Group Detail		
Sr No	**Name**	**ID**
1	Hamza Aslam	191400088
2	Bilal Tariq	191400036
3	Muhammad Uzair	191400013
4	Waqar Ali	191400132
5	Husnain Ali	191400118

1. Introduction of the Project:

This project is aimed at developing a chatbot that can interact with users and respond to their queries. The chatbot is developed using Python and various machine learning and deep learning frameworks such as TensorFlow and Keras. The chatbot uses Natural Language Processing (NLP) techniques to understand the user's intent and generate appropriate responses.

Overview:

The project consists of two main parts: the training of the chatbot and the actual interaction with the user.

Training the Chatbot:

The chatbot is trained on a dataset of user queries and responses. The dataset is provided in the form of a JSON file containing various intents and their corresponding patterns and responses. The training data is preprocessed and then used to train a deep-learning model using the Keras library. The model consists of an embedding layer, a GlobalAveragePooling1D layer, and multiple Dense layers. The model is trained using the sparse categorical cross-entropy loss function and the Adam optimizer.

Once the model is trained, it is saved along with the tokenizer and label encoder objects using the pickle library.

Interaction with the User:

The trained chatbot can be used to interact with the user by loading the saved model and associated objects. The user's input is processed using the tokenizer object and then passed through the model to obtain the predicted intent. The predicted intent is then used to generate an appropriate response from the dataset of responses.

2. System Requirements of the Project:

To run this project, the following system requirements must be fulfilled:

- Python 3.x
- TensorFlow library
- Keras library
- Scikit-learn library
- JSON library

3. **Python Coding:**

The Python code for the chatbot project consists of two parts:

- Training the Model: In this part, we have loaded the training data from the "intents.json" file and preprocessed it using various techniques, such as tokenization and padding. Then, we have created a sequential model using the Keras library and trained it on the preprocessed data. Finally, we have saved the trained model, tokenizer, and label encoder to use in the second part.
- Chatting with the Bot: In this part, we have loaded the saved model, tokenizer, and label encoder. Then, we have implemented a loop to take user input and predict the response using the loaded model. The predicted response is then displayed to the user.

```
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, GlobalAveragePooling
1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder

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_classes = len(labels)

#lable encoder use converting lables in numric form
lbl_encoder = LabelEncoder()
```

```
#'fit' method trains the algorithm on the training data, after the model i
s initialized
lbl_encoder.fit(training_labels)

training_labels = lbl_encoder.transform(training_labels)

vocab_size = 1000
embedding_dim = 16
max_len = 20
oov_token = "<OOV>"

# tokenization basically refers to splitting up a larger body of text into
 smaller words or lines
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(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()
epochs = 500
history = model.fit(padded_sequences, np.array(training_labels), epochs=ep
ochs)


# to save the trained model
model.save("chat_model")

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)

import json
import numpy as np
from sklearn.preprocessing import LabelEncoder
import colorama
from colorama import Fore, Style, Back
import random
import pickle

colorama.init()

with open("intents.json") as file:
    data = json.load(file)


def chat():
    # load trained model
    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)

    # parameters
    max_len = 20

    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 i in data['intents']:
            if i['tag'] == tag:
                print(Fore.GREEN + "ChatBot:" + Style.RESET_ALL , np.rando
m.choice(i['responses']))

        # print(Fore.GREEN + "ChatBot:" + Style.RESET_ALL,random.choice(re
sponses))

print(Fore.YELLOW + "Start messaging with the bot (type quit to stop)!" +
Style.RESET_ALL)
chat()
```

4. **Output of the Project:**

   The chatbot created in this project can be used to communicate with users and provide responses to their queries. The chatbot has been trained on various intents, such as greetings, goodbyes, and product-related queries. The output of the project is a chatbot that can respond to user queries in a natural language format.

5. **References:**

- TensorFlow documentation: https://www.tensorflow.org/
- Keras documentation: https://keras.io/
- Scikit-learn documentation: https://scikit-learn.org/stable/documentation.html
- JSON documentation: https://docs.python.org/3/library/json.html

Conclusion:

This project demonstrates the development of a chatbot using deep learning techniques and the Python programming language. The chatbot is capable of understanding the user's intent and generating appropriate responses. The chatbot can be further improved by increasing the size and diversity of the training dataset and by using more advanced deep-learning models.