**IHC – Conversational Experiences**

**Industry Assignment No. 1: To build a hotel booking chatbot.**

import json

import numpy as np

from tensorflow import keras

from tensorflow.keras.layers import  LSTM, Dense, Embedding

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from sklearn.preprocessing import LabelEncoder

from keras.models import Sequential

from tensorflow.keras.layers import RepeatVector

import colorama

colorama.init()

from colorama import Fore, Style

with open('intents.json') as file:

    data = json.load(file)

training\_sentences = []

training\_labels = []

labels = []

responses = []

for intent in data['intents']:

    for pattern in intent['patterns']:

        training\_sentences.append(pattern)

        training\_labels.append(intent['tag'])

    responses.append(intent['responses'])

    if intent['tag'] not in labels:

        labels.append(intent['tag'])

num\_class = len(labels)

num\_class

lbl\_encoder = LabelEncoder()

lbl\_encoder.fit(training\_labels)

training\_labels = lbl\_encoder.transform(training\_labels)

from keras.utils import to\_categorical

# Convert labels to one-hot encoded format

training\_labels = to\_categorical(training\_labels, num\_class)

vocab\_size = 1000

embedding\_dim = 16

max\_len = 20

oov\_token = "<OOV>"

tokenizer = Tokenizer(num\_words=vocab\_size, oov\_token=oov\_token)

tokenizer.fit\_on\_texts(training\_sentences)

word\_index = tokenizer.word\_index

sequences = tokenizer.texts\_to\_sequences(training\_sentences)

padded\_sequences = pad\_sequences(sequences, truncating='post', maxlen=max\_len)

model = Sequential()

model.add(Embedding(vocab\_size, embedding\_dim, input\_length=max\_len))

model.add(GlobalAveragePooling1D())

model.add(Reshape((1, embedding\_dim)))

model.add(LSTM(16, activation='relu'))

model.add(RepeatVector(max\_len))

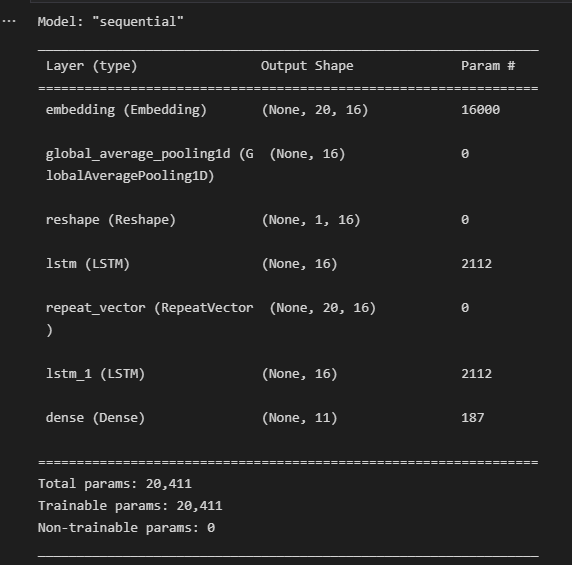
model.add(LSTM(16, activation='relu'))

model.add(Dense(num\_class, activation='softmax'))

model.compile(loss='binary\_crossentropy',

              optimizer='adam', metrics=['accuracy'])

model.summary()

****

epochs = 100

history = model.fit(padded\_sequences, training\_labels, epochs=epochs)

Epoch 1/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1205 - accuracy: 0.7407

Epoch 2/100 3/3 [==============================] - 0s 7ms/step - loss: 0.1192 - accuracy: 0.7407

Epoch 3/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1190 - accuracy: 0.7407

Epoch 4/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1182 - accuracy: 0.7407

Epoch 5/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1175 - accuracy: 0.7407

Epoch 6/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1167 - accuracy: 0.7654

Epoch 7/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1162 - accuracy: 0.7778

Epoch 8/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1151 - accuracy: 0.7778

Epoch 9/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1143 - accuracy: 0.7778

Epoch 10/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1135 - accuracy: 0.7778

Epoch 11/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1133 - accuracy: 0.7778

Epoch 12/100 3/3 [==============================] - 0s 8ms/step - loss: 0.1131 - accuracy: 0.7778 Epoch 13/100

...

Epoch 99/100 3/3 [==============================] - 0s 7ms/step - loss: 0.0691 - accuracy: 0.8519

Epoch 100/100 3/3 [==============================] - 0s 7ms/step - loss: 0.0661 - accuracy: 0.8519

model.save("Fchat\_model")

import pickle

# to save the fitted tokenizer

with open('tokenizer1.pickle', 'wb') as handle:

    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST\_PROTOCOL)

# to save the fitted label encoder

with open('label\_encoder1.pickle', 'wb') as ecn\_file:

    pickle.dump(lbl\_encoder, ecn\_file, protocol=pickle.HIGHEST\_PROTOCOL)

import json

import numpy as np

from tensorflow import keras

from sklearn.preprocessing import LabelEncoder

import colorama

colorama.init()

from colorama import Fore, Style, Back

import random

import pickle

with open("intents.json") as file:

    data = json.load(file)

def chat():

    # Load trained model

    model = keras.models.load\_model('Fchat\_model')

    # Load tokenizer object

    with open('tokenizer1.pickle', 'rb') as handle:

        tokenizer = pickle.load(handle)

    # Load label encoder object

    with open('label\_encoder1.pickle', 'rb') as enc:

        lbl\_encoder = pickle.load(enc)

    # Parameters

    max\_len = 20

    booking\_info = {}  # Store booking information

    # Greet the user

    print(Fore.GREEN + "ChatBot:" + Style.RESET\_ALL, np.random.choice(data['intents'][0]['responses']))

    while True:

        print(Fore.LIGHTBLUE\_EX + "User: " + Style.RESET\_ALL, end="")

        inp = input()

        if inp.lower() == "quit":

            break

        result = model.predict(keras.preprocessing.sequence.pad\_sequences(tokenizer.texts\_to\_sequences([inp]),

                                             truncating='post', maxlen=max\_len))

        tag = lbl\_encoder.inverse\_transform([np.argmax(result)])

        for intent in data['intents']:

            if intent['tag'] == tag:

                if intent['tag'] == 'book\_hotel':

                    print(Fore.GREEN + "ChatBot:" + Style.RESET\_ALL, np.random.choice(intent['responses']))

                    for question in intent['questions']:

                        answer = input(question + " ")

                        booking\_info[question] = answer

                    booking\_id = random.randint(1000, 9999)

                    booking\_info['booking\_id'] = booking\_id

                    print("ChatBot: Your booking has been confirmed! Booking ID: {}".format(booking\_id))

                    break

                elif intent['tag'] == 'cancel\_booking':

                    print(Fore.GREEN + "ChatBot:" + Style.RESET\_ALL, np.random.choice(intent['responses']))

                    booking\_id = input(intent['questions'][0] + " ")

                    print("ChatBot: Your booking with ID {} has been canceled.".format(booking\_id))

                    break

                elif intent['tag'] == 'modify\_booking':

                    print(Fore.GREEN + "ChatBot:" + Style.RESET\_ALL, np.random.choice(intent['responses']))

                    for question in intent['questions']:

                        answer = input(question + " ")

                        booking\_info[question] = answer

                    booking\_id = booking\_info['Enter your booking ID:']

                    print("ChatBot: Your booking with ID {} has been modified.".format(booking\_id))

                    break

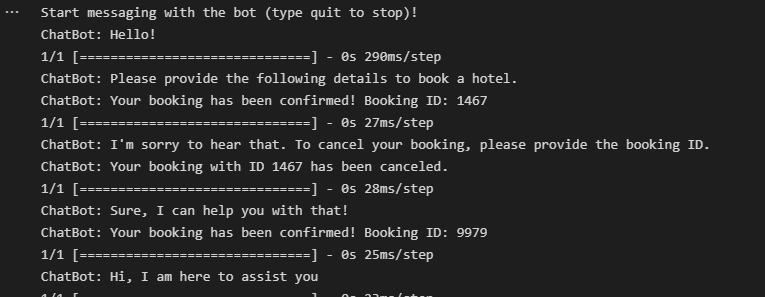
                else:

                    print(Fore.GREEN + "ChatBot:" + Style.RESET\_ALL, np.random.choice(intent['responses']))

                    break

print(Fore.YELLOW + "Start messaging with the bot (type quit to stop)!" + Style.RESET\_ALL)

chat()

****

**Intent.json**

{

    "intents":[

  {

    "tag": "greeting",

    "patterns": [

       "Hi",

       "hiiii",

       "How are you?",

       "Is anyone there?",

       "Hello",

       "Good day",

       "What's up",

       "how are ya",

       "heyy",

       "whatsup",

       "??? ??? ??"

