```
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]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Unzipping corpora/omw-1.4.zip.
```

```
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'])
# model.compile(optimizer='adam',loss = tf.keras.losses.SparseCategoricalCrossentropy(from
```

```
/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
7/7 [==============================] - 1s 3ms/step - loss: 2.1264 - accuracy: 0.15
Epoch 2/200
7/7 [==============================] - 0s 4ms/step - loss: 2.1253 - accuracy: 0.21
```

```
Epoch 3/200
7/7 [==============================] - 0s 3ms/step - loss: 1.9818 - accuracy: 0.30
Epoch 4/200
7/7 [==============================] - 0s 3ms/step - loss: 1.9404 - accuracy: 0.36
Epoch 5/200
7/7 [==============================] - 0s 3ms/step - loss: 1.9398 - accuracy: 0.27
Epoch 6/200
7/7 [==============================] - 0s 5ms/step - loss: 1.8758 - accuracy: 0.33
Epoch 7/200
7/7 [==============================] - 0s 3ms/step - loss: 1.8533 - accuracy: 0.36
Epoch 8/200
7/7 [==============================] - 0s 3ms/step - loss: 1.7652 - accuracy: 0.39
Epoch 9/200
7/7 [==============================] - 0s 3ms/step - loss: 1.5980 - accuracy: 0.45
Epoch 10/200
7/7 [==============================] - 0s 3ms/step - loss: 1.6428 - accuracy: 0.45
Epoch 11/200
7/7 [==============================] - 0s 3ms/step - loss: 1.5692 - accuracy: 0.51
Epoch 12/200
7/7 [==============================] - 0s 3ms/step - loss: 1.4921 - accuracy: 0.51
Epoch 13/200
7/7 [==============================] - 0s 4ms/step - loss: 1.4473 - accuracy: 0.66
Epoch 14/200
7/7 [==============================] - 0s 3ms/step - loss: 1.3585 - accuracy: 0.54
Epoch 15/200
7/7 [==============================] - 0s 3ms/step - loss: 1.4007 - accuracy: 0.51
Epoch 16/200
7/7 [==============================] - 0s 3ms/step - loss: 1.2865 - accuracy: 0.63
Epoch 17/200
7/7 [==============================] - 0s 3ms/step - loss: 1.0598 - accuracy: 0.72
Epoch 18/200
7/7 [==============================] - 0s 3ms/step - loss: 1.0905 - accuracy: 0.63
Epoch 19/200
7/7 [==============================] - 0s 3ms/step - loss: 0.8701 - accuracy: 0.78
Epoch 20/200
7/7 [==============================] - 0s 3ms/step - loss: 1.1161 - accuracy: 0.60
Epoch 21/200
7/7 [==============================] - 0s 3ms/step - loss: 0.9585 - accuracy: 0.69
Epoch 22/200
7/7 [==============================] - 0s 3ms/step - loss: 0.9149 - accuracy: 0.75
Epoch 23/200
7/7 [==============================] - 0s 3ms/step - loss: 0.9260 - accuracy: 0.72
Epoch 24/200
7/7 [==============================] - 0s 3ms/step - loss: 0.7504 - accuracy: 0.72
Epoch 25/200
7/7 [==============================] - 0s 3ms/step - loss: 0.7953 - accuracy: 0.69
Epoch 26/200
7/7 [==============================] - 0s 3ms/step - loss: 0.6523 - accuracy: 0.81
Epoch 27/200
7/7 [==============================] - 0s 3ms/step - loss: 0.6172 - accuracy: 0.78
Epoch 28/200
7/7 [==============================] - 0s 3ms/step - loss: 0.7197 - accuracy: 0.78
Epoch 29/200
7/7 [==============================] - 0s 4ms/step - loss: 0.6459 - accuracy: 0.81
```

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 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("")
    ints = predict_class(message)
    res = get_response(ints,intents)
    print(res)
```

```
Go! Bot is running!
Hey
Hello, thanks for visiting
what items do you sell
We sell coffee and tea
can we do payment in cash
We accept most major credit cards, and Paypal
Do you accept Mastercard?
We accept VISA, Mastercard and Paypal
thank you
Happy to help!
till when i can accept my delivery
Delivery takes 2-4 days
also at what time are you guys open
24/7
okay...thanks a lot
My pleasure
bye! See you later
Have a nice day
```

Executing (14m 58s) Cell › raw_input() › _input_request() › recv() › recv_multipart()