Importing Libraries

pickle: This library is used for serializing and deserializing Python objects. Here, it's used to save and load Python objects like lists and dictionaries.

nltk: Natural Language Toolkit (NLTK) is a library used for natural language processing (NLP) tasks like tokenization, lemmatization, stemming, etc.

numpy: This library is used for numerical computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions.

keras: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It's used here for building and training the neural network model.

In [4]:

```
import pickle
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\anike\anaconda3\lib\site-packages (3.7)
Requirement already satisfied: click in c:\users\anike\anaconda3\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: regex>=2021.8.3 in c:\users\anike\anaconda3\lib\site-packages (from nlt
k) (2022.7.9)
Requirement already satisfied: joblib in c:\users\anike\anaconda3\lib\site-packages (from nltk) (1.1.0)
Requirement already satisfied: tqdm in c:\users\anike\anaconda3\lib\site-packages (from nltk) (4.64.1)
Requirement already satisfied: colorama in c:\users\anike\anaconda3\lib\site-packages (from click->nlt
k) (0.4.5)
```

In [5]:

```
!pip install tensorflow keras nltk
```

```
Requirement already satisfied: tensorflow in c:\users\anike\anaconda3\lib\site-packages (2.16.1)
Requirement already satisfied: keras in c:\users\anike\anaconda3\lib\site-packages (3.2.1)
Requirement already satisfied: nltk in c:\users\anike\anaconda3\lib\site-packages (3.7)
Requirement already satisfied: tensorflow-intel==2.16.1 in c:\users\anike\anaconda3\lib\site-packages
(from tensorflow) (2.16.1)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\anike\anaconda3\lib\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (0.2.0)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\anike\anaconda3\lib\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (24.3.25)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\anike\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.16.1->tensorflow) (0.31.0)
Requirement already satisfied: setuptools in c:\users\anike\anaconda3\lib\site-packages (from tensorflo
w-intel==2.16.1->tensorflow) (63.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\anike\anaconda3\lib\site-packages (from te
nsorflow-intel==2.16.1->tensorflow) (3.3.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\anike\anaconda3\lib\site-packages (from ten
sorflow-intel==2.16.1->tensorflow) (2.4.0)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\anike\anaconda3\lib\site
-packages (from tensorflow-intel==2.16.1->tensorflow) (0.5.4)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\anike\anaconda3\lib\site-packages (from tensor
flow-intel==2.16.1->tensorflow) (1.14.1)
Requirement already satisfied: six>=1.12.0 in c:\users\anike\anaconda3\lib\site-packages (from tensorfl
ow-intel==2.16.1->tensorflow) (1.16.0)
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 c:\users\anike\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (4.2
5.3)
Requirement already satisfied: packaging in c:\users\anike\anaconda3\lib\site-packages (from tensorflow
-intel==2.16.1->tensorflow) (21.3)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\anike\anaconda3\lib\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (1.62.1)
Requirement already satisfied: h5py>=3.10.0 in c:\users\anike\anaconda3\lib\site-packages (from tensorf
low-intel==2.16.1->tensorflow) (3.11.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\anike\anaconda3\lib\site-packages (from te
nsorflow-intel==2.16.1->tensorflow) (1.6.3)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\anike\anaconda3\lib\site-packages (from tenso
rflow-intel==2.16.1->tensorflow) (2.1.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\anike\anaconda3\lib\site-packages (from ten
sorflow-intel==2.16.1->tensorflow) (18.1.1)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\anike\anaconda3\lib\site-packages
(from tensorflow-intel==2.16.1->tensorflow) (4.3.0)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\anike\anaconda3\lib\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (2.28.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\anike\anaconda3\lib\site-packages (from
tensorflow-intel==2.16.1->tensorflow) (1.24.3)
Requirement already satisfied: ml-dtypes~=0.3.1 in c:\users\anike\anaconda3\lib\site-packages (from ten
sorflow-intel==2.16.1->tensorflow) (0.3.2)
Requirement already satisfied: tensorboard<2.17,>=2.16 in c:\users\anike\anaconda3\lib\site-packages (f
rom tensorflow-intel==2.16.1->tensorflow) (2.16.2)
Requirement already satisfied: rich in c:\users\anike\anaconda3\lib\site-packages (from keras) (13.7.1)
Requirement already satisfied: optree in c:\users\anike\anaconda3\lib\site-packages (from keras) (0.11.
0)
Requirement already satisfied: namex in c:\users\anike\anaconda3\lib\site-packages (from keras) (0.0.8)
Requirement already satisfied: joblib in c:\users\anike\anaconda3\lib\site-packages (from nltk) (1.1.0)
Requirement already satisfied: tqdm in c:\users\anike\anaconda3\lib\site-packages (from nltk) (4.64.1)
Requirement already satisfied: regex>=2021.8.3 in c:\users\anike\anaconda3\lib\site-packages (from nlt
k) (2022.7.9)
Requirement already satisfied: click in c:\users\anike\anaconda3\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: colorama in c:\users\anike\anaconda3\lib\site-packages (from click->nlt
k) (0.4.5)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\anike\anaconda3\lib\site-packages (fro
m rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\anike\anaconda3\lib\site-packages (f
rom rich->keras) (2.17.2)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\anike\anaconda3\lib\site-packages (from a
stunparse>=1.6.0->tensorflow-intel==2.16.1->tensorflow) (0.37.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\anike\anaconda3\lib\site-packages (from markdown-
it-py>=2.2.0->rich->keras) (0.1.2)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\anike\anaconda3\lib\site-packages
(from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\anike\anaconda3\lib\site-packages (fro
m requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (1.26.11)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anike\anaconda3\lib\site-packages (from request
s<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anike\anaconda3\lib\site-packages (from r
equests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2022.9.14)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\anike\anaconda3\lib\si
te-packages (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\anike\anaconda3\lib\site-packages (from tens
orboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (2.0.3)
Requirement already satisfied: markdown>=2.6.8 in c:\users\anike\anaconda3\lib\site-packages (from tens
orboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (3.3.4)
```

```
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\anike\anaconda3\lib\site-packages
(from packaging->tensorflow-intel==2.16.1->tensorflow) (3.0.9)
```

In [7]:
```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
```

Tokenization and Lemmatization

The script reads a JSON file containing intents for the chatbot. Each intent consists of patterns, which are sentences or phrases users might type in. It loops through each intent and its associated patterns.

For each pattern, it tokenizes the sentence into individual words using NLTK's word_tokenize() function. Tokenization involves breaking a text into individual words or tokens.

It adds these words to a list called words.

It also creates a list called documents, which contains tuples of words and their corresponding intents. This will be used later for training the model. Additionally, it maintains a list called classes, which stores all the unique intents.

In [8]:
```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
```

Data Preparation:

After collecting all the words, it performs lemmatization on them. Lemmatization is the process of reducing words to their base or dictionary form (lemma).

It converts all words to lowercase to ensure consistency. Duplicate words are removed from the list, resulting in a list of unique words.

The script saves the unique words and intents using the pickle module for future use.

In [16]:

```
import json
import nltk

words = []
classes = []
documents = []
ignore_letters = ['!', '?', ',', '.']

# Open the JSON file with UTF-8 encoding
with open('C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Chatbot/chat_json.json', encoding='
    intents_data = json.load(intents_file)

# Loop through intents and their patterns
for intent in intents_data['intents']:
    for pattern in intent['patterns']:
        # Tokenize each word
        word = nltk.word_tokenize(pattern)
        words.extend(word)
        # Add documents in the corpus
        documents.append((word, intent['intent']))  # Use 'intent' instead of 'tag'
        # Add to our classes list
        if intent['intent'] not in classes:
            classes.append(intent['intent'])

print(documents)
print(classes)
print(words)
```

```
[(['hi'], 'greet'), (['hello'], 'greet'), (['hey'], 'greet'), (['good', 'morning'], 'greet'), (['good',
'afternoon'], 'greet'), (['good', 'evening'], 'greet'), (['hiya'], 'greet'), (['bye'], 'goodbye'), (['g
oodbye'], 'goodbye'), (['see', 'you'], 'goodbye'), (['talk', 'to', 'you', 'later'], 'goodbye'), (['fare
well'], 'goodbye'), (['thanks'], 'thanks'), (['thank', 'you'], 'thanks'), (['appreciate', 'it'], 'thank
s'), (['thanks', 'a', 'lot'], 'thanks'), (['thank', 'you', 'very', 'much'], 'thanks'), (['who', 'are',
'you'], 'who_are_you'), (['what', 'is', 'your', 'name'], 'who_are_you'), (['who', 'am', 'I', 'talking',
'to'], 'who_are_you'), (['introduce', 'yourself'], 'who_are_you'), (['how', 'are', 'you'], 'how_are_yo
u'), (['how', "'s", 'it', 'going'], 'how_are_you'), (['how', 'do', 'you', 'do'], 'how_are_you'), (['ho
w', 'have', 'you', 'been'], 'how_are_you'), (['what', 'can', 'you', 'do'], 'what_can_you_do'), (['how',
'can', 'you', 'help', 'me'], 'what_can_you_do'), (['what', 'are', 'your', 'capabilities'], 'what_can_yo
u_do'), (['what', 'services', 'do', 'you', 'offer'], 'what_can_you_do'), (['help'], 'help'), (['I', 'ne
ed', 'help'], 'help'), (['can', 'you', 'help', 'me'], 'help'), (['assistance'], 'help'), (['support'],
'help'), (['tell', 'me', 'a', 'joke'], 'tell_joke'), (['make', 'me', 'laugh'], 'tell_joke'), (['joke'],
'tell_joke'), (['say', 'something', 'funny'], 'tell_joke'), (['what', "'s", 'the', 'weather'], 'weathe
r'), (['weather', 'update'], 'weather'), (['how', "'s", 'the', 'weather'], 'weather'), (['current', 'we
ather'], 'weather'), (['what', "'s", 'the', 'news'], 'news'), (['news', 'update'], 'news'), (['curren
t', 'news'], 'news'), (['latest', 'news'], 'news'), (['what', "'s", 'up'], 'small_talk'), (['what',
"'s", 'new'], 'small_talk'), (['anything', 'interesting'], 'small_talk'), (['tell', 'me', 'something'],
'small_talk'), (['you', "'re", 'awesome'], 'compliment'), (['great', 'job'], 'compliment'), (['you',
"'re", 'the', 'best'], 'compliment'), (['nice', 'work'], 'compliment'), (['you', "'re", 'useless'], 'in
sult'), (['you', 'suck'], 'insult'), (['you', "'re", 'terrible'], 'insult'), (['bad', 'bot'], 'insul
t')]
['greet', 'goodbye', 'thanks', 'who_are_you', 'how_are_you', 'what_can_you_do', 'help', 'tell_joke', 'w
eather', 'news', 'small_talk', 'compliment', 'insult']
['hi', 'hello', 'hey', 'good', 'morning', 'good', 'afternoon', 'good', 'evening', 'hiya', 'bye', 'goodb
ye', 'see', 'you', 'talk', 'to', 'you', 'later', 'farewell', 'thanks', 'thank', 'you', 'appreciate', 'i
t', 'thanks', 'a', 'lot', 'thank', 'you', 'very', 'much', 'who', 'are', 'you', 'what', 'is', 'your', 'n
ame', 'who', 'am', 'I', 'talking', 'to', 'introduce', 'yourself', 'how', 'are', 'you', 'how', "'s", 'i
t', 'going', 'how', 'do', 'you', 'do', 'how', 'have', 'you', 'been', 'what', 'can', 'you', 'do', 'how',
'can', 'you', 'help', 'me', 'what', 'are', 'your', 'capabilities', 'what', 'services', 'do', 'you', 'of
fer', 'help', 'I', 'need', 'help', 'can', 'you', 'help', 'me', 'assistance', 'support', 'tell', 'me',
'a', 'joke', 'make', 'me', 'laugh', 'joke', 'say', 'something', 'funny', 'what', "'s", 'the', 'weathe
r', 'weather', 'update', 'how', "'s", 'the', 'weather', 'current', 'weather', 'what', "'s", 'the', 'new
s', 'news', 'update', 'current', 'news', 'latest', 'news', 'what', "'s", 'up', 'what', "'s", 'new', 'an
ything', 'interesting', 'tell', 'me', 'something', 'you', "'re", 'awesome', 'great', 'job', 'you', "'r
e", 'the', 'best', 'nice', 'work', 'you', "'re", 'useless', 'you', 'suck', 'you', "'re", 'terrible', 'b
ad', 'bot']
```

Bag of Words Creation:

The script initializes an empty list called training to hold training data. It iterates through each document in documents.

For each document, it creates a bag of words representation. This is a binary vector indicating which words from the vocabulary are present in the current document.

It creates an output row, which is a list of zeros with a one at the index corresponding to the intent of the document.

The document's bag of words and output row are appended to the training list. The training data is shuffled randomly to prevent the model from learning any order dependencies.

In [29]:

```
# lemmaztize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_letters]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)
pickle.dump(words,open('C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Chatbot/words.pkl','wb
pickle.dump(classes,open('C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Chatbot/classes.pkl'
```

```
58 documents
13 classes ['compliment', 'goodbye', 'greet', 'help', 'how_are_you', 'insult', 'news', 'small_talk', 't
ell_joke', 'thanks', 'weather', 'what_can_you_do', 'who_are_you']
79 unique lemmatized words ["'re", "'s", 'a', 'afternoon', 'am', 'anything', 'appreciate', 'are', 'assi
stance', 'awesome', 'bad', 'been', 'best', 'bot', 'bye', 'can', 'capability', 'current', 'do', 'evenin
g', 'farewell', 'funny', 'going', 'good', 'goodbye', 'great', 'have', 'hello', 'help', 'hey', 'hi', 'hi
ya', 'how', 'i', 'interesting', 'introduce', 'is', 'it', 'job', 'joke', 'later', 'latest', 'laugh', 'lo
t', 'make', 'me', 'morning', 'much', 'name', 'need', 'new', 'news', 'nice', 'offer', 'say', 'see', 'ser
vice', 'something', 'suck', 'support', 'talk', 'talking', 'tell', 'terrible', 'thank', 'thanks', 'the',
'to', 'up', 'update', 'useless', 'very', 'weather', 'what', 'who', 'work', 'you', 'your', 'yourself']
```

In [22]:

```
# Create output labels
output_empty = [0] * len(classes)

# Initialize training data
training = []

# Create bag of words for each sentence
for doc in documents:
    bag = []
    pattern_words = doc[0]
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    for word in words:
        bag.append(1) if word in pattern_words else bag.append(0)
    # Create output row
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    training.append([bag, output_row])
```

In [24]:

```
# Shuffle the training data
random.shuffle(training)

# Separate features and labels
train_x = []
train_y = []

for features, label in training:
    train_x.append(features)
    train_y.append(label)

# Convert lists to numpy arrays
train_x = np.array(train_x)
train_y = np.array(train_y)
```

Model Creation:

The neural network model is defined using Keras Sequential API. This API allows stacking of layers sequentially.

The model consists of an input layer, two hidden layers, and an output layer. The input layer has neurons equal to the length of the bag of words representation.

Two hidden layers with ReLU activation are added to introduce non-linearity. Dropout layers are added after each hidden layer to prevent overfitting by randomly setting a fraction of input units to zero during training.

In [25]:

```
# Define the 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'))
```

```
C:\Users\anike\anaconda3\lib\site-packages\keras\src\layers\core\dense.py:86: UserWarning: Do not pass
an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input
(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model Compilation and Training:

The model is compiled with categorical cross-entropy loss function, which is suitable for multi-class classification problems.

The Adam optimizer is used for optimization, which is an extension to stochastic gradient descent (SGD).

The model is then trained on the training data for a specified number of epochs (iterations over the entire dataset) and with a specified batch size (number of samples per gradient update).

Model Saving:

Once trained, the model is saved to a file using the save() method provided by Keras.

Additionally, other necessary files like words, classes, and training data are saved using pickle for later use.

In [27]:

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

# Train the model
history = model.fit(np.array(train_x), np.array(train_y), epochs=100, batch_size=5, verbose=1)

# Save the trained model and other necessary files
model.save('C:/Users/anike/OneDrive/Desktop/Projects/Machine Learning/Chatbot/chatbot_model.h5')
pickle.dump({'words': words, 'classes': classes, 'train_x': train_x, 'train_y': train_y}, open('training
```

```
Epoch 1/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9881 - loss: 0.1175
Epoch 2/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9395 - loss: 0.1932
Epoch 3/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9404 - loss: 0.1746
Epoch 4/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9820 - loss: 0.0818
Epoch 5/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 1.0000 - loss: 0.0742
Epoch 6/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9657 - loss: 0.2008
Epoch 7/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9944 - loss: 0.0797
Epoch 8/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.1143
Epoch 9/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9456 - loss: 0.1592
Epoch 10/100
```

In [28]:

```
print("model created")
```

```
model created
```