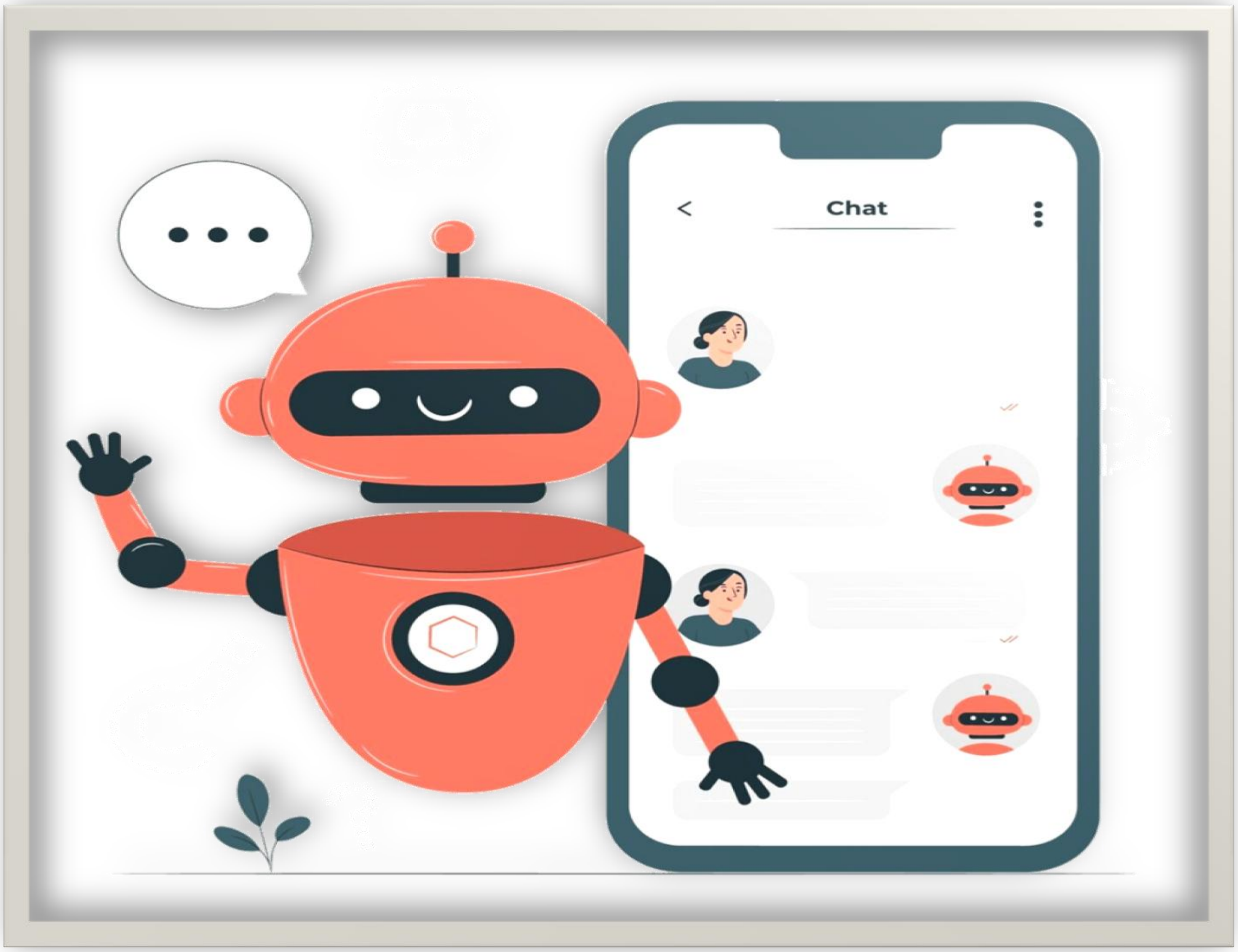
Create a CHATBOT using python

TEAM MEMBER

PHASE_2 Document submission

Project: create a chatBot using python



CHATBOT

INTRODUCTION :

✓ In this phase, we take chatbot innovation to the next level by exploring advanced techniques that leverage pre-trained language models like GPT-3 to enhance the quality of responses. The integration of such models not only elevates the chatbot's conversational abilities but also enables it to generate more contextually relevant and human-like responses.

