In [1]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in ./opt/anaconda3/lib/python3.9/site-packages
(2.11.0)
Requirement already satisfied: keras<2.12,>=2.11.0 in ./opt/anaconda3/lib/python3.9/site
-packages (from tensorflow) (2.11.0)
Requirement already satisfied: tensorboard<2.12,>=2.11 in ./opt/anaconda3/lib/python3.9/
site-packages (from tensorflow) (2.11.2)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in ./opt/anaconda3/lib/python3.9/site
-packages (from tensorflow) (0.4.0)
Requirement already satisfied: setuptools in ./opt/anaconda3/lib/python3.9/site-packages
(from tensorflow) (65.6.3)
Requirement already satisfied: h5py>=2.9.0 in ./opt/anaconda3/lib/python3.9/site-package
s (from tensorflow) (3.7.0)
Requirement already satisfied: typing-extensions>=3.6.6 in ./opt/anaconda3/lib/python3.
9/site-packages (from tensorflow) (4.4.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in ./opt/anaconda3/lib/python3.9/si
te-packages (from tensorflow) (3.19.6)
Requirement already satisfied: six>=1.12.0 in ./opt/anaconda3/lib/python3.9/site-package
s (from tensorflow) (1.16.0)
Requirement already satisfied: tensorflow-estimator<2.12,>=2.11.0 in ./opt/anaconda3/li
b/python3.9/site-packages (from tensorflow) (2.11.0)
Requirement already satisfied: libclang>=13.0.0 in ./opt/anaconda3/lib/python3.9/site-pa
ckages (from tensorflow) (15.0.6.1)
Requirement already satisfied: packaging in ./opt/anaconda3/lib/python3.9/site-packages
(from tensorflow) (22.0)
Requirement already satisfied: numpy>=1.20 in ./.local/lib/python3.9/site-packages (from
tensorflow) (1.21.6)
Requirement already satisfied: opt-einsum>=2.3.2 in ./opt/anaconda3/lib/python3.9/site-p
ackages (from tensorflow) (3.3.0)
Requirement already satisfied: flatbuffers>=2.0 in ./opt/anaconda3/lib/python3.9/site-pa
ckages (from tensorflow) (23.1.21)
Requirement already satisfied: absl-py>=1.0.0 in ./opt/anaconda3/lib/python3.9/site-pack
ages (from tensorflow) (1.4.0)
Requirement already satisfied: termcolor>=1.1.0 in ./opt/anaconda3/lib/python3.9/site-pa
ckages (from tensorflow) (2.2.0)
Requirement already satisfied: google-pasta>=0.1.1 in ./opt/anaconda3/lib/python3.9/site
-packages (from tensorflow) (0.2.0)
Requirement already satisfied: wrapt>=1.11.0 in ./opt/anaconda3/lib/python3.9/site-packa
ges (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in ./opt/anaconda3/l
ib/python3.9/site-packages (from tensorflow) (0.30.0)
Requirement already satisfied: astunparse>=1.6.0 in ./opt/anaconda3/lib/python3.9/site-p
ackages (from tensorflow) (1.6.3)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in ./opt/anaconda3/lib/python3.9/site
-packages (from tensorflow) (1.42.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in ./opt/anaconda3/lib/python3.9/site-
packages (from astunparse>=1.6.0->tensorflow) (0.37.1)
Requirement already satisfied: requests<3,>=2.21.0 in ./opt/anaconda3/lib/python3.9/site
-packages (from tensorboard<2.12,>=2.11->tensorflow) (2.28.1)
Requirement already satisfied: werkzeug>=1.0.1 in ./opt/anaconda3/lib/python3.9/site-pac
kages (from tensorboard<2.12,>=2.11->tensorflow) (2.0.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in ./opt/anaconda3/lib/p
ython3.9/site-packages (from tensorboard<2.12,>=2.11->tensorflow) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in ./opt/anaconda3/lib/python3.9/site-pac
kages (from tensorboard<2.12,>=2.11->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in ./opt/anaconda3/
lib/python3.9/site-packages (from tensorboard<2.12,>=2.11->tensorflow) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in ./opt/anaconda3/lib/pyth
on3.9/site-packages (from tensorboard<2.12,>=2.11->tensorflow) (1.8.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in ./opt/anaconda3/lib/python3.9/si
te-packages (from tensorboard<2.12,>=2.11->tensorflow) (2.6.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in ./opt/anaconda3/lib/python3.9/s
ite-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (4.2.2)
```

