```
import numpy as np
import random
import json
import pickle
import tensorflow as tf
```

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation,Dropout
from tensorflow.keras.optimizers import SGD
```

```
lemmatizer = WordNetLemmatizer()
```

```
intents = json.loads(open('intents.json').read())
```

```
words = []
classes = []
documents = []
ignore_letters = ['?','!','.',',']
```

```
# loop through each sentence in our intents patterns
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # tokenize each word in the sentence
        word_list = nltk.word_tokenize(pattern)
        # add to our words list
        words.extend(word_list)
        # add to documents pair
        documents.append((word_list, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

```
# stem and lower each word
words = [lemmatizer.lemmatize(word) for word in words if word not in ignore_letters]
# remove duplicates and sort
words = sorted(set(words))
classes = sorted(set(classes))
```

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


# create training data
training = []
output_empty = [0] * len(classes)
for document in documents:
    bag = []
    word_patterns = document[0]
    word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns]
    for word in words:
         bag.append(1) if word in word_patterns else bag.append(0)

    output_row = list(output_empty)
    output_row[classes.index(document[1])] = 1
    training.append([bag,output_row])


random.shuffle(training)
training = np.array(training)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: VisibleDeprecationWar
```

```
x_train = list(training[:,0])
y_train = list(training[:,1])


model = Sequential()
model.add(Dense(128,input_shape=(len(x_train[0]),),activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(y_train[0]),activation='softmax'))


sgd = SGD(lr=0.01,decay=1e-6,momentum=0.9,nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: Us
  super(SGD, self).__init__(name, **kwargs)
```

```
hist = model.fit(np.array(x_train),np.array(y_train),epochs=200,batch_size=5,verbose=1)
model.save('chatbotmodel.h5',hist)
print("Done")
```

```
Epoch 1/200
8/8 [==============================] - 0s 3ms/step - loss: 2.2436 - accuracy: 0.10
Epoch 2/200
8/8 [==============================] - 0s 3ms/step - loss: 2.1568 - accuracy: 0.23
Epoch 3/200
```

```
8/8 [==============================] - 0s 3ms/step - loss: 2.1239 - accuracy: 0.31
Epoch 4/200
8/8 [==============================] - 0s 3ms/step - loss: 2.0548 - accuracy: 0.36
Epoch 5/200
8/8 [==============================] - 0s 3ms/step - loss: 1.9744 - accuracy: 0.36
Epoch 6/200
8/8 [==============================] - 0s 3ms/step - loss: 1.8741 - accuracy: 0.39
Epoch 7/200
8/8 [==============================] - 0s 3ms/step - loss: 1.9135 - accuracy: 0.31
Epoch 8/200
8/8 [==============================] - 0s 3ms/step - loss: 1.7164 - accuracy: 0.47
Epoch 9/200
8/8 [==============================] - 0s 3ms/step - loss: 1.7561 - accuracy: 0.31
Epoch 10/200
8/8 [==============================] - 0s 5ms/step - loss: 1.5702 - accuracy: 0.55
Epoch 11/200
8/8 [==============================] - 0s 4ms/step - loss: 1.6034 - accuracy: 0.50
Epoch 12/200
8/8 [==============================] - 0s 2ms/step - loss: 1.5603 - accuracy: 0.52
Epoch 13/200
8/8 [==============================] - 0s 2ms/step - loss: 1.3248 - accuracy: 0.71
Epoch 14/200
8/8 [==============================] - 0s 2ms/step - loss: 1.2275 - accuracy: 0.63
Epoch 15/200
8/8 [==============================] - 0s 2ms/step - loss: 1.3011 - accuracy: 0.63
Epoch 16/200
8/8 [==============================] - 0s 3ms/step - loss: 1.0794 - accuracy: 0.71
Epoch 17/200
8/8 [==============================] - 0s 3ms/step - loss: 1.1772 - accuracy: 0.65
Epoch 18/200
8/8 [==============================] - 0s 3ms/step - loss: 0.9289 - accuracy: 0.76
Epoch 19/200
8/8 [==============================] - 0s 4ms/step - loss: 0.9047 - accuracy: 0.78
Epoch 20/200
8/8 [==============================] - 0s 3ms/step - loss: 0.9041 - accuracy: 0.71
Epoch 21/200
8/8 [==============================] - 0s 3ms/step - loss: 0.6847 - accuracy: 0.84
Epoch 22/200
8/8 [==============================] - 0s 3ms/step - loss: 1.0853 - accuracy: 0.63
Epoch 23/200
8/8 [==============================] - 0s 2ms/step - loss: 0.7817 - accuracy: 0.78
Epoch 24/200
8/8 [==============================] - 0s 3ms/step - loss: 0.7053 - accuracy: 0.81
Epoch 25/200
8/8 [==============================] - 0s 3ms/step - loss: 0.7144 - accuracy: 0.76
Epoch 26/200
8/8 [==============================] - 0s 3ms/step - loss: 0.6477 - accuracy: 0.81
Epoch 27/200
8/8 [==============================] - 0s 3ms/step - loss: 0.5823 - accuracy: 0.86
Epoch 28/200
8/8 [==============================] - 0s 3ms/step - loss: 0.7393 - accuracy: 0.73
Epoch 29/200
8/8 [==============================] - 0s 2ms/step - loss: 0.6313 - accuracy: 0.78
```

```
### Chabot.py


import random
import json
import pickle
import numpy as np
import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import load_model


lemmatizer = lemmatizer = WordNetLemmatizer()
intents = json.loads(open('intents.json').read())


words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
model = load_model('chatbotmodel.h5')


def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word) for word in sentence_words]
    return sentence_words


def bag_of_words(sentence):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for w in sentence_words:
        for i, word in enumerate(words):
            if word == w:
                bag[i]=1
    return np.array(bag)


def predict_class(sentence):
    bow = bag_of_words(sentence)
    res = model.predict(np.array([bow]))[0]
    ERROR_THRESHOLD = 0.25
    result = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]

    result.sort(key=lambda x:x[1], reverse=True)
    return_list = []
    for r in result:
        return_list.append({'intent': classes[r[0]],'probability':str(r[1])})
    return_list.reverse() # Higher probability which is correct at last so reverse
    return return_list


def get_response(intents_list,intents_json):
    tag = intents_list[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
```

```
print("Go! Bot is running!")
while True:
    message = input("Me: ")
    ints = predict_class(message)
    res = get_response(ints,intents)
    print("Bot:",res)
    if(ints[0]['intent'] == 'goodbye'):
      break
print("End")
```

```
Go! Bot is running!
Me: Hey
Bot: Hi there, what can I do for you?
Me: What do you sell
Bot: We have Cold-Coffee,Tea,Brownie,Chocolicks,Frolicks,Smoothies,Thick Shake
Me: What s the amount
Bot: Cold-Coffee - 70,Tea - 20,Brownie - 50,Chocolicks - 100,Frolicks - 110,Smoothie
Me: Payment in cash is allowed?
Bot: We accept VISA, Mastercard and Paypal
Me: Okay...Thanks
Bot: Happy to help!
Me: How long does delivery take
Bot: Shipping takes 2-4 days
Me: okay.. What time are you guys open
Bot: 24/7
Me: Thank you so much
Bot: Any time!
Me: See you soon
Bot: Any time!
Me: Bye
Bot: My pleasure
Me: Goodbye
Bot: Hi there, what can I do for you?
Me:
```

Executing (7m 5s) Cell > raw_input() > _input_request() > recv() > recv_multipart()