1581663590_chatbotCourseEndProject

November 3, 2024

1 1. NLP and Speech Recognition Chatbot

Table of contents

## 1.1. Problem Statement

A company holds an event that has been given the deserved promotion through marketing in hopes of attracting as big an audience as possible. Now, it's up to the customer support team to guide the audience and answer any queries. Providing high-quality support and guidance is the challenge. The chatbot is very helpful for its 24/7 presence and ability to reply instantly.

## 1.2. Objectives

Develop a real-time chatbot to engage with the customers in order to boost their business growth by using NLP and Speech Recognition.

Domain: Customer Support

## 1.3. Analysis To Be Done

Create a set of prebuilt commands or inputs as a dataset. Here, we use command .json as Dataset that contains the patterns we need to find and the responses we want to return to the user.

### 1.3.1. Ensure Necessary Modules are Installed

[1]:
```
%pip install python-dotenv
%pip install nltk
%pip install keras
%pip install SpeechRecognition
%pip install tensorflow
# %pip install pickle - standard library in python does not need to be installed
```

```
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.1
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.6)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages
(3.4.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-
packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from keras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras) (13.9.3)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages
(from keras) (3.12.1)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras) (0.13.0)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-
packages (from keras) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from keras) (24.1)
Requirement already satisfied: typing-extensions>=4.5.0 in
/usr/local/lib/python3.10/dist-packages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from rich->keras) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
Collecting SpeechRecognition
  Downloading SpeechRecognition-3.11.0-py2.py3-none-any.whl.metadata (28 kB)
Requirement already satisfied: requests>=2.26.0 in
/usr/local/lib/python3.10/dist-packages (from SpeechRecognition) (2.32.3)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from SpeechRecognition) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.26.0->SpeechRecognition) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.26.0->SpeechRecognition) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.26.0->SpeechRecognition) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.26.0->SpeechRecognition) (2024.8.30)
Downloading SpeechRecognition-3.11.0-py2.py3-none-any.whl (32.8 MB)
                         32.8/32.8 MB
48.4 MB/s eta 0:00:00
Installing collected packages: SpeechRecognition
Successfully installed SpeechRecognition-3.11.0
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-
packages (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (3.12.1)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (24.1)
Requirement already satisfied:
```

```
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->tensorflow) (13.9.3)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->tensorflow) (0.13.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
(3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.0.6)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow)
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow)
(2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow) (0.1.2)
```

### 1.3.2. Import Modules and Set Default Environment Variables and Load Data

[2]:
```python
import json
import pickle
import random
import copy

import numpy as np
import tensorflow as tf

import nltk
from nltk.stem import WordNetLemmatizer

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

[5]:
```python
# Download necessary NLTK data if not already downloaded
nltk.download('punkt')
nltk.download('wordnet')

# Directory for dataset and dependency files
dataset_dir = '/content'

# Initialize lists
words = []
classes = []
documents = []
ignore_words = ['?', '!']
```

```
# Initialize lemmatizer
lemmatizer = WordNetLemmatizer()

# Load intents file
with open(f'{dataset_dir}/commands.json') as data_file:
    intents = json.load(data_file)
```

```
[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…
```

### 1.3.3. Preprocess Data

[6]:
```
# Loop through each intent in the intents dataset
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # Tokenize each word in the sentence
        word_list = nltk.word_tokenize(pattern)
        words.extend(word_list)

        # Add to documents in our corpus
        documents.append((word_list, intent['tag']))

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

# 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)))

# Display basic stats
print(len(documents), "documents")
print(len(classes), "classes", classes)
print(len(words), "unique lemmatized words", words)
```

```
47 documents
9 classes ['adverse_drug', 'blood_pressure', 'blood_pressure_search', 'goodbye',
'greeting', 'hospital_search', 'options', 'pharmacy_search', 'thanks']
88 unique lemmatized words ["'s", ',', 'a', 'adverse', 'all', 'anyone', 'are',
'awesome', 'be', 'behavior', 'blood', 'by', 'bye', 'can', 'causing', 'chatting',
'check', 'could', 'data', 'day', 'detail', 'do', 'dont', 'drug', 'entry',
'find', 'for', 'give', 'good', 'goodbye', 'have', 'hello', 'help', 'helpful',
'helping', 'hey', 'hi', 'history', 'hola', 'hospital', 'how', 'i', 'id', 'is',
'later', 'list', 'load', 'locate', 'log', 'looking', 'lookup', 'management',
'me', 'module', 'nearby', 'next', 'nice', 'of', 'offered', 'open', 'patient',
'pharmacy', 'pressure', 'provide', 'reaction', 'related', 'result', 'search',
'searching', 'see', 'show', 'suitable', 'support', 'task', 'thank', 'thanks',
'that', 'there', 'till', 'time', 'to', 'transfer', 'up', 'want', 'what',
'which', 'with', 'you']
```

1.0.1 1.3.4. Create Pickle Files

```
[7]: pickle.dump(words,open(f'{dataset_dir}/words.pkl','wb'))
     pickle.dump(classes,open(f'{dataset_dir}/classes.pkl','wb'))
```

1.0.2 1.3.5. Create Training And Testing Datasets

```
[8]: # Define output_empty based on the number of classes
     output_empty = [0] * len(classes)

     # Create separate lists for training data inputs (X) and outputs (Y)
     train_x = []
     train_y = []

     # Create bag of words for each sentence and corresponding output
     for doc in documents:
         # Initialize our bag of words
         bag = []
         # List of tokenized words for the pattern
         pattern_words = doc[0]
         # Lemmatize each word to create the base form
         pattern_words = [lemmatizer.lemmatize(word.lower()) for word in␣
      ↪pattern_words]
         # Create the bag of words array with 1 if word match found in current␣
      ↪pattern
         bag = [1 if w in pattern_words else 0 for w in words]

         # Output is '0' for each tag and '1' for current tag
         output_row = list(output_empty)
         output_row[classes.index(doc[1])] = 1

         # Append the bag of words and output row to their respective lists
         train_x.append(bag)
         train_y.append(output_row)

     # Initial deep verification of train_x and train_y
     def verify_array(array, name):
         print(f"Verifying {name}...")
         for i, row in enumerate(array):
```

```
        if not isinstance(row, np.ndarray):
            print(f"Row {i} in {name} is not a numpy array. Found type:␣
↪{type(row)}")
        elif row.shape[0] != array.shape[1]:
            print(f"Row {i} in {name} has inconsistent shape. Expected {array.
↪shape[1]}, found {row.shape[0]}")
        elif row.dtype != np.float32:
            print(f"Row {i} in {name} has incorrect dtype. Expected float32,␣
↪found {row.dtype}")

# Convert train_x and train_y to numpy arrays
train_x = np.array(train_x, dtype=np.float32)
train_y = np.array(train_y, dtype=np.float32)

# Deep verification before proceeding
verify_array(train_x, "train_x")
verify_array(train_y, "train_y")

# Ensure consistency by stacking rows
try:
    train_x = np.vstack(train_x)
    train_y = np.vstack(train_y)
    print("train_x and train_y successfully stacked.")
except ValueError as e:
    print(f"Error in stacking train_x or train_y: {e}")
    raise

# Print shapes and types for debugging
print("Shape of train_x:", train_x.shape)
print("Shape of train_y:", train_y.shape)
print("Data type of train_x:", train_x.dtype)
print("Data type of train_y:", train_y.dtype)

# Double-check by printing sample values if needed
print(f"Sample values from train_x:\n", train_x[:5])
print(f"Sample values from train_y:\n", train_y[:5])
```

```
Verifying train_x…
Verifying train_y…
train_x and train_y successfully stacked.
Shape of train_x: (47, 88)
Shape of train_y: (47, 9)
Data type of train_x: float32
Data type of train_y: float32
Sample values from train_x:
 [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
Sample values from train_y:
 [[0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0.]]
```

1.0.3 1.3.6. Build the Model

[10]:
```python
# Define model structure
model = Sequential()
model.add(Input(shape=(train_x.shape[1],)))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(train_y.shape[1], activation='softmax'))