```
Requirement already satisfied: pyasn1-modules>=0.2.1 in ./opt/anaconda3/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in ./opt/anaconda3/lib/python3.9/site-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (4.7.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in ./opt/anaconda3/lib/python3.9/site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in ./opt/anaconda3/lib/python3.9/site-packages (from markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (4.11.3)
Requirement already satisfied: certifi>=2017.4.17 in ./opt/anaconda3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2022.12.7)
Requirement already satisfied: charset-normalizer<3,>=2 in ./opt/anaconda3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./opt/anaconda3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./opt/anaconda3/lib/python3.9/site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow) (1.26.14)
Requirement already satisfied: zipp>=0.5 in ./opt/anaconda3/lib/python3.9/site-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow) (3.11.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in ./opt/anaconda3/lib/python3.9/site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in ./opt/anaconda3/lib/python3.9/site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow) (3.2.2)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install keras
```

```
Requirement already satisfied: keras in ./opt/anaconda3/lib/python3.9/site-packages (2.11.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in ./.local/lib/python3.9/site-packages (3.8.1)
Requirement already satisfied: regex>=2021.8.3 in ./opt/anaconda3/lib/python3.9/site-packages (from nltk) (2022.7.9)
Requirement already satisfied: joblib in ./opt/anaconda3/lib/python3.9/site-packages (from nltk) (1.1.1)
Requirement already satisfied: click in ./opt/anaconda3/lib/python3.9/site-packages (from nltk) (8.0.4)
Requirement already satisfied: tqdm in ./opt/anaconda3/lib/python3.9/site-packages (from nltk) (4.64.1)
Note: you may need to restart the kernel to use updated packages.
```

firstly importing the necessary packages for our chatbot

then initialize the variables

In [4]:
```
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
```

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

words=[]
classes = []
documents = []
ignore_words = ['?', '!']

## Intents data file which has predefined patterns and responses.
data_file = open('/Users/jalilkhan/Downloads/chatbot/PredefinedPatternsResponses.json').
intents = json.loads(data_file)
```

```
2023-01-24 18:17:21.337240: I tensorflow/core/platform/cpu_feature_guard.cc:193] This Te
nsorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the
following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler fla
gs.
```

Preprocessing Data

Tokenizing is first thing to do on text data. Its process of breaking the whole text into small parts like words.

In [6]:
```
# iterate through the patterns and tokenize the sentence using nltk.word_tokenize() func
# also creating a list of classes for our tags.

import nltk
nltk.download('punkt')

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'])
```

```
[nltk_data] Downloading package punkt to /Users/jalilkhan/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

removing duplicate words from the list

then converting a word into its lemma form and creating a pickle file to store the Python objects which we will use while predicting.

In [7]:
```
import nltk
nltk.download('wordnet')

# lemmatize, 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('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/jalilkhan/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
94 documents
9 classes ['adverse_drug', 'blood_pressure', 'blood_pressure_search', 'goodbye', 'greeti
ng', 'hospital_search', 'options', 'pharmacy_search', 'thanks']
88 unique lemmatized words ["'s", ',', 'a', 'adverse', 'all', 'anyone', 'are', 'awesom
e', 'be', 'behavior', 'blood', 'by', 'bye', 'can', 'causing', 'chatting', 'check', 'coul
d', 'data', 'day', 'detail', 'do', 'dont', 'drug', 'entry', 'find', 'for', 'give', 'goo
d', 'goodbye', 'have', 'hello', 'help', 'helpful', 'helping', 'hey', 'hi', 'history', 'h
ola', 'hospital', 'how', 'i', 'id', 'is', 'later', 'list', 'load', 'locate', 'log', 'loo
king', 'lookup', 'management', 'me', 'module', 'nearby', 'next', 'nice', 'of', 'offere
d', 'open', 'patient', 'pharmacy', 'pressure', 'provide', 'reaction', 'related', 'resul
t', 'search', 'searching', 'see', 'show', 'suitable', 'support', 'task', 'thank', 'thank
s', 'that', 'there', 'till', 'time', 'to', 'transfer', 'up', 'want', 'what', 'which', 'w
ith', 'you']
```

Creatingtraining and testing data

creating the training data in which we will provide the input and the output. 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.

In [7]:
```
# 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")
```

