In [ ]:
```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
```

```
WARNING:tensorflow:From C:\Users\aswan\AppData\Roaming\Python\Python311\site-package
s\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is depre
cated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [ ]:
```
bot_dataset = pd.read_csv("topical_chat.csv")
bot_dataset.head()
```

Out[ ]:

	conversation_id	message	sentiment
0	1	Are you a fan of Google or Microsoft?	Curious to dive deeper
1	1	Both are excellent technology they are helpfu...	Curious to dive deeper
2	1	I'm not a huge fan of Google, but I use it a...	Curious to dive deeper
3	1	Google provides online related services and p...	Curious to dive deeper
4	1	Yeah, their services are good. I'm just not a...	Curious to dive deeper

In [ ]:
```
nltk.download('punkt')
nltk.download('stopwords')
```

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

Out[ ]:
```
True
```

In [ ]:
```
def preprocess_text(text):
    tokens = word_tokenize(text)
    tokens = [word.lower() for word in tokens if word.isalnum() and word.lower() no
    return " ".join(tokens)
```

In [ ]:
```
bot_dataset["processed_message"] = bot_dataset["message"].apply(preprocess_text)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
```

In [ ]:
```
X = bot_dataset["processed_message"]
y = bot_dataset["sentiment"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [ ]:
```
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [ ]:
```
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

In [ ]:
```
tokenizer = Tokenizer(num_words=5000, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)
```

In [ ]:
```
X_train_sequences = tokenizer.texts_to_sequences(X_train)
X_test_sequences = tokenizer.texts_to_sequences(X_test)

X_train_padded = pad_sequences(X_train_sequences, maxlen=100, padding="post", trunc
X_test_padded = pad_sequences(X_test_sequences, maxlen=100, padding="post", truncat
```

In [ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
```

In [ ]:
```
model = Sequential([
    Embedding(input_dim=5000, output_dim=128, input_length=100),
    LSTM(128, return_sequences=True),
    LSTM(64),
    Dense(64, activation='relu'),
    #Dropout(0.5),
    Dense(8, activation='linear')
])
```

In [ ]:
```
model.compile(
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer=tf.keras.optimizers.Adam(0.001),
    metrics=['accuracy']
)
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 100, 128)          640000

 lstm_2 (LSTM)               (None, 100, 128)          131584

 lstm_3 (LSTM)               (None, 64)                49408

 dense_2 (Dense)             (None, 64)                4160

 dense_3 (Dense)             (None, 8)                 520

=================================================================
Total params: 825672 (3.15 MB)
Trainable params: 825672 (3.15 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
model.fit(X_train_padded, y_train_encoded, epochs=5, batch_size=45, validation_spli
```

```
Epoch 1/5
WARNING:tensorflow:From C:\Users\aswan\AppData\Roaming\Python\Python311\site-package
s\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecate
d. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\aswan\AppData\Roaming\Python\Python311\site-package
s\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_fu
nctions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions i
nstead.

3015/3015 [==============================] - 387s 127ms/step - loss: 1.4362 - accura
cy: 0.4288 - val_loss: 1.4249 - val_accuracy: 0.4328
Epoch 2/5
3015/3015 [==============================] - 366s 122ms/step - loss: 1.4332 - accura
cy: 0.4288 - val_loss: 1.4240 - val_accuracy: 0.4328
Epoch 3/5
3015/3015 [==============================] - 369s 122ms/step - loss: 1.4326 - accura
cy: 0.4288 - val_loss: 1.4243 - val_accuracy: 0.4328
Epoch 4/5
3015/3015 [==============================] - 363s 120ms/step - loss: 1.4327 - accura
cy: 0.4288 - val_loss: 1.4239 - val_accuracy: 0.4328
Epoch 5/5
3015/3015 [==============================] - 362s 120ms/step - loss: 1.4325 - accura
cy: 0.4288 - val_loss: 1.4240 - val_accuracy: 0.4328
```

Out[ ]:
```
<keras.src.callbacks.History at 0x20d77a0c090>
```

In [ ]:
```
loss, accuracy = model.evaluate(X_test_padded, y_test_encoded)
print("Test accuracy:", accuracy)
```

```
1178/1178 [==============================] - 42s 36ms/step - loss: 1.4316 - accurac
y: 0.4301
Test accuracy: 0.4300881326198578
```

In [ ]:
```
def predict_sentiment(text):
    processed_text = preprocess_text(text)
    sequence = tokenizer.texts_to_sequences([processed_text])
    padded_sequence = pad_sequences(sequence, maxlen=100, padding="post", truncatin
    sentiment_probabilities = model.predict(padded_sequence)
    predicted_sentiment_id = np.argmax(sentiment_probabilities)
    predicted_sentiment = label_encoder.inverse_transform([predicted_sentiment_id])
    return predicted_sentiment

user_input = input("Enter a message: ")
predicted_sentiment = predict_sentiment(user_input)
print("Predicted sentiment:", predicted_sentiment)
```

```
1/1 [==============================] - 1s 891ms/step
Predicted sentiment:  Curious to dive deeper
```

In [ ]:
```
def generate_rule_based_response(predicted_sentiment):
    if predicted_sentiment == "Happy":
        response = "I'm glad to hear that you're feeling happy!"
    elif predicted_sentiment == "Sad":
        response = "I'm sorry to hear that you're feeling sad. Is there anything I
    else:
        response = "I'm here to chat with you. How can I assist you today?"

    return response
```

In [ ]:
```
def generate_rule_based_response_chatbot(user_input):

    predicted_sentiment = predict_sentiment(user_input)
    response = generate_rule_based_response(predicted_sentiment)

    return response
```

In [ ]:
```
def generate_pattern_response(user_input):
    patterns = {
        "hello": "Hello! How can I assist you today?",
        "how are you": "I'm just a chatbot, but I'm here to help! How can I assist
        "help": "Sure, I'd be happy to help. What do you need assistance with?",
        "bye": "Goodbye! If you have more questions in the future, feel free to ask

    }


    for pattern, response in patterns.items():
        if pattern in user_input.lower():
            return response


    return generate_rule_based_response_chatbot(user_input)

while True:
    user_input = input("You: ")
    if user_input.lower() == "exit":
        print("Bot: Goodbye!")
        break
```

```
    bot_response = generate_pattern_response(user_input)
    print("Bot:", bot_response)
```

In [ ]:
```
print(bot_dataset.info())
print(bot_dataset["sentiment"].value_counts())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188378 entries, 0 to 188377
Data columns (total 4 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   conversation_id    188378 non-null  int64 
 1   message            188378 non-null  object
 2   sentiment          188378 non-null  object
 3   processed_message  188378 non-null  object
dtypes: int64(1), object(3)
memory usage: 5.7+ MB
None
sentiment
 Curious to dive deeper    80888
 Neutral                   41367
 Surprised                 30638
 Happy                     29617
 Sad                        2533
 Disgusted                  1433
 Fearful                    1026
 Angry                       876
Name: count, dtype: int64
```