ChatBot

July 17, 2022

[1]:
```
#2 Importing Relevant Libraries

import json
import string
import random
import pandas as pd
import seaborn as sns
import matplotlib as plt
import json
%matplotlib inline

import nltk
import numpy as np
from nltk.stem import WordNetLemmatizer
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout
nltk.download("punkt")
nltk.download("wordnet")
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[1]:
```
True
```

[2]:
```
with open('C:\\Users\\DELL\\Desktop\\Feynn Lab\\chatbot\\intents.json') as f:
    data = json.load(f)
df=pd.DataFrame(data)
```

[3]:
```
df.head()
```

[3]:

	intents
0	{'tag': 'google', 'patterns': ['google', 'sear…
1	{'tag': 'greeting', 'patterns': ['Hi there', '…

```
2  {'tag': 'goodbye', 'patterns': ['Bye', 'See yo…
3  {'tag': 'thanks', 'patterns': ['Thanks', 'Than…
4  {'tag': 'noanswer', 'patterns': [], 'responses…
```

[4]:
```
#4 Extracting data_X(features) and data_Y(Target)

words = [] #For Bow model/ vocabulary for patterns
classes = [] #For Bow  model/ vocabulary for tags
data_X = [] #For storing each pattern
data_y = [] #For storing tag corresponding to each pattern in data_X
# Iterating over all the intents

for intent in data["intents"]:
    for pattern in intent["patterns"]:
        tokens = nltk.word_tokenize(pattern) # tokenize each pattern
        words.extend(tokens) #and append tokens to words
        data_X.append(pattern) #appending pattern to data_X
        data_y.append(intent["tag"]) ,# appending the associated tag to each␣
 ↪pattern

    # adding the tag to the classes if it's not there already
    if intent["tag"] not in classes:
        classes.append(intent["tag"])

# initializing lemmatizer to get stem of words
lemmatizer = WordNetLemmatizer()

# lemmatize all the words in the vocab and convert them to lowercase
# if the words don't appear in punctuation
words = [lemmatizer.lemmatize(word.lower()) for word in words if word not in␣
 ↪string.punctuation]
# sorting the vocab and classes in alphabetical order and taking the # set to␣
 ↪ensure no duplicates occur
words = sorted(set(words))
classes = sorted(set(classes))
```

[5]:
```
# 5 Text to Numbers
training = []
out_empty = [0] * len(classes)
# creating the bag of words model
for idx, doc in enumerate(data_X):
    bow = []
    text = lemmatizer.lemmatize(doc.lower())
    for word in words:
        bow.append(1) if word in text else bow.append(0)
    # mark the index of class that the current pattern is associated
    # to
```

```
    output_row = list(out_empty)
    output_row[classes.index(data_y[idx])] = 1
    # add the one hot encoded BoW and associated classes to training
    training.append([bow, output_row])
# shuffle the data and convert it to an array
random.shuffle(training)
training = np.array(training, dtype=object)
# split the features and target labels
train_X = np.array(list(training[:, 0]))
train_Y = np.array(list(training[:, 1]))
```

[6]:
```
#6 The Neural Network Model
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"))
adam = tf.keras.optimizers.Adam(learning_rate=0.01, decay=1e-6)
model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=["accuracy"])
print(model.summary())
model.fit(x=train_X, y=train_Y, epochs=150, verbose=1)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               16640

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 31)                2015

=================================================================
Total params: 26,911
Trainable params: 26,911
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/150
4/4 [==============================] - 1s 3ms/step - loss: 3.4766 - accuracy:
```