    ],

    "responses": [

      "Hi there, how can I help?",

      "Hello!",

      "hello alix here , how can I help you ",

      "hey, have a nice day",

      "Hi, I am here to assist you"

    ],

    "context\_set": ""

 },

  {

    "tag": "goodbye",

    "patterns": [

       "cya",

       "see you",

       "bye bye",

       "See you later",

       "Goodbye",

       "I am Leaving",

       "Bye",

       "Have a Good day",

       "talk to you later",

       "ttyl",

       "i got to go",

       "gtg"

    ],

    "responses": [

       "Sad to see you go :(",

       "Talk to you later",

       "Goodbye!",

       "Come back soon"

    ],

    "context\_set": ""

 },

  {

    "tag": "creator",

    "patterns": [

       "what is the name of your developers",

       "what is the name of your creators",

       "what is the name of the developers",

       "what is the name of the creators",

       "who created you",

       "your developers",

       "your creators",

       "who are your developers",

       "developers",

       "you are made by",

       "you are made by whom",

       "who created you",

       "who create you",

       "creators",

       "who made you",

       "who designed you"

    ],

    "responses": [

       "AI Student Anushka Jadhav"

    ],

    "context\_set": ""

 },

  {

    "tag": "name",

    "patterns": [

       "name",

       "your name",

       "do you have a name",

       "what are you called",

       "what is your name",

       "what should I call you",

       "whats your name?",

       "what are you",

       "who are you",

       "who is this",

       "what am i chatting to",

       "who am i taking to",

       "what are you"

    ],

    "responses": [

       "You can call me alix.",

       "I'm Mind Reader",

       "I am a Chatbot.",

       "I am your helper"],

    "context\_set": ""

   },

   {

    "tag": "thanks",

    "patterns": ["Thanks",

        "Thank you",

        "That's helpful",

        "Thanks for the help"],

    "responses": [

        "Happy to help!",

        "Any time!",

        "My pleasure",

        "You're most welcome!"],

    "context\_set": ""

    },

    {"tag": "about",

     "patterns": [

        "Who are you?",

        "What are you?",

        "Who you are?" ],

     "responses": [

        "I.m Joana, your bot assistant",

        "I'm Joana, an Artificial Intelligent bot"],

     "context\_set": ""

    },

    {

    "tag": "help",

    "patterns": [

        "Could you help me?",

        "give me a hand please",

        "Can you help?",

        "What can you do for me?",

        "I need a support",

        "I need a help",

        "support me please"],

    "responses": [

        "Tell me how can assist you",

        "Tell me your problem to assist you",

        "Yes Sure, How can I support you"],

    "context\_set": ""

    },

    {

    "tag": "complaint",

    "patterns": [

        "have a complaint",

        "I want to raise a complaint",

        "there is a complaint about a service"],

    "responses": [

        "Please provide us your complaint in order to assist you",

        "Please mention your complaint, we will reach you and sorry for any inconvenience caused"],

    "context\_set": ""

    },

   {

      "tag": "book\_hotel",

      "patterns": [

        "book a hotel",

        "make a hotel reservation",

        "reserve a hotel room",

        "hotel booking"

      ],

      "responses": [

        "Sure, I can help you with that!",

        "Please provide the following details to book a hotel.",

        "Let's get started with your hotel booking!"

      ],

      "questions": [

        "Enter the check-in date (DD-MM-YY):",

        "Enter the check-out date (DD-MM-YY):",

        "Enter the city(Mumbai,Gujrat,Surat,Lonavla):",

        "Enter the number of people:"

      ],

      "context\_set": ""

   },

   {

      "tag": "cancel\_booking",

      "patterns": [

         "cancel booking",

         "cancel my booking",

         "i want to cancel my booking",

         "Cancel my room",

         "cancel hotel reservation"],

      "responses": ["I'm sorry to hear that. To cancel your booking, please provide the booking ID."],

      "questions": [

        "Enter your booking ID:"

      ]

    },

    {

      "tag": "modify\_booking",

      "patterns": [

         "modify booking",

         "modify my booking",

         "change hotel reservation"],

      "responses": ["Sure, to modify your booking, please provide the booking ID and the updated details."],

      "questions": [

        "Enter your booking ID:",

        "Enter the updated check-in date (YYYY-MM-DD):",

        "Enter the updated check-out date (YYYY-MM-DD):",

        "Enter the updated city:",

        "Enter the updated number of people:"

      ]

    }

   ]

}