# Compile model with updated learning rate parameter
sgd = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd,␣
 ↪metrics=['accuracy'])

# Train and save the model
hist = model.fit(train_x, train_y, epochs=200, batch_size=5, verbose=1)
model.save(f'{dataset_dir}/chatbot_model.keras', hist)

print("Model created and saved")
```

```
Epoch 1/200
```

```
10/10             3s 93ms/step -
accuracy: 0.0225 - loss: 2.2943
Epoch 2/200
10/10             1s 3ms/step -
accuracy: 0.0875 - loss: 2.1613
Epoch 3/200
10/10             0s 3ms/step -
accuracy: 0.1442 - loss: 2.1039
Epoch 4/200
10/10             0s 4ms/step -
accuracy: 0.2066 - loss: 2.0551
Epoch 5/200
10/10             0s 3ms/step -
accuracy: 0.2533 - loss: 1.9286
Epoch 6/200
10/10             0s 3ms/step -
accuracy: 0.4563 - loss: 1.7430
Epoch 7/200
10/10             0s 3ms/step -
accuracy: 0.4178 - loss: 1.6951
Epoch 8/200
10/10             0s 3ms/step -
accuracy: 0.5705 - loss: 1.5733
Epoch 9/200
10/10             0s 3ms/step -
accuracy: 0.6280 - loss: 1.4621
Epoch 10/200
10/10             0s 2ms/step -
accuracy: 0.6161 - loss: 1.3698
Epoch 11/200
10/10             0s 2ms/step -
accuracy: 0.5657 - loss: 1.3686
Epoch 12/200
10/10             0s 3ms/step -
accuracy: 0.7085 - loss: 0.9618
Epoch 13/200
10/10             0s 2ms/step -
accuracy: 0.6010 - loss: 1.1272
Epoch 14/200
10/10             0s 3ms/step -
accuracy: 0.8476 - loss: 0.9853
Epoch 15/200
10/10             0s 3ms/step -
accuracy: 0.7121 - loss: 0.8652
Epoch 16/200
10/10             0s 3ms/step -
accuracy: 0.6977 - loss: 0.8643
Epoch 17/200
```

```
10/10              0s 3ms/step -
accuracy: 0.7108 - loss: 0.9278
Epoch 18/200
10/10              0s 4ms/step -
accuracy: 0.8067 - loss: 0.8040
Epoch 19/200
10/10              0s 5ms/step -
accuracy: 0.8396 - loss: 0.6257
Epoch 20/200
10/10              0s 3ms/step -
accuracy: 0.7907 - loss: 0.6845
Epoch 21/200
10/10              0s 3ms/step -
accuracy: 0.7821 - loss: 0.6692
Epoch 22/200
10/10              0s 3ms/step -
accuracy: 0.8745 - loss: 0.4649
Epoch 23/200
10/10              0s 3ms/step -
accuracy: 0.8296 - loss: 0.4774
Epoch 24/200
10/10              0s 3ms/step -
accuracy: 0.7151 - loss: 0.6732
Epoch 25/200
10/10              0s 3ms/step -
accuracy: 0.8421 - loss: 0.4208
Epoch 26/200
10/10              0s 3ms/step -
accuracy: 0.8495 - loss: 0.4069
Epoch 27/200
10/10              0s 3ms/step -
accuracy: 0.9169 - loss: 0.4295
Epoch 28/200
10/10              0s 3ms/step -
accuracy: 0.8641 - loss: 0.4461
Epoch 29/200
10/10              0s 4ms/step -
accuracy: 0.9235 - loss: 0.2879
Epoch 30/200
10/10              0s 3ms/step -
accuracy: 0.9305 - loss: 0.3442
Epoch 31/200
10/10              0s 4ms/step -
accuracy: 0.9629 - loss: 0.2629
Epoch 32/200
10/10              0s 3ms/step -
accuracy: 0.8908 - loss: 0.3792
Epoch 33/200
```

```
10/10            0s 2ms/step -
accuracy: 0.9447 - loss: 0.2449
Epoch 34/200
10/10            0s 2ms/step -
accuracy: 0.9785 - loss: 0.2476
Epoch 35/200
10/10            0s 2ms/step -
accuracy: 0.9780 - loss: 0.1623
Epoch 36/200
10/10            0s 2ms/step -
accuracy: 0.9961 - loss: 0.1734
Epoch 37/200
10/10            0s 2ms/step -
accuracy: 0.9826 - loss: 0.1233
Epoch 38/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0735
Epoch 39/200
10/10            0s 2ms/step -
accuracy: 0.9570 - loss: 0.1715
Epoch 40/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0872
Epoch 41/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.1009
Epoch 42/200
10/10            0s 2ms/step -
accuracy: 0.8621 - loss: 0.2570
Epoch 43/200
10/10            0s 2ms/step -
accuracy: 0.9545 - loss: 0.1371
Epoch 44/200
10/10            0s 2ms/step -
accuracy: 0.9167 - loss: 0.1812
Epoch 45/200
10/10            0s 2ms/step -
accuracy: 0.9826 - loss: 0.1018
Epoch 46/200
10/10            0s 2ms/step -
accuracy: 0.9961 - loss: 0.0940
Epoch 47/200
10/10            0s 2ms/step -
accuracy: 0.8968 - loss: 0.2095
Epoch 48/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0728
Epoch 49/200
```

```
10/10             0s 2ms/step -
accuracy: 0.9258 - loss: 0.1635
Epoch 50/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.1378
Epoch 51/200
10/10             0s 3ms/step -
accuracy: 0.8894 - loss: 0.2453
Epoch 52/200
10/10             0s 2ms/step -
accuracy: 0.9547 - loss: 0.1370
Epoch 53/200
10/10             0s 4ms/step -
accuracy: 0.9267 - loss: 0.2441
Epoch 54/200
10/10             0s 2ms/step -
accuracy: 0.9301 - loss: 0.1185
Epoch 55/200
10/10             0s 2ms/step -
accuracy: 1.0000 - loss: 0.0573
Epoch 56/200
10/10             0s 2ms/step -
accuracy: 0.9258 - loss: 0.1114
Epoch 57/200
10/10             0s 2ms/step -
accuracy: 1.0000 - loss: 0.0753
Epoch 58/200
10/10             0s 2ms/step -
accuracy: 0.9076 - loss: 0.1431
Epoch 59/200
10/10             0s 2ms/step -
accuracy: 1.0000 - loss: 0.0533
Epoch 60/200
10/10             0s 2ms/step -
accuracy: 0.9642 - loss: 0.1072
Epoch 61/200
10/10             0s 2ms/step -
accuracy: 1.0000 - loss: 0.0807
Epoch 62/200
10/10             0s 2ms/step -
accuracy: 0.9447 - loss: 0.0859
Epoch 63/200
10/10             0s 2ms/step -
accuracy: 0.9892 - loss: 0.0567
Epoch 64/200
10/10             0s 2ms/step -
accuracy: 0.9941 - loss: 0.1259
Epoch 65/200
```

```
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0522
Epoch 66/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0364
Epoch 67/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0779
Epoch 68/200
10/10            0s 2ms/step -
accuracy: 0.9918 - loss: 0.0775
Epoch 69/200
10/10            0s 3ms/step -
accuracy: 0.9309 - loss: 0.1160
Epoch 70/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.1171
Epoch 71/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0372
Epoch 72/200
10/10            0s 2ms/step -
accuracy: 0.9941 - loss: 0.0552
Epoch 73/200
10/10            0s 2ms/step -
accuracy: 0.9309 - loss: 0.1997
Epoch 74/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0199
Epoch 75/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0446
Epoch 76/200
10/10            0s 2ms/step -
accuracy: 0.9961 - loss: 0.0745
Epoch 77/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0390
Epoch 78/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0257
Epoch 79/200
10/10            0s 2ms/step -
accuracy: 0.9348 - loss: 0.1090
Epoch 80/200
10/10            0s 2ms/step -
accuracy: 0.9892 - loss: 0.0328
Epoch 81/200
```

```
10/10              0s 2ms/step -
accuracy: 0.9941 - loss: 0.0481
Epoch 82/200
10/10              0s 2ms/step -
accuracy: 0.9439 - loss: 0.0672
Epoch 83/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0672
Epoch 84/200
10/10              0s 2ms/step -
accuracy: 0.9629 - loss: 0.0406
Epoch 85/200
10/10              0s 2ms/step -
accuracy: 0.9720 - loss: 0.0916
Epoch 86/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0362
Epoch 87/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0222
Epoch 88/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0232
Epoch 89/200
10/10              0s 3ms/step -
accuracy: 0.9961 - loss: 0.0887
Epoch 90/200
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0341
Epoch 91/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0548
Epoch 92/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0442
Epoch 93/200
10/10              0s 5ms/step -
accuracy: 0.9629 - loss: 0.0659
Epoch 94/200
10/10              0s 4ms/step -
accuracy: 0.9447 - loss: 0.1216
Epoch 95/200
10/10              0s 4ms/step -
accuracy: 1.0000 - loss: 0.0223
Epoch 96/200
10/10              0s 4ms/step -
accuracy: 0.9826 - loss: 0.0587
Epoch 97/200
```

```
10/10             0s 4ms/step -
accuracy: 0.9629 - loss: 0.0988
Epoch 98/200
10/10             0s 5ms/step -
accuracy: 0.9826 - loss: 0.0557
Epoch 99/200
10/10             0s 5ms/step -
accuracy: 1.0000 - loss: 0.0368
Epoch 100/200
10/10             0s 4ms/step -
accuracy: 0.9629 - loss: 0.1003
Epoch 101/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0199
Epoch 102/200
10/10             0s 3ms/step -
accuracy: 0.9918 - loss: 0.0339
Epoch 103/200
10/10             0s 9ms/step -
accuracy: 1.0000 - loss: 0.0169
Epoch 104/200
10/10             0s 4ms/step -
accuracy: 0.9720 - loss: 0.0490
Epoch 105/200
10/10             0s 4ms/step -
accuracy: 1.0000 - loss: 0.0277
Epoch 106/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0182
Epoch 107/200
10/10             0s 8ms/step -
accuracy: 1.0000 - loss: 0.0322
Epoch 108/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0166
Epoch 109/200
10/10             0s 4ms/step -
accuracy: 0.9720 - loss: 0.0871
Epoch 110/200
10/10             0s 8ms/step -
accuracy: 0.9826 - loss: 0.0301
Epoch 111/200
10/10             0s 3ms/step -
accuracy: 0.9447 - loss: 0.1576
Epoch 112/200
10/10             0s 3ms/step -
accuracy: 0.9629 - loss: 0.0918
Epoch 113/200
```

```
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0725
Epoch 114/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0362
Epoch 115/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0481
Epoch 116/200
10/10            0s 2ms/step -
accuracy: 0.9661 - loss: 0.0563
Epoch 117/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0358
Epoch 118/200
10/10            0s 2ms/step -
accuracy: 0.9521 - loss: 0.1082
Epoch 119/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0203
Epoch 120/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0102
Epoch 121/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0181
Epoch 122/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0485
Epoch 123/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0090
Epoch 124/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0058
Epoch 125/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0419
Epoch 126/200
10/10            0s 2ms/step -
accuracy: 0.9780 - loss: 0.0957
Epoch 127/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0391
Epoch 128/200
10/10            0s 2ms/step -
accuracy: 0.9661 - loss: 0.0784
Epoch 129/200
```

```
10/10            0s 2ms/step -
accuracy: 0.9439 - loss: 0.0743
Epoch 130/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0112
Epoch 131/200
10/10            0s 2ms/step -
accuracy: 0.9388 - loss: 0.1437
Epoch 132/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0236
Epoch 133/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0451
Epoch 134/200
10/10            0s 2ms/step -
accuracy: 0.9918 - loss: 0.0279
Epoch 135/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0105
Epoch 136/200
10/10            0s 2ms/step -
accuracy: 0.9892 - loss: 0.0339
Epoch 137/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0274
Epoch 138/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0061
Epoch 139/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0141
Epoch 140/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0131
Epoch 141/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0502
Epoch 142/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0158
Epoch 143/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0422
Epoch 144/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0130
Epoch 145/200
```

```
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0153
Epoch 146/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0093
Epoch 147/200
10/10              0s 2ms/step -
accuracy: 0.9918 - loss: 0.0213
Epoch 148/200
10/10              0s 2ms/step -
accuracy: 0.9590 - loss: 0.0424
Epoch 149/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0188
Epoch 150/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0098
Epoch 151/200
10/10              0s 2ms/step -
accuracy: 0.9862 - loss: 0.0463
Epoch 152/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0297
Epoch 153/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0220
Epoch 154/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0244
Epoch 155/200
10/10              0s 3ms/step -
accuracy: 0.9629 - loss: 0.0525
Epoch 156/200
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0164
Epoch 157/200
10/10              0s 3ms/step -
accuracy: 0.9167 - loss: 0.1620
Epoch 158/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0197
Epoch 159/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0182
Epoch 160/200
10/10              0s 2ms/step -
accuracy: 1.0000 - loss: 0.0056
Epoch 161/200
```

```
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0095
Epoch 162/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0080
Epoch 163/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0071
Epoch 164/200
10/10            0s 3ms/step -
accuracy: 1.0000 - loss: 0.0134
Epoch 165/200
10/10            0s 2ms/step -
accuracy: 0.9941 - loss: 0.0154
Epoch 166/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0122
Epoch 167/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0381
Epoch 168/200
10/10            0s 2ms/step -
accuracy: 0.9720 - loss: 0.0717
Epoch 169/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0094
Epoch 170/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0667
Epoch 171/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0116
Epoch 172/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0110
Epoch 173/200
10/10            0s 2ms/step -
accuracy: 0.9780 - loss: 0.0322
Epoch 174/200
10/10            0s 2ms/step -
accuracy: 1.0000 - loss: 0.0072
Epoch 175/200
10/10            0s 4ms/step -
accuracy: 1.0000 - loss: 0.0093
Epoch 176/200
10/10            0s 6ms/step -
accuracy: 1.0000 - loss: 0.0196
Epoch 177/200
```

```
10/10             0s 4ms/step -
accuracy: 1.0000 - loss: 0.0312
Epoch 178/200
10/10             0s 4ms/step -
accuracy: 1.0000 - loss: 0.0328
Epoch 179/200
10/10             0s 4ms/step -
accuracy: 1.0000 - loss: 0.0062
Epoch 180/200
10/10             0s 4ms/step -
accuracy: 1.0000 - loss: 0.0154
Epoch 181/200
10/10             0s 4ms/step -
accuracy: 1.0000 - loss: 0.0153
Epoch 182/200
10/10             0s 4ms/step -
accuracy: 0.9582 - loss: 0.1122
Epoch 183/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0198
Epoch 184/200
10/10             0s 2ms/step -
accuracy: 1.0000 - loss: 0.0164
Epoch 185/200
10/10             0s 2ms/step -
accuracy: 0.9941 - loss: 0.0298
Epoch 186/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0365
Epoch 187/200
10/10             0s 3ms/step -
accuracy: 0.9447 - loss: 0.1334
Epoch 188/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0330
Epoch 189/200
10/10             0s 4ms/step -
accuracy: 1.0000 - loss: 0.0158
Epoch 190/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0097
Epoch 191/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0262
Epoch 192/200
10/10             0s 3ms/step -
accuracy: 1.0000 - loss: 0.0112
Epoch 193/200
```

```
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0602
Epoch 194/200
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0036
Epoch 195/200
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0098
Epoch 196/200
10/10              0s 2ms/step -
accuracy: 0.9767 - loss: 0.0565
Epoch 197/200
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0253
Epoch 198/200
10/10              0s 3ms/step -
accuracy: 1.0000 - loss: 0.0276
Epoch 199/200
10/10              0s 5ms/step -
accuracy: 1.0000 - loss: 0.0174
Epoch 200/200
10/10              0s 4ms/step -
accuracy: 0.9892 - loss: 0.0435
Model created and saved
```

1.0.4 1.3.7. Predict The Responses

1.3.7.1. Load Required Python Modules

[11]:
```python
import nltk
from nltk.stem import WordNetLemmatizer
import pickle
import numpy as np

from tensorflow.keras.models import load_model
```

1.3.7.2. Establish Environment Variables and Load Data

[12]:
```python
# Define dataset directory
dataset_dir = '/content'

# Initialize lemmatizer
lemmatizer = WordNetLemmatizer()

# Load the trained model
model = load_model(f'{dataset_dir}/chatbot_model.keras')

# Load words and classes
with open(f'{dataset_dir}/words.pkl', 'rb') as f:
    words = pickle.load(f)
with open(f'{dataset_dir}/classes.pkl', 'rb') as f:
    classes = pickle.load(f)

# Load intents
with open(f'{dataset_dir}/commands.json', 'r') as f:
    intents = json.load(f)
```

1.3.7.3. Creat Prediction Functions

[15]:
```
# Function to clean up the sentence
def clean_up_sentence(sentence):
    # Tokenize the sentence
    sentence_words = nltk.word_tokenize(sentence)
    # Lemmatize each word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in␣
 ↪sentence_words]
    return sentence_words

# Function to create a bag of words from the sentence
def bow(sentence, words, show_details=True):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
                bag[i] = 1
                if show_details:
                    print(f'found in bag: {w}')
    return np.array(bag)

# Function to predict the class
def predict_class(sentence, model):
    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]
    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

# Function to get the response
def get_response(intents_list, intents_json):
    """
    Retrieves the chatbot's response based on predicted intent.
    Args:
        intents_list: A list of predicted intents.
        intents_json: The JSON data containing intents and responses.

    Returns:
        The chatbot's response string.
    """
    if intents_list:
        tag = intents_list[0]['intent']
        # Access the 'intents' key from the intents_json dictionary
        # Previously: list_of_intents = intents_json['intents'] # intents_json␣
 ↪is actually a list and not a dictionary
        list_of_intents = intents_json.get('intents', []) # Get the 'intents'␣
 ↪key from the dictionary 'intents' declared on line 12 of␣
 ↪ipython-input-12-858ce5ee1d16
                                                          # or assign an empty list␣
 ↪in case of absence of an 'intents' key.
        for i in list_of_intents:
            if i['tag'] == tag:
                result = random.choice(i['responses'])
                break
    else:
        result = "I don't understand, please try again."
    return result

# Function to chat with the model
def chatbot_response(text):
    """
    Gets the chatbot's response to the user's input.

    Args:
        text: The user's input text.

    Returns:
        The chatbot's response string.
    """
    intents_list = predict_class(text, model)
    # Pass the intents data (intents) loaded from JSON as the second argument
    # Previously: response = get_response(intents, intents)
    response = get_response(intents_list, intents) # Pass 'intents' variable␣
 ↪from line 12 of ipython-input-12-858ce5ee1d16 to get_response() method.
    return response
```

1.3.7.4. Interactive loop for testing

[16]:
```
print("Chatbot is ready to chat! (Type 'exit' to stop)")
while True:
    message = input("You: ")
    if message.lower() == "exit":
        print("Goodbye!")
        break
    response = chatbot_response(message)
    print("Bot:", response)
```

```
Chatbot is ready to chat! (Type 'exit' to stop)
You: Hello
1/1              0s 25ms/step
Bot: Hi there, how can I help?
You: Hey, is anyone there?
1/1              0s 16ms/step
Bot: Hi there, how can I help?
You: Hola
1/1              0s 16ms/step
Bot: Hello, thanks for asking
You: Goodbye
1/1              0s 16ms/step
Bot: See you!
You: Nice chatting with you, bye!
1/1              0s 17ms/step
Bot: Bye! Come back again soon.
You: Till next time
1/1              0s 19ms/step
Bot: See you!
You: Thank you for your help
1/1              0s 17ms/step
Bot: Happy to help!
You: That's helpful
1/1              0s 16ms/step
Bot: Any time!
You: Awesome, thanks
1/1              0s 24ms/step
Bot: My pleasure
You: How can you help me?
1/1              0s 16ms/step
Bot: Offering support for Adverse drug reaction, Blood pressure, Hospitals and
Pharmacies
You: What support do you offer?
1/1              0s 26ms/step
Bot: Offering support for Adverse drug reaction, Blood pressure, Hospitals and
Pharmacies
You: What can you do?
1/1              0s 17ms/step
Bot: Offering support for Adverse drug reaction, Blood pressure, Hospitals and
Pharmacies
You: How do I check for adverse drug reactions?
```

```
1/1             0s 25ms/step
Bot: Navigating to Adverse drug reaction module
You: Can you open the adverse drugs module?
1/1             0s 16ms/step
Bot: Navigating to Adverse drug reaction module
You: Give me a list of drugs causing adverse reactions
1/1             0s 25ms/step
Bot: Navigating to Adverse drug reaction module
You: Which drugs are suitable for a patient with adverse reactions?
1/1             0s 16ms/step
Bot: Navigating to Adverse drug reaction module
You: Can you manage blood pressure data?
1/1             0s 26ms/step
Bot: Navigating to Blood Pressure module
You: Show me the blood pressure results for a patient
1/1             0s 17ms/step
Bot: Patient ID?
You: abc123
1/1             0s 16ms/step
Bot: Good to see you again
You: I want to log blood pressure results
1/1             0s 16ms/step
Bot: Navigating to Blood Pressure module
You: Find a pharmacy for me
1/1             0s 17ms/step
Bot: Please provide pharmacy name
You: CVS
1/1             0s 27ms/step
Bot: Good to see you again
You: Locate nearby pharmacies
1/1             0s 17ms/step
Bot: Please provide pharmacy name
You: CVS
1/1             0s 27ms/step
Bot: Good to see you again
You: List pharmacies close by
1/1             0s 16ms/step
Bot: Please provide pharmacy name
You: CVS
1/1             0s 36ms/step
Bot: Hello, thanks for asking
You: Find me a hospital
1/1             0s 17ms/step
Bot: Please provide hospital name or location
You: Community Health
1/1             0s 16ms/step
Bot: Hi there, how can I help?
You: Look up hospital information for patient transfer
```

```
1/1              0s 16ms/step
Bot: Please provide hospital name or location
You: Community Health
1/1              0s 16ms/step
Bot: Hi there, how can I help?
You: Can you book a table for me?
1/1              0s 16ms/step
Bot: I can guide you through Adverse drug reaction list, Blood pressure
tracking, Hospitals and Pharmacies
You: I need help with my car insurance
1/1              0s 25ms/step
Bot: Offering support for Adverse drug reaction, Blood pressure, Hospitals and
Pharmacies
You: What's the weather like?
1/1              0s 25ms/step
Bot: Offering support for Adverse drug reaction, Blood pressure, Hospitals and
Pharmacies
You: Tell me a joke
1/1              0s 19ms/step
Bot: Bye! Come back again soon.
You: Play some music
1/1              0s 17ms/step
Bot: Hi there, how can I help?
You: exit
Goodbye!
```

[18]:
```
!python --version
print(f"TensorFlow version: {tf.__version__}")
```

```
Python 3.10.12
TensorFlow version: 2.17.0
```

[ ]:
```
```