```
0.0265
Epoch 2/150
4/4 [==============================] - 0s 67ms/step - loss: 3.2969 - accuracy:
0.0885
Epoch 3/150
4/4 [==============================] - 0s 4ms/step - loss: 3.0950 - accuracy:
0.1504
Epoch 4/150
4/4 [==============================] - 0s 4ms/step - loss: 2.9787 - accuracy:
0.1504
Epoch 5/150
4/4 [==============================] - 0s 3ms/step - loss: 2.8133 - accuracy:
0.1947
Epoch 6/150
4/4 [==============================] - 0s 3ms/step - loss: 2.5796 - accuracy:
0.2478
Epoch 7/150
4/4 [==============================] - 0s 3ms/step - loss: 2.3034 - accuracy:
0.3274
Epoch 8/150
4/4 [==============================] - 0s 3ms/step - loss: 2.0540 - accuracy:
0.4425
Epoch 9/150
4/4 [==============================] - 0s 3ms/step - loss: 1.8088 - accuracy:
0.4956
Epoch 10/150
4/4 [==============================] - 0s 3ms/step - loss: 1.5710 - accuracy:
0.5398
Epoch 11/150
4/4 [==============================] - 0s 3ms/step - loss: 1.3302 - accuracy:
0.6106
Epoch 12/150
4/4 [==============================] - 0s 3ms/step - loss: 1.2878 - accuracy:
0.6195
Epoch 13/150
4/4 [==============================] - 0s 4ms/step - loss: 1.1360 - accuracy:
0.6903
Epoch 14/150
4/4 [==============================] - 0s 4ms/step - loss: 1.0222 - accuracy:
0.7345
Epoch 15/150
4/4 [==============================] - 0s 3ms/step - loss: 0.8963 - accuracy:
0.7522
Epoch 16/150
4/4 [==============================] - 0s 3ms/step - loss: 0.9060 - accuracy:
0.7168
Epoch 17/150
4/4 [==============================] - 0s 3ms/step - loss: 0.7309 - accuracy:
```

```
0.7965
Epoch 18/150
4/4 [==============================] - 0s 3ms/step - loss: 0.7053 - accuracy:
0.7611
Epoch 19/150
4/4 [==============================] - 0s 3ms/step - loss: 0.6361 - accuracy:
0.8319
Epoch 20/150
4/4 [==============================] - 0s 3ms/step - loss: 0.6218 - accuracy:
0.8053
Epoch 21/150
4/4 [==============================] - 0s 3ms/step - loss: 0.5576 - accuracy:
0.8319
Epoch 22/150
4/4 [==============================] - 0s 3ms/step - loss: 0.4905 - accuracy:
0.8319
Epoch 23/150
4/4 [==============================] - 0s 3ms/step - loss: 0.5480 - accuracy:
0.8230
Epoch 24/150
4/4 [==============================] - 0s 4ms/step - loss: 0.5533 - accuracy:
0.8319
Epoch 25/150
4/4 [==============================] - 0s 3ms/step - loss: 0.6704 - accuracy:
0.7876
Epoch 26/150
4/4 [==============================] - 0s 4ms/step - loss: 0.5061 - accuracy:
0.8319
Epoch 27/150
4/4 [==============================] - 0s 4ms/step - loss: 0.4582 - accuracy:
0.8938
Epoch 28/150
4/4 [==============================] - 0s 4ms/step - loss: 0.3852 - accuracy:
0.8673
Epoch 29/150
4/4 [==============================] - 0s 3ms/step - loss: 0.5360 - accuracy:
0.8496
Epoch 30/150
4/4 [==============================] - 0s 3ms/step - loss: 0.5591 - accuracy:
0.8496
Epoch 31/150
4/4 [==============================] - 0s 4ms/step - loss: 0.3475 - accuracy:
0.9027
Epoch 32/150
4/4 [==============================] - 0s 4ms/step - loss: 0.3806 - accuracy:
0.8319
Epoch 33/150
4/4 [==============================] - 0s 3ms/step - loss: 0.4042 - accuracy:
```

```
0.8938
Epoch 34/150
4/4 [==============================] - 0s 3ms/step - loss: 0.4059 - accuracy:
0.8584
Epoch 35/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3795 - accuracy:
0.8761
Epoch 36/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3456 - accuracy:
0.9115
Epoch 37/150
4/4 [==============================] - 0s 3ms/step - loss: 0.4435 - accuracy:
0.8230
Epoch 38/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3112 - accuracy:
0.9204
Epoch 39/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2790 - accuracy:
0.9027
Epoch 40/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3399 - accuracy:
0.8850
Epoch 41/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2639 - accuracy:
0.9204
Epoch 42/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2377 - accuracy:
0.9381
Epoch 43/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3288 - accuracy:
0.9115
Epoch 44/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1974 - accuracy:
0.9381
Epoch 45/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3088 - accuracy:
0.9027
Epoch 46/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3310 - accuracy:
0.8938
Epoch 47/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3203 - accuracy:
0.9027
Epoch 48/150
4/4 [==============================] - 0s 4ms/step - loss: 0.2436 - accuracy:
0.8938
Epoch 49/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3584 - accuracy:
```

```
0.9027
Epoch 50/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3175 - accuracy:
0.8938
Epoch 51/150
4/4 [==============================] - 0s 4ms/step - loss: 0.2084 - accuracy:
0.9204
Epoch 52/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1280 - accuracy:
0.9823
Epoch 53/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1497 - accuracy:
0.9558
Epoch 54/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3968 - accuracy:
0.8584
Epoch 55/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3213 - accuracy:
0.9115
Epoch 56/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1985 - accuracy:
0.9292
Epoch 57/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2211 - accuracy:
0.9292
Epoch 58/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1761 - accuracy:
0.9381
Epoch 59/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2068 - accuracy:
0.9558
Epoch 60/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2179 - accuracy:
0.9292
Epoch 61/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2282 - accuracy:
0.9204
Epoch 62/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1552 - accuracy:
0.9646
Epoch 63/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2171 - accuracy:
0.9292
Epoch 64/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1855 - accuracy:
0.9292
Epoch 65/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3164 - accuracy:
```

```
0.9204
Epoch 66/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1538 - accuracy:
0.9381
Epoch 67/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1666 - accuracy:
0.9469
Epoch 68/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1641 - accuracy:
0.9381
Epoch 69/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1388 - accuracy:
0.9558
Epoch 70/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1343 - accuracy:
0.9558
Epoch 71/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2585 - accuracy:
0.8850
Epoch 72/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2395 - accuracy:
0.9204
Epoch 73/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1390 - accuracy:
0.9469
Epoch 74/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2697 - accuracy:
0.9204
Epoch 75/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1937 - accuracy:
0.9469
Epoch 76/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2465 - accuracy:
0.9292
Epoch 77/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1559 - accuracy:
0.9292
Epoch 78/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1098 - accuracy:
0.9646
Epoch 79/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1388 - accuracy:
0.9558
Epoch 80/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2267 - accuracy:
0.9115
Epoch 81/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1772 - accuracy:
```

```
0.9469
Epoch 82/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2366 - accuracy:
0.9027
Epoch 83/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1238 - accuracy:
0.9646
Epoch 84/150
4/4 [==============================] - 0s 3ms/step - loss: 0.0978 - accuracy:
0.9646
Epoch 85/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1689 - accuracy:
0.9292
Epoch 86/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2328 - accuracy:
0.9292
Epoch 87/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2743 - accuracy:
0.9115
Epoch 88/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2432 - accuracy:
0.9292
Epoch 89/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2903 - accuracy:
0.9204
Epoch 90/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1528 - accuracy:
0.9381
Epoch 91/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1206 - accuracy:
0.9646
Epoch 92/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2663 - accuracy:
0.9115
Epoch 93/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1555 - accuracy:
0.9469
Epoch 94/150
4/4 [==============================] - 0s 3ms/step - loss: 0.0853 - accuracy:
0.9823
Epoch 95/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2138 - accuracy:
0.9292
Epoch 96/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3256 - accuracy:
0.9204
Epoch 97/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1966 - accuracy:
```

```
0.9115
Epoch 98/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1902 - accuracy:
0.9292
Epoch 99/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1743 - accuracy:
0.9381
Epoch 100/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1732 - accuracy:
0.9469
Epoch 101/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2914 - accuracy:
0.8938
Epoch 102/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2341 - accuracy:
0.9204
Epoch 103/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2695 - accuracy:
0.9292
Epoch 104/150
4/4 [==============================] - 0s 2ms/step - loss: 0.2137 - accuracy:
0.9204
Epoch 105/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1294 - accuracy:
0.9381
Epoch 106/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1352 - accuracy:
0.9646
Epoch 107/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1259 - accuracy:
0.9558
Epoch 108/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1516 - accuracy:
0.9646
Epoch 109/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1286 - accuracy:
0.9646
Epoch 110/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1662 - accuracy:
0.9292
Epoch 111/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1821 - accuracy:
0.9292
Epoch 112/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1591 - accuracy:
0.9558
Epoch 113/150
4/4 [==============================] - 0s 3ms/step - loss: 0.0988 - accuracy:
```