✓ Pre-trained language models have revolutionized the field of natural language processing (NLP) by learning from vast amounts of text data. GPT-3, for instance, has been trained on diverse internet text and exhibits a remarkable understanding of language nuances. By incorporating these models into your chatbot, you can tap into their vast knowledge and linguistic capabilities to provide users with responses that feel both informative and natural.

✓ The process involves fine-tuning the language model on specific domains or tasks to ensure that it aligns with the chatbot's objectives. This step is crucial for tailoring the model's output to meet the unique needs of your chatbot application, whether it's customer support, content generation, or any other domain-specific task.

✓ By embracing the power of pre-trained language models like GPT-3, you open up exciting opportunities for your chatbot to excel in various conversational scenarios. Users will experience more intelligent, context-aware interactions, leading to increased satisfaction and engagement.

✓ Throughout this guide, we'll delve into the intricacies of integrating and fine-tuning pre-trained language models within your Python-based chatbot, unlocking the potential for innovation in conversation quality and user experience. So, let's dive in and discover how to harness the full potential of advanced language models to take your chatbot to new heights of innovation.

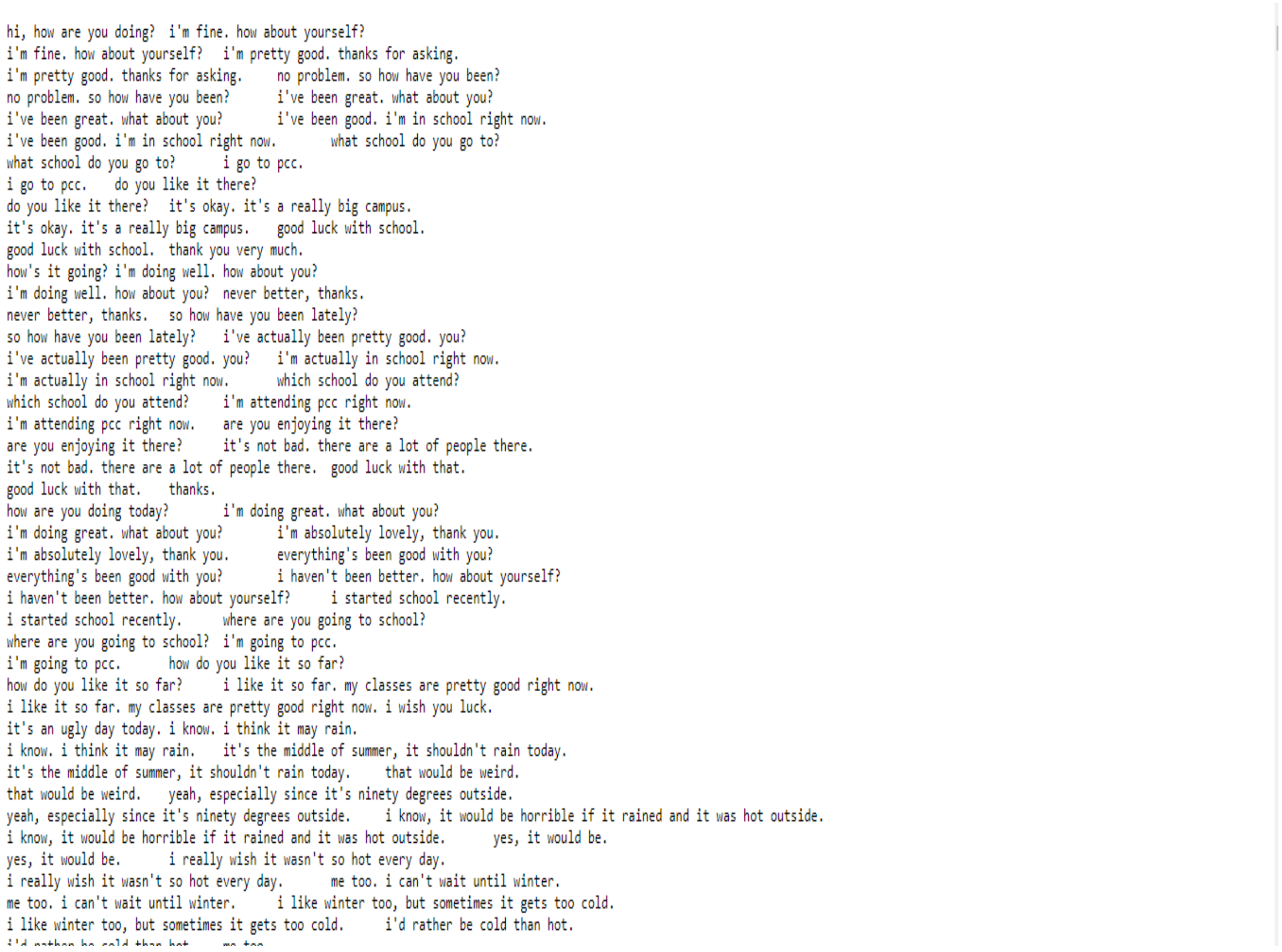
Contents for phase_2 project:

In this phase, consider exploring advanced techniques like using pre-trained language models (e.g., GPT-3) to enhance the quality of responses.

DATA SOURCE:

A good data source for create a CHATBOT using python with advanced techniques natural language processing(NLP),pre-trained language models (e.g., GPT-3) to enhance the quality of responses.

DATASET link(https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot**)**



```
hi, how are you doing?	i'm fine. how about yourself?
i'm fine. how about yourself?	i'm pretty good. thanks for asking.
i'm pretty good. thanks for asking.	no problem. so how have you been?
no problem. so how have you been?	i've been great. what about you?
i've been great. what about you?	i've been good. i'm in school right now.
i've been good. i'm in school right now.	what school do you go to?
what school do you go to?	i go to pcc.
i go to pcc.	do you like it there?
do you like it there?	it's okay. it's a really big campus.
it's okay. it's a really big campus.	good luck with school.
good luck with school.	thank you very much.
how's it going?	i'm doing well. how about you?
i'm doing well. how about you?	never better, thanks.
never better, thanks.	so how have you been lately?
so how have you been lately?	i've actually been pretty good. you?
i've actually been pretty good. you?	i'm actually in school right now.
i'm actually in school right now.	which school do you attend?
which school do you attend?	i'm attending pcc right now.
i'm attending pcc right now.	are you enjoying it there?
are you enjoying it there?	it's not bad. there are a lot of people there.
it's not bad. there are a lot of people there.	good luck with that.
good luck with that.	thanks.
how are you doing today?	i'm doing great. what about you?
i'm doing great. what about you?	i'm absolutely lovely, thank you.
i'm absolutely lovely, thank you.	everything's been good with you?
everything's been good with you?	i haven't been better. how about yourself?
i haven't been better. how about yourself?	i started school recently.
i started school recently.	where are you going to school?
where are you going to school?	i'm going to pcc.
i'm going to pcc.	how do you like it so far?
how do you like it so far?	i like it so far. my classes are pretty good right now.
i like it so far. my classes are pretty good right now.	i wish you luck.
it's an ugly day today.	i know. i think it may rain.
i know. i think it may rain.	it's the middle of summer, it shouldn't rain today.
it's the middle of summer, it shouldn't rain today.	that would be weird.
that would be weird.	yeah, especially since it's ninety degrees outside.
yeah, especially since it's ninety degrees outside.	i know, it would be horrible if it rained and it was hot outside.
i know, it would be horrible if it rained and it was hot outside.	yes, it would be.
yes, it would be.	i really wish it wasn't so hot every day.
i really wish it wasn't so hot every day.	me too. i can't wait until winter.
me too. i can't wait until winter.	i like winter too, but sometimes it gets too cold.
i like winter too, but sometimes it gets too cold.	i'd rather be cold than hot.
i'd rather be cold than hot.	me too.
```

ADVANCED : Pre-trained language models:

Pre-trained language models, like GPT-3 (Generative Pre-trained Transformer 3), represent a significant advancement in natural language processing (NLP) and artificial intelligence. These models are trained on massive datasets containing text from the internet and are designed to understand and generate human-like text in a wide variety of contexts. Here are some key characteristics and examples of pre-trained language models:

1. **GPT-3 (Generative Pre-trained Transformer 3)**:

- Developed by OpenAI, GPT-3 is one of the most well-known pre-trained language models.
- GPT-3 has 175 billion parameters, making it one of the largest and most powerful language models available.
- It can perform a wide range of NLP tasks, such as text generation, language translation, text summarization, question-answering, and more.
- GPT-3 is known for its ability to generate coherent and contextually relevant text, often indistinguishable from human-generated content.

2. **GPT-2 (Generative Pre-trained Transformer 2)**:

- GPT-2 is the predecessor to GPT-3 and is also developed by OpenAI.
- Although smaller in scale than GPT-3, GPT-2 is a powerful language model with 1.5 billion parameters.
- It can generate human-like text and has been used in various creative and practical applications.

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

- Developed by Google, BERT is another influential pre-trained language model.
- BERT is designed to understand the context of words in a sentence by considering both the left and right context.
- It has been widely used for tasks like sentiment analysis, text classification, and named entity recognition.

4. **RoBERTa (A Robustly Optimized BERT Pretraining Approach)**:

- RoBERTa is a variant of BERT developed by Facebook AI.
- It optimizes the training process and has achieved state-of-the-art results on various NLP benchmarks.

5. **XLNet (Generalized Autoregressive Pretraining for Language Understanding)**:

- XLNet is another model developed by Google AI, which extends the autoregressive training of GPT-2 to a permutation-based approach.
- It achieves competitive results on various NLP tasks.

6. **T5 (Text-to-Text Transfer Transformer)**:

- T5 is a pre-trained model developed by Google AI.
- It casts all NLP tasks into a text-to-text format, where input and output are both treated as text.
- T5 has demonstrated strong performance across a range of NLP tasks.

These pre-trained language models have made significant advancements in NLP and are widely used in both research and practical applications. They serve as powerful tools for chatbot development, content generation, translation, summarization, and more, due to their ability to understand and generate human-like text in a way that was previously challenging to achieve with traditional NLP techniques.

PROGRAM:

GPT-3 (Generative Pre-trained Transformer 3)

~~~
```

pip install openai
```
import openai

api_key = 'YOUR_API_KEY'
```
```python
# Set up the OpenAI API client
openai.api_key = api_key

# Define your prompt
prompt = "Translate the following English text to French: 'Hello,
how are you?'"

# Make an API call to generate text
response = openai.Completion.create(
    engine="text-davinci-002",  # You can use a different engine if
needed
    prompt=prompt,
    max_tokens=50  # Adjust the max tokens as needed
)
~~~

```
# Get the generated text from the response

 generated_text = response.choices[0].text.strip()


 print(generated_text)
```

<u>**GPT-2 (Generative Pre-trained Transformer 2)**</u>

```
pip install transformers

import torch

from transformers import GPT2LMHeadModel, GPT2Tokenizer


# Load pre-trained GPT-2 model and tokenizer

model_name = "gpt2"  # You can also use "gpt2-medium", "gpt2-large", or "gpt2-
xl"

model = GPT2LMHeadModel.from_pretrained(model_name)

tokenizer = GPT2Tokenizer.from_pretrained(model_name)


# Set the device to GPU if available, else use CPU

device = "cuda" if torch.cuda.is_available() else "cpu"

model.to(device)


# Input text

input_text = "Once upon a time,"
```

```
# Tokenize the input text
input_ids = tokenizer.encode(input_text, return_tensors="pt").to(device)


# Generate text
output = model.generate(input_ids, max_length=100, num_return_sequences=1, pad_token_id=50256)  # 50256 is the GPT-2's padding token


# Decode and print the generated text
generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
print(generated_text)
```

BERT (Bidirectional Encoder Representations from Transformers)

```
pip install transformers
import torch
from transformers import BertTokenizer, BertForSequenceClassification


# Load pre-trained BERT model and tokenizer
model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name)


# Input text for classification
text = "BERT is an amazing NLP model."
```

```
# Tokenize and prepare input
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)

# Forward pass through the model
with torch.no_grad():
    outputs = model(**inputs)

# Get predicted class probabilities
logits = outputs.logits
probabilities = torch.softmax(logits, dim=1)

# Get the predicted class index
predicted_class = torch.argmax(probabilities, dim=1).item()

# Print the result
print(f"Input text: {text}")
print(f"Predicted class: {predicted_class}")
print(f"Class probabilities: {probabilities[0].tolist()}")
```

RoBERTa (A Robustly Optimized BERT Pretraining Approach)

```
pip install transformers

import torch

from transformers import RobertaTokenizer, RobertaForSequenceClassification


# Load pre-trained RoBERTa model and tokenizer

model_name = "roberta-base"

tokenizer = RobertaTokenizer.from_pretrained(model_name)

model = RobertaForSequenceClassification.from_pretrained(model_name)


# Input text for classification

text = "RoBERTa is a robust variant of BERT."


# Tokenize and prepare input

inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)


# Forward pass through the model

with torch.no_grad():

    outputs = model(**inputs)


# Get predicted class probabilities

logits = outputs.logits
```

```
probabilities = torch.softmax(logits, dim=1)


# Get the predicted class index

predicted_class = torch.argmax(probabilities, dim=1).item()


# Print the result

print(f"Input text: {text}")

print(f"Predicted class: {predicted_class}")

print(f"Class probabilities: {probabilities[0].tolist()}")
```

XLNet (Generalized Autoregressive Pretraining for Language Understanding)

```
pip install transformers

import torch

from transformers import XLNetTokenizer, XLNetForSequenceClassification


# Load pre-trained XLNet model and tokenizer

model_name = "xlnet-base-cased"

tokenizer = XLNetTokenizer.from_pretrained(model_name)

model = XLNetForSequenceClassification.from_pretrained(model_name)


# Input text for classification

text = "XLNet is a powerful model for NLP tasks."
```

```
# Tokenize and prepare input

inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)


# Forward pass through the model

with torch.no_grad():

    outputs = model(**inputs)


# Get predicted class probabilities

logits = outputs.logits

probabilities = torch.softmax(logits, dim=1)


# Get the predicted class index

predicted_class = torch.argmax(probabilities, dim=1).item()


# Print the result

print(f"Input text: {text}")

print(f"Predicted class: {predicted_class}")

print(f"Class probabilities: {probabilities[0].tolist()}")
```

T5 (Text-to-Text Transfer Transformer):

```
pip install transformers

import torch

from transformers import XLNetTokenizer, XLNetForSequenceClassification


# Load pre-trained XLNet model and tokenizer

model_name = "xlnet-base-cased"

tokenizer = XLNetTokenizer.from_pretrained(model_name)

model = XLNetForSequenceClassification.from_pretrained(model_name)


# Input text for classification

text = "XLNet is a powerful model for NLP tasks."


# Tokenize and prepare input

inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)


# Forward pass through the model

with torch.no_grad():

    outputs = model(**inputs)


# Get predicted class probabilities

logits = outputs.logits
```

```
probabilities = torch.softmax(logits, dim=1)


# Get the predicted class index

predicted_class = torch.argmax(probabilities, dim=1).item()


# Print the result

print(f"Input text: {text}")

print(f"Predicted class: {predicted_class}")

print(f"Class probabilities: {probabilities[0].tolist()}")
```

Conclusion:

In the Phase 2 conclusion, we will summarize the key findings and insights from the advanced regression techniques. We will reiterate the impact of these techniques on improving the accuracy and robustness of CHATBOT for diabetes prediction.