3E%3Cpath d='M39 115a8 8 0 1 0 0-16 8 8 0 0 0 16zm6-8a6 6 0 1 1-12 0 6 6 0 0 1 12 0zm-3-29v-2h8v-6H40a4 4 0 0 0-4 4v10H22l-1.33 4-.67 2h2.19L26 130h26l3.81-40H58l-.67-2L56 84H42v-6zm-4-4v10h2V74h8v-2h-8a2 2 0 0 0-2 2zm2 12h14.56l.67 2H22.77l.67-2H40zm13.8 4H24.2l3.62 38h22.36l3.62-38z'/%3E%3Cpath d='M129 92h-6v4h-6v4h-6v14h-3l.24 2 3.76 32h36l3.76-32 .24-2h-3v-14h-6v-4h-6v-4h-8zm18 22v-12h-4v4h3v8h1zm-3 0v-6h-4v6h4zm-6 6v-16h-4v19.17c1.6-.7 2.97-1.8 4-3.17zm-6 3.8V100h-4v23.8a10.04 10.04 0 0 0 4 0zm-6-.63V104h-4v16a10.04 10.04 0 0 0 4 3.17zm-6-9.17v-6h-4v6h4zm-6 0v-8h3v-4h-4v12h1zm27-12v-4h-4v4h3v4h1v-4zm-6 0v-8h-4v4h3v4h1zm-6-4v-4h-4v8h1v-4h3zm-6 4v-4h-4v8h1v-4h3zm7 24a12 12 0 0 0 11.83-10h7.92l-3.53 30h-32.44l-3.53-30h7.92A12 12 0 0 0 130 126z'/%3E%3Cpath d='M212 86v2h-4v-2h4zm4 0h-2v2h2v-2zm-20 0v.1a5 5 0 0 0-.56 9.65l.06.25 1.12 4.48a2 2 0 0 0 1.94 1.52h.01l7.02 24.55a2 2 0 0 0 1.92 1.45h4.98a2 2 0 0 0 1.92-1.45l7.02-24.55a2 2 0 0 0 1.95-1.52L224.5 96l.06-.25a5 5 0 0 0-.56-9.65V86a14 14 0 0 0-28 0zm4 0h6v2h-9a3 3 0 1 0 0 6H223a3 3 0 1 0 0-6H220v-2h2a12 12 0 1 0-24 0h2zm-1.44 14l-1-4h24.88l-1 4h-22.88zm8.95 26l-6.86-24h18.7l-6.86 24h-4.98zM150 242a22 22 0 1 0 0-44 22 22 0 0 0 44zm24-22a24 24 0 1 1-48 0 24 24 0 0 1 48 0zm-28.38 17.73l2.04-.87a6 6 0 0 1 4.68 0l2.04.87a2 2 0 0 0 2.5-.82l1.14-1.9a6 6 0 0 1 3.79-2.75l2.15-.5a2 2 0 0 0 1.54-2.12l-.19-2.2a6 6 0 0 1 1.45-4.46l1.45-1.67a2 2 0 0 0 0-2.62l-1.45-1.67a6 6 0 0 1-1.45-4.46l.2-2.2a2 2 0 0 0-1.55-2.13l-2.15-.5a6 6 0 0 1-3.8-2.75l-1.13-1.9a2 2 0 0 0-2.5-.8l-2.04.86a6 6 0 0 1-4.68 0l-2.04-.87a2 2 0 0 0-2.5.82l-1.14 1.9a6 6 0 0 1-3.79 2.75l-2.15.5a2 2 0 0 0-1.54 2.12l.19 2.2a6 6 0 0 1-1.45 4.46l-1.45 1.67a2 2 0 0 0 0 2.62l1.45 1.67a6 6 0 0 1 1.45 4.46l-.2 2.2a2 2 0 0 0 1.55 2.13l2.15.5a6 6 0 0 1 3.8 2.75l1.13 1.9a2 2 0 0 0 2.5.8zm2.82.97a4 4 0 0 1 3.12 0l2.04.87a4 4 0 0 0 4.99-1.62l1.14-1.9a4 4 0 0 1 2.53-1.84l2.15-.5a4 4 0 0 0 3.09-4.24l-.2-2.2a4 4 0 0 1 .97-2.98l1.45-1.67a4 4 0 0 0 0-5.24l-1.45-1.67a4 4 0 0 1-.97-2.97l.2-2.2a4 4 0 0 0-3.09-4.25l-2.15-.5a4 4 0 0 1-2.53-1.84l-1.14-1.9a4 4 0 0 0-5-1.62l-2.03.87a4 4 0 0 1-3.12 0l-2.04-.87a4 4 0 0 0-

4.99 1.62l-1.14 1.9a4 4 0 0 1-2.53 1.84l-2.15.5a4 4 0 0 0-3.09 4.24l.2 2.2a4 4 0 0 1-.97 2.98l-  
1.45 1.67a4 4 0 0 0 5.24l1.45 1.67a4 4 0 0 1 .97 2.97l-.2 2.2a4 4 0 0 0 3.09 4.25l2.15.5a4 4 0 0  
1 2.53 1.84l1.14 1.9a4 4 0 0 0 5 1.62l2.03-.87zM152 207a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm6 2a1 1 0  
1 1 2 0 1 1 0 0 1-2 0zm-11 1a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm-6 0a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm3-  
5a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm-8 8a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm3 6a1 1 0 1 1 2 0 1 1 0 0 1-2  
0zm0 6a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm4 7a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm5-2a1 1 0 1 1 2 0 1 1 0 0  
1-2 0zm5 4a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm4-6a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm6-4a1 1 0 1 1 2 0 1 1 0  
0 1-2 0zm-4-3a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm4-3a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm-5-4a1 1 0 1 1 2 0 1  
1 0 0 1-2 0zm-24 6a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm16 5a5 5 0 1 0 0-10 5 5 0 0 0 10zm7-5a7 7 0 1  
1-14 0 7 7 0 0 1 14 0zm86-29a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm19 9a1 1 0 0 1 1-1h2a1 1 0 0 1  
0 2h-2a1 1 0 0 1-1-1zm-14 5a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm-25 1a1 1 0 0 0 0 2h2a1 1 0 0 0  
0-2h-2zm5 4a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm9 0a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-  
1-1zm15 1a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm12-2a1 1 0 0 0 0 2h2a1 1 0 0 0 0-  
2h-2zm-11-14a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-19 0a1 1 0 0 0 0 2h2a1 1 0 0 0  
0-2h-2zm6 5a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-25 15c0-.47.01-.94.03-1.4a5 5 0  
0 1-1.7-8 3.99 3.99 0 0 1 1.88-5.18 5 5 0 0 1 3.4-6.22 3 3 0 0 1 1.46-1.05 5 5 0 0 1 7.76-  
3.27A30.86 30.86 0 0 1 246 184c6.79 0 13.06 2.18 18.17 5.88a5 5 0 0 1 7.76 3.27 3 3 0 0 1 1.47  
1.05 5 5 0 0 1 3.4 6.22 4 4 0 0 1 1.87 5.18 4.98 4.98 0 0 1-1.7 8c.02.46.03.93.03 1.4v1h-62v-  
1zm.83-7.17a30.9 30.9 0 0 0-.62 3.57 3 3 0 0 1-.61-4.2c.37.28.78.49 1.23.63zm1.49-4.61c-  
.36.87-.68 1.76-.96 2.68a2 2 0 0 1-.21-3.71c.33.4.73.75 1.17 1.03zm2.32-4.54c-.54.86-1.03 1.76-  
1.49 2.68a3 3 0 0 1-.07-4.67 3 3 0 0 0 1.56 1.99zm1.14-1.7c.35-.5.72-.98 1.1-1.46a1 1 0 1 0-1.1  
1.45zm5.34-5.77c-1.03.86-2 1.79-2.9 2.77a3 3 0 0 0-1.11-.77 3 3 0 0 1 4-2zm42.66 2.77c-.9-.98-  
1.87-1.9-2.9-2.77a3 3 0 0 1 4.01 2 3 3 0 0 0-1.1.77zm1.34 1.54c.38.48.75.96 1.1 1.45a1 1 0 1 0-  
1.1-1.45zm3.73 5.84c-.46-.92-.95-1.82-1.5-2.68a3 3 0 0 0 1.57-1.99 3 3 0 0 1-.07 4.67zm1.8  
4.53c-.29-.9-.6-1.8-.97-2.67.44-.28.84-.63 1.17-1.03a2 2 0 0 1-.2 3.7zm1.14 5.51c-.14-1.21-.35-  
2.4-.62-3.57.45-.14.86-.35 1.23-.63a2.99 2.99 0 0 1-.6 4.2zM275 214a29 29 0 0 0-57.97  
0h57.96zM72.33 198.12c-.21-.32-.34-.7-.34-1.12v-12h-2v12a4.01 4.01 0 0 0 7.09 2.54c.57-  
.69.91-1.57.91-2.54v-12h-2v12a1.99 1.99 0 0 1-2 2 2 2 0 0 1-1.66-.88zM75 176c.38 0 .74-.04  
1.1-.12a4 4 0 0 0 6.19 2.4A13.94 13.94 0 0 1 84 185v24a6 6 0 0 1-6 6h-3v9a5 5 0 1 1-10 0v-9h-  
3a6 6 0 0 1-6-6v-24a14 14 0 0 1 14-14 5 5 0 0 0 5 5zm-17 15v12a1.99 1.99 0 0 0 1.22 1.84 2 2 0  
0 0 2.44-.72c.21-.32.34-.7.34-1.12v-12h2v12a3.98 3.98 0 0 1-5.35 3.77 3.98 3.98 0 0 1-.65-  
.3V209a4 4 0 0 0 4 4h16a4 4 0 0 0 4-4v-24c.01-1.53-.23-2.88-.72-4.17-.43.1-.87.16-1.28.17a6 6  
0 0 1-5.2-3 7 7 0 0 1-6.47-4.88A12 12 0 0 0 58 185v6zm9 24v9a3 3 0 1 0 6 0v-9h-



```

6z'/%3E%3Cpath d='M-17 191a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm19 9a1 1 0 0 1 1-1h2a1 1 0 0
1 0 2H3a1 1 0 0 1-1-1zm-14 5a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm-25 1a1 1 0 0 0 0 2h2a1 1 0 0
0 0-2h-2zm5 4a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm9 0a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0
1-1-1zm15 1a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm12-2a1 1 0 0 0 0 2h2a1 1 0 0 0 0-
2H4zm-11-14a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-19 0a1 1 0 0 0 0 2h2a1 1 0 0 0
0-2h-2zm6 5a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-25 15c0-.47.01-.94.03-1.4a5 5 0
0 1-1.7-8 3.99 3.99 0 0 1 1.88-5.18 5 5 0 0 1 3.4-6.22 3 3 0 0 1 1.46-1.05 5 5 0 0 1 7.76-
3.27A30.86 30.86 0 0 1-14 184c6.79 0 13.06 2.18 18.17 5.88a5 5 0 0 1 7.76 3.27 3 3 0 0 1 1.47
1.05 5 5 0 0 1 3.4 6.22 4 4 0 0 1 1.87 5.18 4.98 4.98 0 0 1-1.7 8c.02.46.03.93.03 1.4v1h-62v-
1zm.83-7.17a30.9 30.9 0 0 0-.62 3.57 3 3 0 0 1-.61-4.2c.37.28.78.49 1.23.63zm1.49-4.61c-
.36.87-.68 1.76-.96 2.68a2 2 0 0 1-.21-3.71c.33.47.75 1.17 1.03zm2.32-4.54c-.54.86-1.03 1.76-
1.49 2.68a3 3 0 0 1-.07-4.67 3 3 0 0 0 1.56 1.99zm1.14-1.7c.35-.57.98 1.1-1.46a1 1 0 1 0-1.1
1.45zm5.34-5.77c-1.03.86-2 1.79-2.9 2.77a3 3 0 0 0-1.11-.77 3 3 0 0 1 4-2zm42.66 2.77c-.9-.98-
1.87-1.9-2.9-2.77a3 3 0 0 1 4.01 2 3 3 0 0 0-1.1.77zm1.34 1.54c.38.48.75.96 1.1 1.45a1 1 0 1 0-
1.1-1.45zm3.73 5.84c-.46-.92-.95-1.82-1.5-2.68a3 3 0 0 0 1.57-1.99 3 3 0 0 1-.07 4.67zm1.8
4.53c-.29-.9-.6-1.8-.97-2.67.44-.28.84-.63 1.17-1.03a2 2 0 0 1-.2 3.7zm1.14 5.51c-.14-1.21-.35-
2.4-.62-3.57.45-.14.86-.35 1.23-.63a2.99 2.99 0 0 1-.6 4.2zM15 214a29 29 0 0 0-57.97
0h57.96z'/%3E%3C/g%3E%3C/g%3E%3C/svg%3E");
}

```

templates/index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Chatbot</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<link rel="stylesheet" href="{ { url_for('static', filename='styles/style.css') } }">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
</head>
<body>

```

```

<!-- partial:index.partial.html -->
<section class="msger">
  <header class="msger-header">
    <div class="msger-header-title">
      <i class="fas fa-bug"></i> Chatbot <i class="fas fa-bug"></i>
    </div>
  </header>
  <main class="msger-chat">
    <div class="msg left-msg">
      <div class="msg-img" style="background-image:
url(https://image.flaticon.com/icons/svg/327/327779.svg)"></div>
      <div class="msg-bubble">
        <div class="msg-info">
          <div class="msg-info-name">Chatbot</div>
          <div class="msg-info-time">12:45</div>
        </div>
        <div class="msg-text">
          Hi, welcome to ChatBot! Go ahead and send me a message.
        </div>
      </div>
    </div>
    </div>
  </main>
  <form class="msger-inputarea">
    <input type="text" class="msger-input" id="textInput" placeholder="Enter your message...">
    <button type="submit" class="msger-send-btn">Send</button>
  </form>
</section>
<!-- partial -->
<script src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>
<script>
const msgerForm = get(".msger-inputarea");
const msgerInput = get(".msger-input");
const msgerChat = get(".msger-chat");

```

```

// Icons made by Freepik from www.flaticon.com
const BOT_IMG = "https://image.flaticon.com/icons/svg/327/327779.svg";
const PERSON_IMG = "https://image.flaticon.com/icons/svg/145/145867.svg";
const BOT_NAME = " ChatBot";
const PERSON_NAME = "You";
msgerForm.addEventListener("submit", event => {
  event.preventDefault();
  const msgText = msgerInput.value;
  if (!msgText) return;
  appendMessage(PERSON_NAME, PERSON_IMG, "right", msgText);
  msgerInput.value = "";
  botResponse(msgText);
});
function appendMessage(name, img, side, text) {
  // Simple solution for small apps
  const msgHTML = `
<div class="msg ${side}-msg">
<div class="msg-img" style="background-image: url(${img})"></div>
<div class="msg-bubble">
<div class="msg-info">
<div class="msg-info-name">${name}</div>
<div class="msg-info-time">${formatDate(new Date())}</div>
</div>
<div class="msg-text">${text}</div>
</div>
</div>
`;
  msgerChat.insertAdjacentHTML("beforeend", msgHTML);
  msgerChat.scrollTop += 500;
}
function botResponse(rawText) {
  // Bot Response
  $.get("/get", { msg: rawText }).done(function (data) {

```

```

console.log(rawText);
console.log(data);
const msgText = data;
appendMessage(BOT_NAME, BOT_IMG, "left", msgText);
});
}
// Utils
function get(selector, root = document) {
return root.querySelector(selector);
}
function formatDate(date) {
const h = "0" + date.getHours();
const m = "0" + date.getMinutes();
return `${h.slice(-2)}:${m.slice(-2)}`;
}
</script>
</body>
</html>

```

app.py

```

import nltk
nltk.download('popular')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
from keras.models import load_model
model = load_model('model.h5')
import json
import random
intents = json.loads(open('data.json').read())
words = pickle.load(open('texts.pkl','rb'))
classes = pickle.load(open('labels.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
results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
# sort by strength of probability
results.sort(key=lambda x: x[1], reverse=True)
return_list = []
for r in results:
return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
return return_list

def getResponse(ints, intents_json):

```

```

tag = ints[0]['intent']
list_of_intents = intents_json['intents']
for i in list_of_intents:
    if(i['tag']== tag):
        result = random.choice(i['responses'])
        break
return result
def chatbot_response(msg):
    ints = predict_class(msg, model)
    res = getResponse(ints, intents)
    return res
from flask import Flask, render_template, request
app = Flask(__name__)
app.static_folder = 'static'
@app.route("/")
def home():
    return render_template("index.html")
@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return chatbot_response(userText)
if __name__ == "__main__":
    app.run()

```

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 a random response from the list of responses.

Again we import the necessary packages and load the 'texts.pkl' and 'labels.pkl' pickle files which we have created when we trained our model:

To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text preprocessing and then predict the class. After predicting the class, we will get a random response from the list of intents.

## OUTPUT:

