Importing libraries and downloading packages

In [1]:

```
import nltk
import numpy as np
```

In [2]:

```
# downloading model to tokenize message
nltk.download('punkt')
# downloading stopwords
nltk.download('stopwords')
# downloading wordnet, which contains all lemmas of english language
nltk.download('wordnet')
```

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

Out[2]:

```
True
```

In [3]:

```
from nltk.tokenize import word_tokenize

from nltk.corpus import stopwords
stop_words = stopwords.words('english')

from nltk.stem import WordNetLemmatizer
```

In [4]:

```
stop_words
```

Out[4]:

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
```

Function to clean text

In [5]:

```
def clean_corpus(corpus):
  # lowering every word in text
  corpus = [ doc.lower() for doc in corpus]
  cleaned_corpus = []

  stop_words = stopwords.words('english')
  wordnet_lemmatizer = WordNetLemmatizer()

  # iterating over every text[a,b,c]='a b c'
  for doc in corpus:
    # tokenizing text
    tokens = word_tokenize(doc)
    cleaned_sentence = []
    for token in tokens:
      # removing stopwords, and punctuation
      if token not in stop_words and token.isalpha():
        # applying lemmatization
        cleaned_sentence.append(wordnet_lemmatizer.lemmatize(token))
    cleaned_corpus.append(' '.join(cleaned_sentence))
  return cleaned_corpus
```

Loading and cleaning intents

In [6]:

```
!wget -O intents.jsonn https://techlearn-cdn.s3.amazonaws.com/bs_swiggy_chatbot/intent.
```

```
--2022-04-10 12:34:55--  https://techlearn-cdn.s3.amazonaws.com/bs_swiggy_ch
atbot/intent.json (https://techlearn-cdn.s3.amazonaws.com/bs_swiggy_chatbot/
intent.json)
Resolving techlearn-cdn.s3.amazonaws.com (techlearn-cdn.s3.amazonaws.com)...
52.219.66.80
Connecting to techlearn-cdn.s3.amazonaws.com (techlearn-cdn.s3.amazonaws.co
m)|52.219.66.80|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4699 (4.6K) [application/json]
Saving to: 'intents.jsonn'

     0K ....                                                  100% 2.95M=0.0
02s

2022-04-10 12:34:56 (2.95 MB/s) - 'intents.jsonn' saved [4699/4699]
```

In [7]:

```
import json
with open('intents.jsonn', 'r') as file:
  intents = json.load(file)
```

In [8]:

```
corpus = []
tags = []

for intent in intents['intents']:
    # taking all patterns in intents to train a neural network
    for pattern in intent['patterns']:
        corpus.append(pattern)
        tags.append(intent['tag'])
```

In [ ]:

```

```

In [9]:

```
cleaned_corpus = clean_corpus(corpus)
cleaned_corpus
```

Out[9]:

```
['hi',
 'anyone',
 'hey',
 'hola',
 'hello',
 'good day',
 'bye',
 'see later',
 'goodbye',
 'nice chatting bye',
 'till next time',
 '',
 'thanks',
 'thank',
 'helpful',
 'awesome thanks',
 'thanks helping',
 'could help',
 'help provide',
 'helpful',
 'support offered',
 'please check order status',
 'able check order status',
 'help order status',
 'order status',
 'order',
 'food',
 'track order',
 'track food',
 'hi want cancel order',
 'want cancel order',
 'please cancel order',
 'cancel order',
 'want add delivery instruction',
 'please add delivery instruction',
 'include delivery instruction',
 'tell joke',
 'feeling bored',
 'joke please',
 'make laugh',
 'want laugh']
```

Vectorizing intents

In [10]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(cleaned_corpus)
```

In [11]:

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()
y = encoder.fit_transform(np.array(tags).reshape(-1,1))
```

Training neural network

In [12]:

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in d:\users\chint\anaconda3\lib
\site-packages (2.8.0)
Requirement already satisfied: termcolor>=1.1.0 in d:\users\chint\anaconda
3\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in d:\users\chint\anac
onda3\lib\site-packages (from tensorflow) (2.8.0)
Requirement already satisfied: libclang>=9.0.1 in d:\users\chint\anaconda3
\lib\site-packages (from tensorflow) (13.0.0)
Requirement already satisfied: astunparse>=1.6.0 in d:\users\chint\anacond
a3\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: opt-einsum>=2.3.2 in d:\users\chint\anacond
a3\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in d:\users\chint
\anaconda3\lib\site-packages (from tensorflow) (3.7.4.3)
Requirement already satisfied: wrapt>=1.11.0 in d:\users\chint\anaconda3\l
ib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: six>=1.12.0 in d:\users\chint\anaconda3\lib
\site-packages (from tensorflow) (1.15.0)
Requirement already satisfied: numpy>=1.20 in d:\users\chint\anaconda3\lib
```

In [13]:

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential([
                    Dense(128, input_shape=(X.shape[1],), activation='relu'),
                    Dropout(0.2),
                    Dense(64, activation='relu'),
                    Dropout(0.2),
                    Dense(y.shape[1], activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               5888

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 8)                 520

=================================================================
Total params: 14,664
Trainable params: 14,664
Non-trainable params: 0
_________________________________________________________________
```

In [14]:

```
history = model.fit(X.toarray(), y.toarray(), epochs=20, batch_size=1)
```

```
Epoch 1/20
41/41 [==============================] - 1s 3ms/step - loss: 2.0833 - accura
cy: 0.1463
Epoch 2/20
41/41 [==============================] - 0s 3ms/step - loss: 2.0065 - accura
cy: 0.2439
Epoch 3/20
41/41 [==============================] - 0s 3ms/step - loss: 1.9123 - accura
cy: 0.3902
Epoch 4/20
41/41 [==============================] - 0s 3ms/step - loss: 1.7492 - accura
cy: 0.5122
Epoch 5/20
41/41 [==============================] - 0s 3ms/step - loss: 1.6344 - accura
cy: 0.6341
Epoch 6/20
41/41 [==============================] - 0s 3ms/step - loss: 1.4387 - accura
cy: 0.6341
Epoch 7/20
41/41 [==============================] - 0s 3ms/step - loss: 1.1762 - accura
cy: 0.8293
Epoch 8/20
41/41 [==============================] - 0s 3ms/step - loss: 1.0033 - accura
cy: 0.9024
Epoch 9/20
41/41 [==============================] - 0s 3ms/step - loss: 0.7900 - accura
cy: 0.9268
Epoch 10/20
41/41 [==============================] - 0s 3ms/step - loss: 0.5984 - accura
cy: 0.9268
Epoch 11/20
41/41 [==============================] - 0s 3ms/step - loss: 0.4663 - accura
cy: 0.9756
Epoch 12/20
41/41 [==============================] - 0s 3ms/step - loss: 0.3487 - accura
cy: 0.9268
Epoch 13/20
41/41 [==============================] - 0s 3ms/step - loss: 0.3056 - accura
cy: 0.9512
Epoch 14/20
41/41 [==============================] - 0s 3ms/step - loss: 0.1749 - accura
cy: 1.0000
Epoch 15/20
41/41 [==============================] - 0s 3ms/step - loss: 0.1639 - accura
cy: 0.9512
Epoch 16/20
41/41 [==============================] - 0s 2ms/step - loss: 0.1442 - accura
cy: 0.9756
Epoch 17/20
41/41 [==============================] - 0s 3ms/step - loss: 0.1266 - accura
cy: 1.0000
Epoch 18/20
41/41 [==============================] - 0s 3ms/step - loss: 0.1120 - accura
cy: 0.9756
Epoch 19/20
41/41 [==============================] - 0s 3ms/step - loss: 0.1359 - accura
```

```
cy: 0.9756
Epoch 20/20
41/41 [==============================] - 0s 3ms/step - loss: 0.1221 - accura
cy: 0.9512
```

Classifying messages to intent

1. If the intent probability does not match with any intent, then send it to no answer.
2. Get Intent
3. Perform Action

In [15]:

```
# if prediction for every tag is low, then we want to classify that message as no answe

INTENT_NOT_FOUND_THRESHOLD = 0.40

def predict_intent_tag(message):
  message = clean_corpus([message])
  X_test = vectorizer.transform(message)
  #print(message)
  #print(X_test.toarray())
  y = model.predict(X_test.toarray())
  #print (y)
  # if probability of all intent is low, classify it as noanswer
  if y.max() < INTENT_NOT_FOUND_THRESHOLD:
    return 'noanswer'

  prediction = np.zeros_like(y[0])
  prediction[y.argmax()] = 1
  tag = encoder.inverse_transform([prediction])[0][0]
  return tag

print(predict_intent_tag('How you could help me?'))
print(predict_intent_tag('swiggy chat bot'))
print(predict_intent_tag('Where\'s my order'))

```

```
options
goodbye
order-status-request
```

In [16]:

```
import random
import time
```

In [17]:

```
def get_intent(tag):
  # to return complete intent from intent tag
  for intent in intents['intents']:
    if intent['tag'] == tag:
      return intent
```

In [18]:

```
def perform_action(action_code, intent):
  # funition to perform an action which is required by intent

  if action_code == 'CHECK_ORDER_STATUS':
    print('\n Checking database \n')
    time.sleep(2)
    order_status = ['in kitchen', 'with delivery executive']
    delivery_time = []
    return {'intent-tag':intent['next-intent-tag'][0],
            'order_status': random.choice(order_status),
            'delivery_time': random.randint(10, 30)}

  elif action_code == 'ORDER_CANCEL_CONFIRMATION':
    ch = input('BOT: Do you want to continue (Y/n) ?')
    if ch == 'y' or ch == 'Y':
      choice = 0
    else:
      choice = 1
    return {'intent-tag':intent['next-intent-tag'][choice]}

  elif action_code == 'ADD_DELIVERY_INSTRUCTIONS':
    instructions = input('Your Instructions: ')
    return {'intent-tag':intent['next-intent-tag'][0]}
```

Complete chat bot

In [19]:

```
while True:
  # get message from user
  message = input('You: ')
  # predict intent tag using trained neural network
  tag = predict_intent_tag(message)
  # get complete intent from intent tag
  intent = get_intent(tag)
  # generate random response from intent
  response = random.choice(intent['responses'])
  print('Bot: ', response)

  # check if there's a need to perform some action
  if 'action' in intent.keys():
    action_code = intent['action']
    # perform action
    data = perform_action(action_code, intent)
    # get follow up intent after performing action
    followup_intent = get_intent(data['intent-tag'])
    # generate random response from follow up intent
    response = random.choice(followup_intent['responses'])

    # print randomly selected response
    if len(data.keys()) > 1:
      print('Bot: ', response.format(**data))
    else:
      print('Bot: ', response)

  # break loop if intent was goodbye
  if tag == 'goodbye':
    break
```

```
You: where is my order
Bot:  I am checking your status of your order, Please wait.

 Checking database

Bot:  Your order is currently with delivery executive, it will be delivered
in 11 minutes. Do you need any more help?
You: I want to cancel my order
Bot:  I can cancel your order, If your order is in kitchen, there will be a
cancellation fee.  Do you want me to proceed?
BOT: Do you want to continue (Y/n) ?y
Bot:  Your order is canceled, you will receive refund for this order in 2 da
ys. Is there anything else, we can help you with?
You: I want to order another item food
Bot:  I am checking up with for your order

 Checking database

Bot:  Your order will be delivered in 27, currently it is in kitchen. Is the
re anything else, we can help you with?
You: In want to order chicken
Bot:  I am checking your status of your order, Please wait.

 Checking database

Bot:  Your order is currently in kitchen, it will be delivered in 29 minute
s. Do you need any more help?
```

```
You: want is the price of chicken
Bot:  What has T in the beginning, T in the middle, and T at the end?
- A teapot.
You: What is the price of chicken
Bot:  See you!
```

In [ ]:

```
1 gui
```

In [ ]:

```
1
```

Type *Markdown* and LaTeX: α^2

In [ ]:

```
1
```

In [ ]:

```
1
```