```
Training data created
/var/folders/yg/h1y3gfrd6v9_sx5hgb2n7zh40000gn/T/ipykernel_24915/2748295590.py:24: Visib
```

```
leDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-
or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated.
If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  training = np.array(training)
```

Building the model

Now after having our training data ready we now build neural network with 3 layers by using keras squential API for this. we achieve 100 percent accuracy after training the model for 200 epochs.

In [8]:
```
# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd outp
# 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 go
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
model.save('chatbot_model.h5', hist)

print("model created")
```

```
Epoch 1/200
/Users/jalilkhan/opt/anaconda3/lib/python3.9/site-packages/keras/optimizers/optimizer_v
2/gradient_descent.py:114: UserWarning: The `lr` argument is deprecated, use `learning_r
ate` instead.
  super().__init__(name, **kwargs)
10/10 [==============================] - 1s 2ms/step - loss: 2.3047 - accuracy: 0.0426
Epoch 2/200
10/10 [==============================] - 0s 3ms/step - loss: 2.1770 - accuracy: 0.1915
Epoch 3/200
10/10 [==============================] - 0s 2ms/step - loss: 2.1124 - accuracy: 0.1915
Epoch 4/200
10/10 [==============================] - 0s 3ms/step - loss: 2.1004 - accuracy: 0.2553
Epoch 5/200
10/10 [==============================] - 0s 3ms/step - loss: 1.9998 - accuracy: 0.2766
Epoch 6/200
10/10 [==============================] - 0s 3ms/step - loss: 1.9416 - accuracy: 0.4681
Epoch 7/200
10/10 [==============================] - 0s 3ms/step - loss: 1.8567 - accuracy: 0.3617
Epoch 8/200
10/10 [==============================] - 0s 3ms/step - loss: 1.7770 - accuracy: 0.4468
Epoch 9/200
10/10 [==============================] - 0s 3ms/step - loss: 1.7114 - accuracy: 0.4255
Epoch 10/200
10/10 [==============================] - 0s 3ms/step - loss: 1.5572 - accuracy: 0.5319
Epoch 11/200
10/10 [==============================] - 0s 9ms/step - loss: 1.2932 - accuracy: 0.7872
Epoch 12/200
10/10 [==============================] - 0s 12ms/step - loss: 1.1800 - accuracy: 0.7447
Epoch 13/200
10/10 [==============================] - 0s 11ms/step - loss: 1.1956 - accuracy: 0.6596
Epoch 14/200
10/10 [==============================] - 0s 4ms/step - loss: 0.9897 - accuracy: 0.7872
Epoch 15/200
10/10 [==============================] - 0s 5ms/step - loss: 0.9196 - accuracy: 0.7021
```

```
Epoch 16/200
10/10 [==============================] - 0s 3ms/step - loss: 0.8394 - accuracy: 0.8298
Epoch 17/200
10/10 [==============================] - 0s 5ms/step - loss: 0.8628 - accuracy: 0.8298
Epoch 18/200
10/10 [==============================] - 0s 3ms/step - loss: 0.7498 - accuracy: 0.7872
Epoch 19/200
10/10 [==============================] - 0s 4ms/step - loss: 0.6723 - accuracy: 0.8511
Epoch 20/200
10/10 [==============================] - 0s 3ms/step - loss: 0.6384 - accuracy: 0.8511
Epoch 21/200
10/10 [==============================] - 0s 4ms/step - loss: 0.5791 - accuracy: 0.8511
Epoch 22/200
10/10 [==============================] - 0s 3ms/step - loss: 0.5615 - accuracy: 0.8511
Epoch 23/200
10/10 [==============================] - 0s 3ms/step - loss: 0.5774 - accuracy: 0.8511
Epoch 24/200
10/10 [==============================] - 0s 3ms/step - loss: 0.4023 - accuracy: 0.9149
Epoch 25/200
10/10 [==============================] - 0s 4ms/step - loss: 0.4617 - accuracy: 0.8298
Epoch 26/200
10/10 [==============================] - 0s 3ms/step - loss: 0.2762 - accuracy: 0.9574
Epoch 27/200
10/10 [==============================] - 0s 3ms/step - loss: 0.4445 - accuracy: 0.9149
Epoch 28/200
10/10 [==============================] - 0s 4ms/step - loss: 0.3768 - accuracy: 0.8936
Epoch 29/200
10/10 [==============================] - 0s 3ms/step - loss: 0.2981 - accuracy: 0.9574
Epoch 30/200
10/10 [==============================] - 0s 3ms/step - loss: 0.3863 - accuracy: 0.8511
Epoch 31/200
10/10 [==============================] - 0s 3ms/step - loss: 0.3319 - accuracy: 0.9149
Epoch 32/200
10/10 [==============================] - 0s 4ms/step - loss: 0.3221 - accuracy: 0.9574
Epoch 33/200
10/10 [==============================] - 0s 4ms/step - loss: 0.3684 - accuracy: 0.9149
Epoch 34/200
10/10 [==============================] - 0s 4ms/step - loss: 0.3548 - accuracy: 0.9149
Epoch 35/200
10/10 [==============================] - 0s 5ms/step - loss: 0.1519 - accuracy: 1.0000
Epoch 36/200
10/10 [==============================] - 0s 5ms/step - loss: 0.2108 - accuracy: 0.9362
Epoch 37/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1588 - accuracy: 0.9787
Epoch 38/200
10/10 [==============================] - 0s 4ms/step - loss: 0.2687 - accuracy: 0.8936
Epoch 39/200
10/10 [==============================] - 0s 3ms/step - loss: 0.2725 - accuracy: 0.9574
Epoch 40/200
10/10 [==============================] - 0s 3ms/step - loss: 0.2131 - accuracy: 0.8936
Epoch 41/200
10/10 [==============================] - 0s 5ms/step - loss: 0.1704 - accuracy: 0.9574
Epoch 42/200
10/10 [==============================] - 0s 4ms/step - loss: 0.1352 - accuracy: 0.9787
Epoch 43/200
10/10 [==============================] - 0s 2ms/step - loss: 0.2460 - accuracy: 0.9149
Epoch 44/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1901 - accuracy: 0.9787
Epoch 45/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1282 - accuracy: 0.9574
Epoch 46/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1509 - accuracy: 0.9574
Epoch 47/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1630 - accuracy: 1.0000
Epoch 48/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0776 - accuracy: 0.9787
```

```
Epoch 49/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1100 - accuracy: 0.9787
Epoch 50/200
10/10 [==============================] - 0s 2ms/step - loss: 0.2100 - accuracy: 0.9574
Epoch 51/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1149 - accuracy: 0.9787
Epoch 52/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0737 - accuracy: 1.0000
Epoch 53/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1384 - accuracy: 0.9362
Epoch 54/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0930 - accuracy: 0.9787
Epoch 55/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1364 - accuracy: 0.9787
Epoch 56/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0795 - accuracy: 0.9787
Epoch 57/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1433 - accuracy: 0.9574
Epoch 58/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0603 - accuracy: 1.0000
Epoch 59/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1249 - accuracy: 0.9574
Epoch 60/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1165 - accuracy: 1.0000
Epoch 61/200
10/10 [==============================] - 0s 2ms/step - loss: 0.1054 - accuracy: 0.9574
Epoch 62/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1233 - accuracy: 0.9574
Epoch 63/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1187 - accuracy: 0.9574
Epoch 64/200
10/10 [==============================] - 0s 3ms/step - loss: 0.1337 - accuracy: 0.9787
Epoch 65/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0787 - accuracy: 0.9787
Epoch 66/200
10/10 [==============================] - 0s 2ms/step - loss: 0.1175 - accuracy: 0.9574
Epoch 67/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0507 - accuracy: 1.0000
Epoch 68/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0527 - accuracy: 1.0000
Epoch 69/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0665 - accuracy: 0.9787
Epoch 70/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0646 - accuracy: 0.9787
Epoch 71/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0696 - accuracy: 1.0000
Epoch 72/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0685 - accuracy: 0.9787
Epoch 73/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0513 - accuracy: 1.0000
Epoch 74/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0633 - accuracy: 0.9787
Epoch 75/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0906 - accuracy: 0.9787
Epoch 76/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0484 - accuracy: 1.0000
Epoch 77/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0467 - accuracy: 0.9787
Epoch 78/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0264 - accuracy: 1.0000
Epoch 79/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0144 - accuracy: 1.0000
Epoch 80/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0654 - accuracy: 1.0000
Epoch 81/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0405 - accuracy: 1.0000
```

```
Epoch 82/200
10/10 [==============================] - 0s 2ms/step - loss: 0.1356 - accuracy: 0.9574
Epoch 83/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0351 - accuracy: 1.0000
Epoch 84/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0299 - accuracy: 1.0000
Epoch 85/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0704 - accuracy: 1.0000
Epoch 86/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0202 - accuracy: 1.0000
Epoch 87/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0532 - accuracy: 1.0000
Epoch 88/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0471 - accuracy: 0.9787
Epoch 89/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0846 - accuracy: 0.9787
Epoch 90/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0347 - accuracy: 1.0000
Epoch 91/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0351 - accuracy: 1.0000
Epoch 92/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0566 - accuracy: 1.0000
Epoch 93/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0753 - accuracy: 0.9787
Epoch 94/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0318 - accuracy: 1.0000
Epoch 95/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0462 - accuracy: 0.9787
Epoch 96/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0758 - accuracy: 0.9787
Epoch 97/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0564 - accuracy: 0.9787
Epoch 98/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0892 - accuracy: 0.9787
Epoch 99/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0274 - accuracy: 1.0000
Epoch 100/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0476 - accuracy: 0.9787
Epoch 101/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0602 - accuracy: 0.9787
Epoch 102/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0828 - accuracy: 0.9787
Epoch 103/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0073 - accuracy: 1.0000
Epoch 104/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0891 - accuracy: 0.9574
Epoch 105/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0757 - accuracy: 0.9787
Epoch 106/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0452 - accuracy: 1.0000
Epoch 107/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0527 - accuracy: 1.0000
Epoch 108/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0279 - accuracy: 1.0000
Epoch 109/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0508 - accuracy: 1.0000
Epoch 110/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0276 - accuracy: 1.0000
Epoch 111/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0363 - accuracy: 1.0000
Epoch 112/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0404 - accuracy: 1.0000
Epoch 113/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0481 - accuracy: 0.9787
Epoch 114/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0099 - accuracy: 1.0000
```

```
Epoch 115/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0678 - accuracy: 0.9787
Epoch 116/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0360 - accuracy: 1.0000
Epoch 117/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0806 - accuracy: 0.9787
Epoch 118/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0526 - accuracy: 0.9787
Epoch 119/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0676 - accuracy: 0.9787
Epoch 120/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0573 - accuracy: 0.9787
Epoch 121/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0233 - accuracy: 1.0000
Epoch 122/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0711 - accuracy: 0.9787
Epoch 123/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0464 - accuracy: 1.0000
Epoch 124/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0476 - accuracy: 0.9787
Epoch 125/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0162 - accuracy: 1.0000
Epoch 126/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0444 - accuracy: 1.0000
Epoch 127/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0337 - accuracy: 1.0000
Epoch 128/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0316 - accuracy: 1.0000
Epoch 129/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0220 - accuracy: 1.0000
Epoch 130/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0490 - accuracy: 0.9787
Epoch 131/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0212 - accuracy: 1.0000
Epoch 132/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0133 - accuracy: 1.0000
Epoch 133/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0183 - accuracy: 1.0000
Epoch 134/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0583 - accuracy: 1.0000
Epoch 135/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0285 - accuracy: 1.0000
Epoch 136/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0097 - accuracy: 1.0000
Epoch 137/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0239 - accuracy: 1.0000
Epoch 138/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0158 - accuracy: 1.0000
Epoch 139/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0249 - accuracy: 1.0000
Epoch 140/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0241 - accuracy: 1.0000
Epoch 141/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0453 - accuracy: 0.9787
Epoch 142/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0430 - accuracy: 0.9787
Epoch 143/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0217 - accuracy: 1.0000
Epoch 144/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0319 - accuracy: 1.0000
Epoch 145/200
10/10 [==============================] - 0s 2ms/step - loss: 0.1233 - accuracy: 0.9574
Epoch 146/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0108 - accuracy: 1.0000
Epoch 147/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0102 - accuracy: 1.0000
```

```
Epoch 148/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0263 - accuracy: 1.0000
Epoch 149/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0412 - accuracy: 0.9787
Epoch 150/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0156 - accuracy: 1.0000
Epoch 151/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0915 - accuracy: 0.9787
Epoch 152/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0170 - accuracy: 1.0000
Epoch 153/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0187 - accuracy: 1.0000
Epoch 154/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0102 - accuracy: 1.0000
Epoch 155/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0136 - accuracy: 1.0000
Epoch 156/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0376 - accuracy: 1.0000
Epoch 157/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0134 - accuracy: 1.0000
Epoch 158/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0098 - accuracy: 1.0000
Epoch 159/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0191 - accuracy: 1.0000
Epoch 160/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0088 - accuracy: 1.0000
Epoch 161/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0415 - accuracy: 1.0000
Epoch 162/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0218 - accuracy: 1.0000
Epoch 163/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0209 - accuracy: 1.0000
Epoch 164/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0096 - accuracy: 1.0000
Epoch 165/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0328 - accuracy: 1.0000
Epoch 166/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0130 - accuracy: 1.0000
Epoch 167/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0560 - accuracy: 0.9787
Epoch 168/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0264 - accuracy: 1.0000
Epoch 169/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0130 - accuracy: 1.0000
Epoch 170/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0056 - accuracy: 1.0000
Epoch 171/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0585 - accuracy: 1.0000
Epoch 172/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0052 - accuracy: 1.0000
Epoch 173/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0349 - accuracy: 1.0000
Epoch 174/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0032 - accuracy: 1.0000
Epoch 175/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0055 - accuracy: 1.0000
Epoch 176/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0039 - accuracy: 1.0000
Epoch 177/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0240 - accuracy: 1.0000
Epoch 178/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0091 - accuracy: 1.0000
Epoch 179/200
10/10 [==============================] - 0s 2ms/step - loss: 0.1023 - accuracy: 0.9574
Epoch 180/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0153 - accuracy: 1.0000
```

```
Epoch 181/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0186 - accuracy: 1.0000
Epoch 182/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0288 - accuracy: 1.0000
Epoch 183/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0104 - accuracy: 1.0000
Epoch 184/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0141 - accuracy: 1.0000
Epoch 185/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0178 - accuracy: 1.0000
Epoch 186/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0356 - accuracy: 0.9787
Epoch 187/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0293 - accuracy: 1.0000
Epoch 188/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0106 - accuracy: 1.0000
Epoch 189/200
10/10 [==============================] - 0s 3ms/step - loss: 0.0077 - accuracy: 1.0000
Epoch 190/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0053 - accuracy: 1.0000
Epoch 191/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0294 - accuracy: 1.0000
Epoch 192/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0291 - accuracy: 1.0000
Epoch 193/200
10/10 [==============================] - 0s 2ms/step - loss: 0.1196 - accuracy: 0.9787
Epoch 194/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0191 - accuracy: 1.0000
Epoch 195/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0410 - accuracy: 0.9787
Epoch 196/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0102 - accuracy: 1.0000
Epoch 197/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0074 - accuracy: 1.0000
Epoch 198/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0171 - accuracy: 1.0000
Epoch 199/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0324 - accuracy: 1.0000
Epoch 200/200
10/10 [==============================] - 0s 2ms/step - loss: 0.0699 - accuracy: 0.9787
model created
```

Now for pridicting the responce from the bot we will load trained model.

The model will only tell us the class it belongs to and then retrieve a response from the list of the responses.

In [11]:

```
#  'words.pkl' and 'classes.pkl' pickle files which we have created when we trained our

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('/Users/jalilkhan/Downloads/chatbot/PredefinedPatternsResponse
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

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.

In [13]:
```
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
```

After predicting the class,

we will get a random response from the list of PredefinedPatternsResponses .

In [15]:
```
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(text):
    ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res
```

Now we will develop a graphical user interface.

Here is the full source code for the GUI.

In [ ]:
```
#Creating GUI with tkinter
import tkinter
from tkinter import *


def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))

        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')

        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)

#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)

ChatLog.config(state=DISABLED)

#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
                    bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
                    command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")
#EntryBox.bind("<Return>", send)


#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)

base.mainloop()
```

```
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
```

In [ ]: