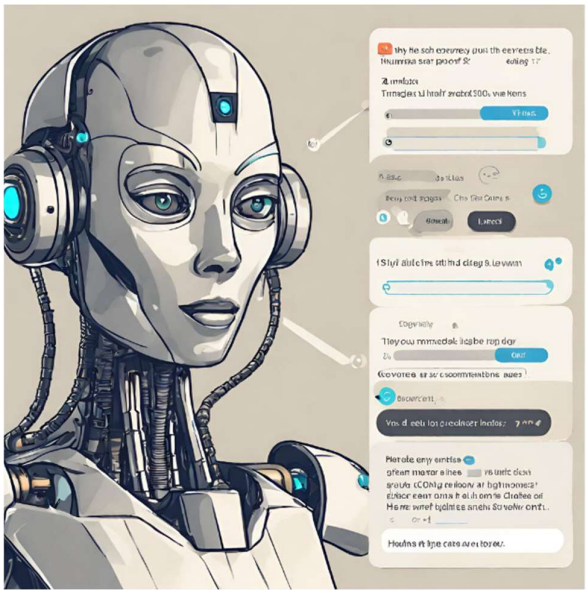
Phase-5: Project Documentation & Submission

PROJECT-3 CREATE A CHATBOT USING PYTHON

PROBLEM DEFINITION:

"The challenge is to create a chatbot in Python that provides exceptional customer service, answering user queries on a website or application. The objective is to deliver high-quality support to users, ensuring a positive user experience and customer satisfaction."

Answering customer queries can be tedious and it involves a lot of skilled technicians working behind to provide high quality support. The customer care is always on top of a company's list of priorities.



Now in the era of automation, where AI has taken over most of the tedious tasks that previously required a lot of labour power, we are witnessing profound transformations. Now it is very much possible to construct a Chat-Bot, powered by NLP (natural language processing) & NLU (natural language understanding) to provide answers to customer queries.

The chatbot will try to interpret the customer's input, analyse it, and will look for a possible solution from the existing data set and provide answer to the query. The chatbot will provide customized answers to the user queries by analysing the customer data.

DESIGN THINKING:

Functionality:

Simply put, the chat bot will be able to answer basic customer queries. It will for sure politely greet and thank the customer. If the customer faces any issue and he/she seeks the help of the chatbot, it will provide them with guidance. It will help the user to efficiently manage his/her resources.

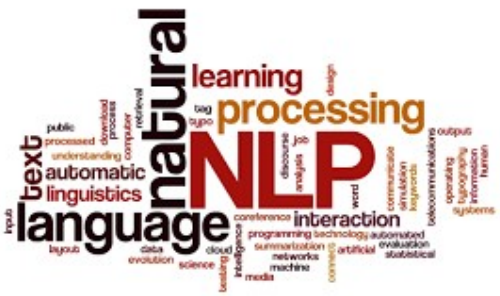
User Interface:

Our designed chatbot can be integrated to any application that provides service to customers. The chatbot will be a separate section in the app that can be used by the customers to seek guidance. This chatbot will be granted access to user's app data to help the customer with the issue

Natural Language Processing (NLP):

To train our chatbot understand/recognize various user inputs it is necessary to implement NLP&NLU techniques.

As the chatbot is created using python, python libraries like NLTK (natural language tool kit) and RASA-NLU can be used to analyse the user inputs. It will help the chatbot to process the output in a conversational manner.



Responses:

The response to user queries will be based on the dataset provided by Kaggle.com. The data set comprises simple dialogues that acts as a reference to the user inputs. The output of the chatbot will be directly based on the existing data set. The accuracy of the answers will be high, if the user query is related to the dialogues in the existing data set.

Integration:

The chatbot will be an essential part of the customer's app. The chatbot feature can be accessed by the user 24/7 to seek guidance. It will be a feature available to the user on a click.

Testing and Improvement:

The chatbot will learn to improve alongside its implementation. However serious monitoring of chatbot's responses is necessary. User feedback about the chatbot can be collected from the customers to improve its performance.

Innovation

We have been asked to utilize or integrate pretrained language models into our chatbot as a step towards innovation. This deals with understanding pretrained language models, interfacing it & making use of it efficiently.

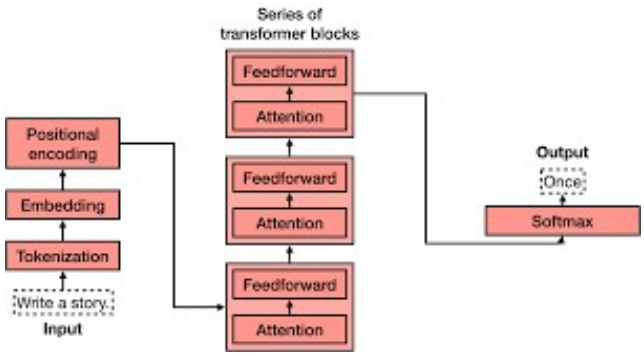
Pretrained Language Models:

Pretrained language models, such as GPT-3 or GPT-4, are advanced artificial intelligence models that have undergone extensive training on vast amounts of text data from the internet. This training equips them with a deep understanding of human language, allowing them to generate coherent and contextually relevant text. They possess the ability to comprehend complex language structures, extract meaning from text, and generate human-like responses. This makes them highly versatile for various applications, including chatbots.

These are basically a deep learning model, that uses NLP to tokenize(split) user input into meaningful words and sentences, interpreting user emotion/sentiment and provides response to user input based on its data source.

Role: Pretrained language models play a pivotal role in chatbot development by serving as the language understanding and generation engine. They enable chatbots to process and respond to user inputs in a more human-like and contextually relevant manner.

Efficient pretrained models: A Model built using 'Transformer' architecture aids us effectively in building a chatbot. The transformer architecture is highly regarded in natural language processing due to its efficient parallel processing, ability to capture long-range dependencies, bidirectional context understanding, and versatility through pretraining and fine-tuning. It consistently achieves state-of-the-art results, is open source, and can be scaled for improved performance, making it one of the best models in NLP.



Transformer models:

Some of the transformer models that suit our project and meet our requirements and goals are

- GPT (Generative Pretrained Transformer)
- BERT (Bidirectional Encoding Representation from Transformers)

1. **GPT (Generative Pretrained Transformer):**

- GPT models, including GPT-3 and GPT-4, are built on the transformer architecture.

- They are primarily designed for autoregressive language modelling, which means they predict the next word or token in a sequence based on the context of the previous words.

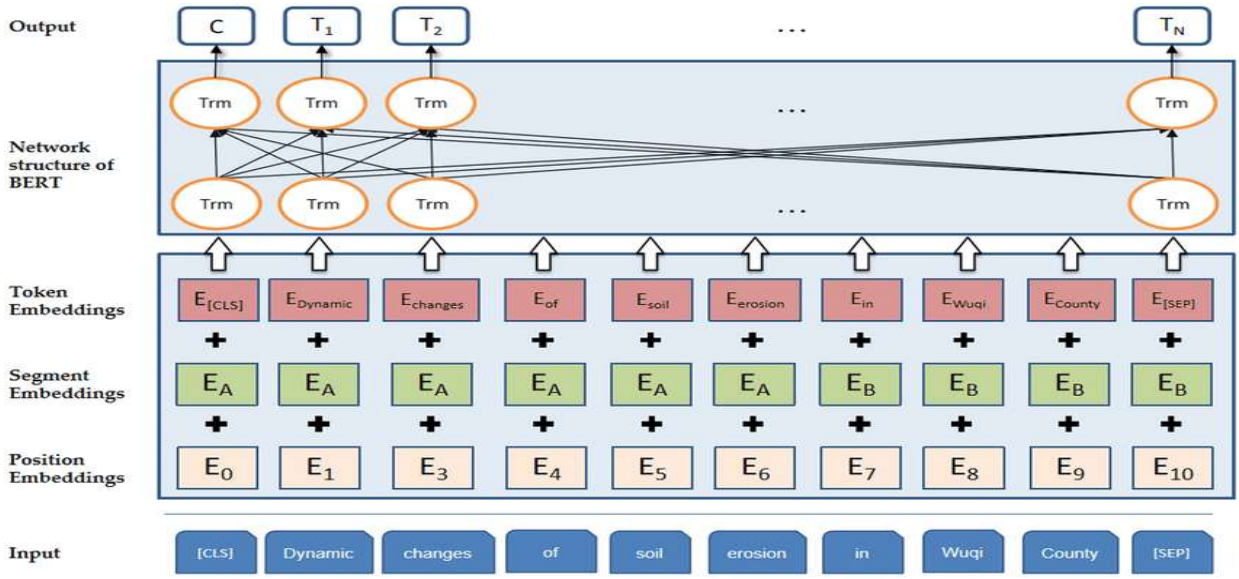
- GPT models are unidirectional, meaning they process text from left to right (or right to left) but not both directions simultaneously.

2. **BERT (Bidirectional Encoder Representations from Transformers):**

- BERT is another transformer-based model, but it focuses on bidirectional language understanding.

- Unlike GPT, BERT looks at both the left and right context of a word when encoding it, making it contextually aware in both directions.

- BERT's bidirectional approach has made it highly effective for various NLP tasks, including text classification, named entity recognition, and question answering.



Choosing the right model:

GPT focuses on language generation and is renowned for its text generation capabilities. It's trained to predict the next word in a sequence, making it adept at creative content generation, chatbots, and text completion tasks. However, GPT may struggle with understanding nuanced context, as it processes text unidirectionally.

On the other hand, BERT emphasizes bidirectional context understanding. It's trained to predict missing words within a sentence, enabling it to capture the meaning and context of words more comprehensively. BERT shines in tasks requiring deep language understanding, such as sentiment analysis, question answering, and named entity recognition.

In summary, GPT excels at text generation, while BERT is superior in language understanding tasks. The choice between them depends on the specific project requirements.

For a chatbot, that addresses user query and involves in a simple conversation with the user GPT is ideal, whereas the help of BERT is preferred for a chatbot that is customized to reply to user-input by referring to a context/source.

Significance:

By leveraging pretrained models, chatbots can hold more engaging and coherent conversations, which leads to higher user satisfaction and improved effectiveness in various domains.

Interfacing the models:

API method:

As the chatbot is built using python, some of the models can be accessed by importing a specific library in python. For e.g., GPT model can be accessed by importing the open ai library.

The models are pretrained using data from the internet. And can be accessed using APIs. These APIs (Application Program Interface) bridge the chatbot program and the trained model. A unique API key is required by the program to access the model online.

The API keys almost always incur cost, as there is a billing procedure followed to capitate the no. of tokens a user generates

Accessing stored models:

In this method, a trained model is accessed from the storage space. This requires downloading an open-source model from internet, and using it to generate & interpret text.

This method is cost-effective but highly trained models take up a lot of disk space.

Advantages:

Integrating pretrained models into chatbots offers several key advantages, including:

- Improved conversational abilities: Chatbots can hold more natural and contextually relevant conversations.
- Enhanced user experiences: Users are more satisfied with chatbot interactions.
- Reduced development time: Leveraging pretrained models speeds up chatbot development.

DEVELOPMENT:

In this phase of project submission, we are all set to develop a program using python to setup a chatbot. In the previous phases we have acquired enough knowledge about the basics of a chatbot and about some innovations that could make it even better.

To start with we are provided a dataset from Kaggle which is a text file comprising set of dialogues. We are about to process the dataset and build a simple chatbot that would reply to your prompt by comparing it to the existing dataset.

DATA CLEANING:

The dataset provided must be processed prior feeding it to the chatbot program. This essential step is known as 'Data Cleaning'. Data can be sorted using the pandas library in python.

The steps are as follows:

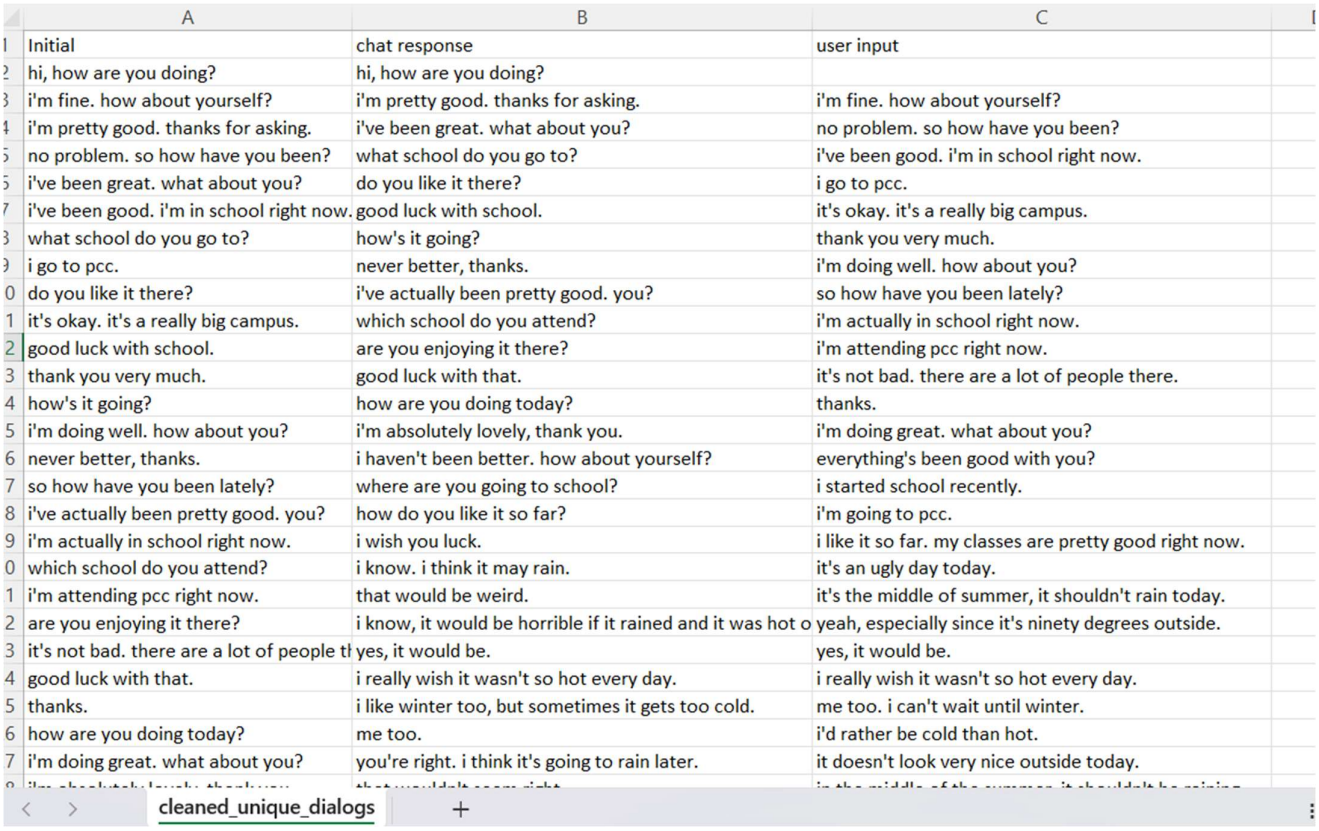
1. Read the text file using pandas library.
2. Store the dialogues as a string in a new csv file under the column ' initial' .
3. Use the unique() function to filter out the repeating values.

4. Iterate through the values of the 'initial' column and store them under 2 new columns- chatbot_response and user_input.
5. Open the CSV file in excel to check the format of the data stored and insert an empty cell in the first row of user_input. This helps in initialization of chatbot's first response later while running the code.

CODE SNIPPET:

```
import pandas as pd

dataset_path = "C:\\Users\\shiva\\Downloads\\archive\\dialogs.txt"

with open(dataset_path, 'r', encoding='utf-8') as file:
    lines = file.readlines()

cleaned_lines = []
for line in lines:
    parts = line.strip().split('\t')
    if len(parts) == 2:
        cleaned_lines.append(parts[0])
        cleaned_lines.append(parts[1])
data = {'Initial': cleaned_lines}
df = pd.DataFrame(data)
df.drop_duplicates(subset='Initial', inplace=True)

cleaned_csv_path =
"C:\\Users\\shiva\\Downloads\\archive\\cleaned_unique_dialogs.csv"

df.to_csv(cleaned_csv_path, index=False)
df = pd.read_csv(cleaned_csv_path)
df['chat_response'] = df.iloc[::2, 0].reset_index(drop=True)
df['user_input'] = df.iloc[1::2, 0].reset_index(drop=True)
df.to_csv(csv_path, index=False)
```

OUTPUT:



	A	B	C
1	Initial	chat response	user input
2	hi, how are you doing?	hi, how are you doing?	
3	i'm fine. how about yourself?	i'm pretty good. thanks for asking.	i'm fine. how about yourself?
4	i'm pretty good. thanks for asking.	i've been great. what about you?	no problem. so how have you been?
5	no problem. so how have you been?	what school do you go to?	i've been good. i'm in school right now.
6	i've been great. what about you?	do you like it there?	i go to pcc.
7	i've been good. i'm in school right now.	good luck with school.	it's okay. it's a really big campus.
8	what school do you go to?	how's it going?	thank you very much.
9	i go to pcc.	never better, thanks.	i'm doing well. how about you?
10	do you like it there?	i've actually been pretty good. you?	so how have you been lately?
11	it's okay. it's a really big campus.	which school do you attend?	i'm actually in school right now.
12	good luck with school.	are you enjoying it there?	i'm attending pcc right now.
13	thank you very much.	good luck with that.	it's not bad. there are a lot of people there.
14	how's it going?	how are you doing today?	thanks.
15	i'm doing well. how about you?	i'm absolutely lovely, thank you.	i'm doing great. what about you?
16	never better, thanks.	i haven't been better. how about yourself?	everything's been good with you?
17	so how have you been lately?	where are you going to school?	i started school recently.
18	i've actually been pretty good. you?	how do you like it so far?	i'm going to pcc.
19	i'm actually in school right now.	i wish you luck.	i like it so far. my classes are pretty good right now.
20	which school do you attend?	i know. i think it may rain.	it's an ugly day today.
21	i'm attending pcc right now.	that would be weird.	it's the middle of summer, it shouldn't rain today.
22	are you enjoying it there?	i know, it would be horrible if it rained and it was hot o	yeah, especially since it's ninety degrees outside.
23	it's not bad. there are a lot of people th	yes, it would be.	yes, it would be.
24	good luck with that.	i really wish it wasn't so hot every day.	i really wish it wasn't so hot every day.
25	thanks.	i like winter too, but sometimes it gets too cold.	me too. i can't wait until winter.
26	how are you doing today?	me too.	i'd rather be cold than hot.
27	i'm doing great. what about you?	you're right. i think it's going to rain later.	it doesn't look very nice outside today.

cleaned_unique_dialogs

CHATBOT DESIGNING:

A simple chatbot can be programmed using the 'dictionary' approach. Wherein the values from the dataset are initialized as a dictionary datatype. The user input are stored as keys while the corresponding chat response is stored as a value in the dictionary.

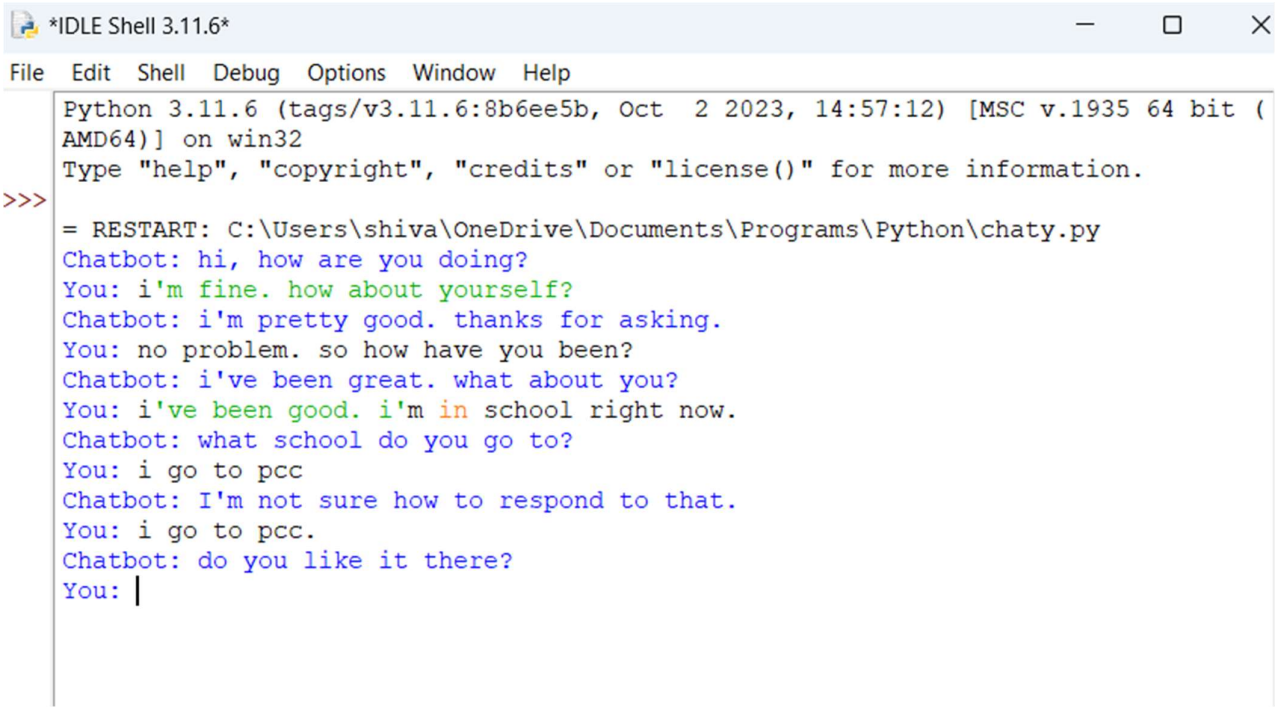
So when a user prompts a dialog similar to the dialogues in the dataset, the chatbot responds by returning the value for the specific key.

CODE SNIPPET:

```
import pandas as pd


dataset =
pd.read_csv("C:\\Users\\shiva\\Downloads\\archive\\cleaned_unique_dialogs
.csv")
```

```
responses = dict(zip(dataset['user input'], dataset['chat response']))
def get_chatbot_response(user_input):
    user_input = user_input.lower()
    return responses.get(user_input, "I'm not sure how to respond to
                                                                that.")

first_dialogue = dataset['user input'].iloc[0]
chatbot_first_response = dataset['chat response'].iloc[0]
print("Chatbot:", chatbot_first_response)

while True:
    user_input = input("You: ")
    chatbot_response = get_chatbot_response(user_input)
    print("Chatbot:", chatbot_response)
```

OUTPUT:



NOTE:

This is a very simple approach to build a chatbot, hence the concept of tokenization in NLU is not applied here. The chat bot's capabilities are very limited and can only reply with the basic templates provided by the dataset. The chatbot is incapable of understanding user intentions as it is not equipped with NLU libraries.

We intend to develop the chatbot capabilities by integrating NLU libraries and will give a try to integrate the chatbot with a web application.

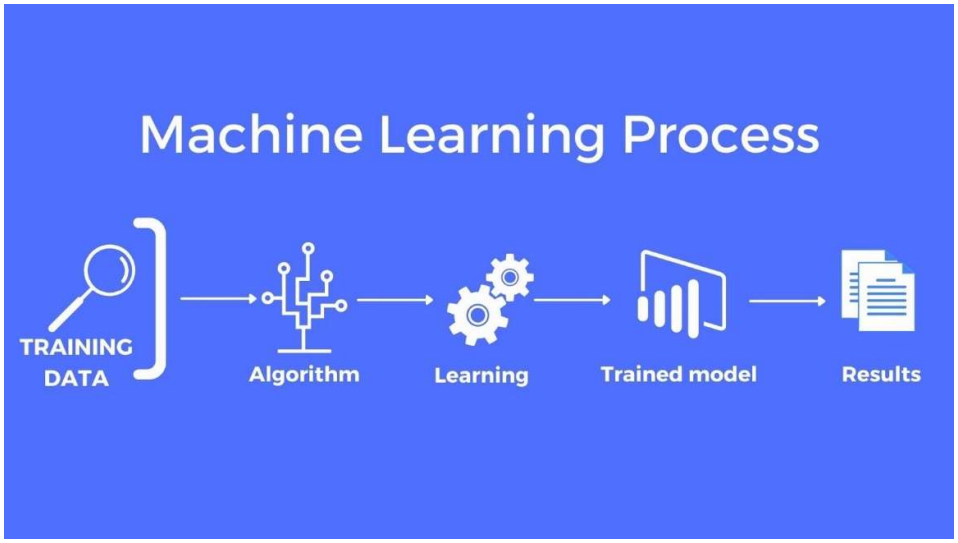
MACHINE LEARNING MODEL:

A machine learning model is a mathematical representation of a problem domain that, through training on data, acquires the ability to generalize and make predictions or classifications on new, unseen data. It learns from historical examples, identifying patterns and relationships within the data to perform tasks like prediction, classification, or decision-making.

Machine learning manages data by splitting it into training, validation, and test datasets.

- *Training data* is used to train the model by exposing it to a large set of examples.
- *Validation data* is used to fine-tune the model and optimize hyperparameters, ensuring it generalizes well.
- *Test data* is kept separate and used to evaluate the model's performance on unseen data, providing an unbiased assessment of its effectiveness.

This data management strategy ensures that the model learns from one set of data, validates on another, and is ultimately tested on a different dataset to assess its real-world applicability.



Our Helping Hand:

The machine learning model aids the chatbot by enabling it to understand and generate human-like responses. The model is trained on a dataset containing user inputs and corresponding chatbot responses. It learns the language patterns and context from this training data, allowing the chatbot to provide relevant and coherent replies to user queries. The model's ability to generalize from the training data helps the chatbot engage in dynamic and context-aware conversations, making it a more effective and responsive conversational agent.

Tensor Flow is our chosen ML framework, we are about to work with. Before diving in let's take a quick look at why tensor flow is an apt ML frame work for our purpose.

TENSOR FLOW:

TensorFlow is a popular open-source machine learning framework that is widely used for building and training deep learning models, including neural networks for natural language processing tasks like chatbots.

- TensorFlow provides a high-level API for building neural network models.
- TensorFlow includes built-in tools for training neural networks. For e.g., the model.compile and model.fit functions are used to define the training process, loss function, and optimizer.
- TensorFlow supports *GPU acceleration*, which can significantly speed up the training of deep learning models. This is crucial for training large chatbot models on extensive datasets

CODE SNIPPET:

```
import tensorflow as tf
import numpy as np
import pandas as pd

file_path="C:\\Users\\shiva\\OneDrive\\Desktop\\cleaned_unique_dialogs.csv"

#DATASET file
df = pd.read_csv(file_path)

# Extract user inputs and chat responses, and clean the text data
user_inputs = df['user_input'].astype(str).str.lower()
chat_responses = df['chat_response'].astype(str).str.lower()

# Tokenization
max_vocab_size = 10000
tokenizer = tf.keras.layers.TextVectorization(max_tokens=max_vocab_size)
tokenizer.adapt(user_inputs + chat_responses)
user_inputs = tokenizer(user_inputs)
chat_responses = tokenizer(chat_responses)

max_sequence_length = 10
user_inputs = tf.keras.preprocessing.sequence.pad_sequences(user_inputs,
maxlen=max_sequence_length, padding='post')
```

```
    chat_responses =
    tf.keras.preprocessing.sequence.pad_sequences(chat_responses,
    maxlen=max_sequence_length, padding='post')


    # Define the model
    model = tf.keras.Sequential([
        tf.keras.layers.Embedding(max_vocab_size, 64, mask_zero=True),
        tf.keras.layers.LSTM(128, return_sequences=True),
        tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(max_vocab_size,

    activation='softmax'))])



    # Compile the model
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',

    metrics=['accuracy'])


    # Training the model
    model.fit(user_inputs, chat_responses, epochs=50)



    # Define a function to generate responses
    def generate_response(input_text):
        input_text = [input_text]
        input_seq = tokenizer(input_text)
        predicted_seq = model.predict(input_seq)
        # Reverse the tokenization process
        predicted_text = " ".join([tokenizer.get_vocabulary()[index] for
    index in
                                               np.argmax(predicted_seq, axis=-
    1)[0]])
        return predicted_text

    while True:
        user_input = input("User: ")
        if user_input.lower() == "exit":
            print("Chatbot: Goodbye!")
            break  # Exit the conversation loop
        response = generate_response(user_input)
        print("Chatbot: " + response)
```

LIMITATIONS:

- This chatbot was trained using a very simple ML model with very less data to support.
- This chatbot may not capture the complexity of human language as well as more advanced models like transformer-based architectures. Real chatbots often use more sophisticated models.

- The model is trained for a fixed number of epochs (50 in this code), but in practice, training might require many more epochs to achieve good performance. Training a chatbot is an ongoing process.
- The code does not include fine-tuning or hyperparameter optimization, which is often required for achieving the best model performance.
- In practice, a chatbot would have a user interface for user interaction, which is not included in this code.

ACHIEVEMENTS:

- The code enables the chatbot to have natural and meaningful conversations with users, improving the user experience.
- The chatbot can understand and maintain context in conversations, providing relevant responses.
- It can be customized for different use cases by training on specific datasets, making it versatile.

OUTPUT: