In [1]:

```
import json
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, GlobalAveragePooling1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
```

In [2]:

```
with open('intent.json') as intents:
    data = json.load(intents)

training_sentences = []
training_labels = []
label = []
responses = []

for i in data['intents']:
    for j in i['patterns']:
        training_sentences.append(j)
        training_labels.append(i['tag'])
    responses.append(i['responses'])

    if i['tag'] not in label:
      label.append(i['tag'])

num_classes = len(label)
```

In [3]:

```
lbl_encoder = LabelEncoder()
lbl_encoder.fit(training_labels)
training_labels = lbl_encoder.transform(training_labels)
```

In [4]:

```
tokenizer = Tokenizer(num_words=1000,oov_token="<OOV>")
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
seq = tokenizer.texts_to_sequences(training_sentences)
pad_seq = pad_sequences(seq,truncating='post',maxlen=20)
```

In [5]:

```
model = Sequential()
model.add(Embedding(1000, 16, input_length=20))
model.add(GlobalAveragePooling1D())
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 20, 16)            16000

 global_average_pooling1d (G  (None, 16)               0
 lobalAveragePooling1D)

 dense (Dense)               (None, 16)                272

 dense_1 (Dense)             (None, 16)                272

 dense_2 (Dense)             (None, 38)                646

=================================================================
Total params: 17,190
Trainable params: 17,190
Non-trainable params: 0
_________________________________________________________________
```

In [6]:

```
epochs = 500
history = model.fit(pad_seq,np.array(training_labels),epochs=epochs)
```

```
Epoch 1/500
13/13 [==============================] - 2s 3ms/step - loss: 3.6373 - ac
curacy: 0.0321
Epoch 2/500
13/13 [==============================] - 0s 4ms/step - loss: 3.6325 - ac
curacy: 0.0716
Epoch 3/500
13/13 [==============================] - 0s 3ms/step - loss: 3.6278 - ac
curacy: 0.0642
Epoch 4/500
13/13 [==============================] - 0s 3ms/step - loss: 3.6230 - ac
curacy: 0.0642
Epoch 5/500
13/13 [==============================] - 0s 3ms/step - loss: 3.6171 - ac
curacy: 0.0642
Epoch 6/500
13/13 [==============================] - 0s 3ms/step - loss: 3.6099 - ac
curacy: 0.0642
Epoch 7/500
```

In [7]:

```
model.save('chat_model')
```

```
WARNING:absl:Found untraced functions such as _update_step_xla while savin
g (showing 1 of 1). These functions will not be directly callable after lo
ading.
```

In [8]:

```
import pickle

# to save the fitted tokenizer
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)

# to save the fitted label encoder
with open('label_encoder.pickle', 'wb') as ecn_file:
    pickle.dump(lbl_encoder, ecn_file, protocol=pickle.HIGHEST_PROTOCOL)
```

In [10]:

```
with open("intent.json") as file:
    data = json.load(file)
model = keras.models.load_model('chat_model')
# load tokenizer object
with open('tokenizer.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

# load label encoder object
with open('label_encoder.pickle', 'rb') as enc:
    lbl_encoder = pickle.load(enc)

print(lbl_encoder)
```

```
LabelEncoder()
```

In [12]:

```
while True:
  print("User : ",end="")
  inp = input()
  if inp.lower() == 'quit':
    break
  result = model.predict(keras.preprocessing.sequence.pad_sequences(tokenizer.texts_to_se
  #print(result)
  tag = lbl_encoder.inverse_transform([np.argmax(result)])
  for i in data['intents']:
    if tag == i['tag']:
      print("ChatBot : ",np.random.choice(i['responses']))
```

```
User : Hi
1/1 [==============================] - 0s 22ms/step
ChatBot :  Hi there, how can I help?
User : Who are you
1/1 [==============================] - 0s 33ms/step
ChatBot :  I'm Mind Reader
User : Okay tell me what will be the college time
1/1 [==============================] - 0s 24ms/step
ChatBot :  Academic calender is given to you by your class-soordinators af
ter you join your respective classes
User : okay
1/1 [==============================] - 0s 35ms/step
ChatBot :  welcome, anything else i can assist you with?
User : what is the college telephone number
1/1 [==============================] - 0s 25ms/step
ChatBot :  You can contact at: NUMBER
User : Okay great
1/1 [==============================] - 0s 25ms/step
ChatBot :  I am glad I helped you
User : thanks
1/1 [==============================] - 0s 21ms/step
ChatBot :  I am glad I helped you
User : bye
1/1 [==============================] - 0s 21ms/step
ChatBot :  Sad to see you go :(
User : quit
```