```
0.9735
Epoch 114/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1529 - accuracy:
0.9469
Epoch 115/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1299 - accuracy:
0.9469
Epoch 116/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1683 - accuracy:
0.9381
Epoch 117/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1496 - accuracy:
0.9381
Epoch 118/150
4/4 [==============================] - 0s 4ms/step - loss: 0.1693 - accuracy:
0.9381
Epoch 119/150
4/4 [==============================] - 0s 4ms/step - loss: 0.1593 - accuracy:
0.9558
Epoch 120/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1560 - accuracy:
0.9381
Epoch 121/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1395 - accuracy:
0.9469
Epoch 122/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2382 - accuracy:
0.9204
Epoch 123/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2042 - accuracy:
0.9381
Epoch 124/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1784 - accuracy:
0.9558
Epoch 125/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1038 - accuracy:
0.9646
Epoch 126/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1822 - accuracy:
0.9558
Epoch 127/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1139 - accuracy:
0.9558
Epoch 128/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1004 - accuracy:
0.9646
Epoch 129/150
4/4 [==============================] - 0s 3ms/step - loss: 0.3116 - accuracy:
```

```
0.9204
Epoch 130/150
4/4 [==============================] - 0s 3ms/step - loss: 0.0576 - accuracy:
0.9823
Epoch 131/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1512 - accuracy:
0.9204
Epoch 132/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2095 - accuracy:
0.9558
Epoch 133/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1993 - accuracy:
0.9381
Epoch 134/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2598 - accuracy:
0.9027
Epoch 135/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1917 - accuracy:
0.9469
Epoch 136/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1646 - accuracy:
0.9558
Epoch 137/150
4/4 [==============================] - 0s 3ms/step - loss: 0.0828 - accuracy:
0.9646
Epoch 138/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1022 - accuracy:
0.9735
Epoch 139/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1317 - accuracy:
0.9646
Epoch 140/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1561 - accuracy:
0.9381
Epoch 141/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2035 - accuracy:
0.9381
Epoch 142/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1063 - accuracy:
0.9735
Epoch 143/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1582 - accuracy:
0.9469
Epoch 144/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1386 - accuracy:
0.9558
Epoch 145/150
4/4 [==============================] - 0s 3ms/step - loss: 0.0867 - accuracy:
```

```
0.9823
Epoch 146/150
4/4 [==============================] - 0s 3ms/step - loss: 0.2823 - accuracy:
0.9292
Epoch 147/150
4/4 [==============================] - 0s 2ms/step - loss: 0.1642 - accuracy:
0.9381
Epoch 148/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1239 - accuracy:
0.9558
Epoch 149/150
4/4 [==============================] - 0s 3ms/step - loss: 0.1378 - accuracy:
0.9381
Epoch 150/150
4/4 [==============================] - 0s 3ms/step - loss: 0.0830 - accuracy:
0.9646
```

[6]: `<keras.callbacks.History at 0x181981c1bd0>`

[7]:
```python
#7 Preprocessing the Input

def clean_text(text):
  tokens = nltk.word_tokenize(text)
  tokens = [lemmatizer.lemmatize(word) for word in tokens]
  return tokens

def bag_of_words(text, vocab):
  tokens = clean_text(text)
  bow = [0] * len(vocab)
  for w in tokens:
    for idx, word in enumerate(vocab):
      if word == w:
        bow[idx] = 1
  return np.array(bow)

def pred_class(text, vocab, labels):
  bow = bag_of_words(text, vocab)
  result = model.predict(np.array([bow]))[0] #Extracting probabilities
  thresh = 0.5
  y_pred = [[indx, res] for indx, res in enumerate(result) if res > thresh]
  y_pred.sort(key=lambda x: x[1], reverse=True) #Sorting by values of␣
 ↪probability in decreasing order
  return_list = []
  for r in y_pred:
    return_list.append(labels[r[0]]) #Contains labels(tags) for highest␣
 ↪probability
  return return_list
```

```
def get_response(intents_list, intents_json):
  if len(intents_list) == 0:
    result = "Sorry! I don't understand."
  else:
    tag = intents_list[0]
    list_of_intents = intents_json["intents"]
    for i in list_of_intents:
      if i["tag"] == tag:
        result = random.choice(i["responses"])
        break
  return result
```

[8]:
```
# Interacting with the chatbot
print("Press 0 if you don't want to chat with our ChatBot.")
while True:
    message = input("")
    if message == "0":
      break
    intents = pred_class(message, words, classes)
    result = get_response(intents, data)
    print(result)
```

```
Press 0 if you don't want to chat with our ChatBot.
hi
Hello
how are you?
All good..What about you?
0
```

[ ]: