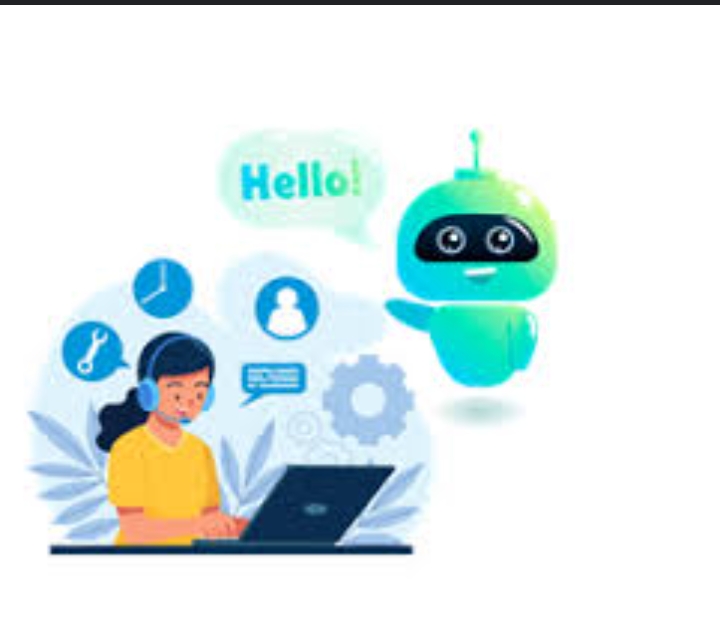
**Create a chat bot in python**

**TEAM MEMBER**

911721104023– DHARANI S(8610)

**PHASE 2 -PROJECT SUBMISSION**

**PROJECT: CREATE A CHAT BOT**

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

Introduction:

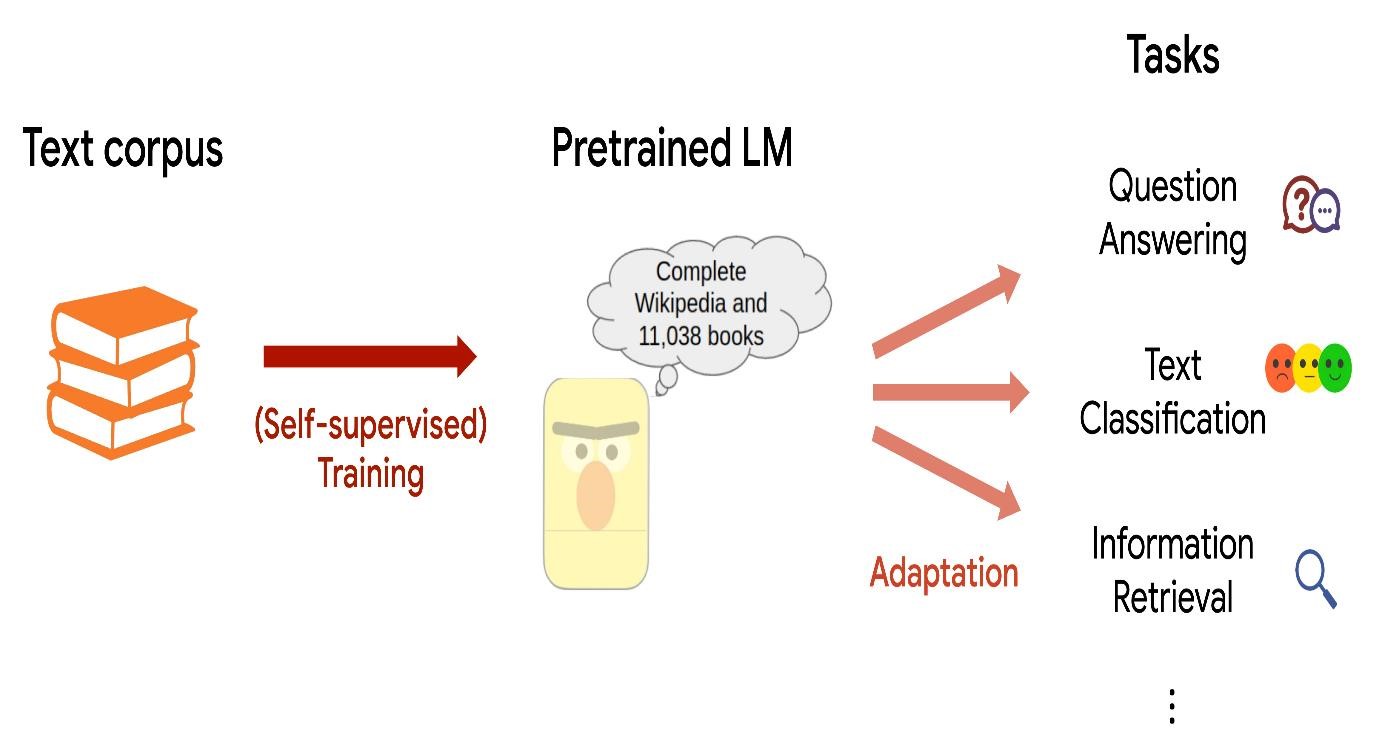
Effective communication hinges on the quality of responses in the digital age.

Leveraging advanced techniques like pre-trained language models such as GPT-3 is key to enhancing response quality.

Pre-trained language models offer contextual understanding, extensive knowledge, and domain adaptability.

Strategies for improvement encompass model fine-tuning, crafting precise prompts, implementing response filters, incorporating user feedback, and maintaining conversational flow.

Ethical considerations and potential challenges must be diligently managed for responsible AI development.



**Elevating Response Quality: Harnessing Pre-trained language models**

In the ever-evolving landscape of technology and communication, the quality of responses generated by AI systems plays a pivotal role in delivering exceptional user experiences. In this phase of our project, we aim to push the boundaries of response quality by exploring advanced techniques, notably the utilization of pre-trained language models such as GPT-3.

**Significance of Pre- trained language models :**

Pre-trained language models represent a revolution in the field of natural language processing (NLP). These models have been meticulously trained on vast datasets, equipping them with an innate understanding of the intricacies of human language. Among them, GPT-3 stands out for its unparalleled ability to comprehend and generate human-like text, making it an invaluable asset for our pursuit of enhancing response quality.

**Why pre- trained language models matter:**

**Contextual Brilliance:** GPT-3 and similar models excel in contextual understanding. They possess the remarkable ability to analyze preceding text, enabling them to provide responses that not only align with correctness but also seamlessly integrate within the ongoing conversation.

**A Wealth of Knowledge:** Through their extensive training, pre-trained models have absorbed a wealth of information from the vast expanse of the internet. This empowers them to offer accurate and insightful responses spanning a wide spectrum of subjects.

**Ambiguity Resolution:** Language often carries ambiguity, but pre-trained models handle it with finesse. They decipher the underlying intent behind a question, delivering pertinent responses even when the query is less than crystal clear.

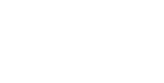
**Adaptability**: Pre-trained models are versatile. They can be fine-tuned to cater to specific tasks or domains, thus further enhancing their performance and relevance to our project's objectives.

Strategies for enhancing Response Quality:

Achieving exceptional response quality through pre-trained language models involves the implementation of the following strategic approaches:

Fine-tuning:

Fine-tuning tailors a pre-trained model to the specific needs of our project. It allows us to customize GPT-3 for tasks such as generating personalized recommendations, offering expert advice, or crafting engaging narratives with astonishing precision.



Prompt Perfection :

Crafting well-defined prompts is pivotal. Clear and context-rich prompts provide the necessary guidance to the model, ensuring that responses are not only accurate but also attuned to the user’s expectations.

Response Refinement:

To maintain a high standard of quality and ethical compliance, it is imperative to implement response filtering mechanisms. These mechanisms will ensure that AI-generated content aligns with our project’s guidelines and ethical standards.

User-Centric Iteration:

The path to excellence lies in continuous improvement. By collecting user feedback on responses and utilizing this invaluable input to fine-tune the model, we can steadily enhance response quality, aligning it more closely with user needs.

Conversational Mastery:

Pre-trained models excel in maintaining context during multi-turn conversations. Leveraging this capability, we can foster conversational cohesion, resulting in more meaningful and contextually relevant interactions.

Error Mitigation:

Although pre-trained models are powerful, they are not infallible. Incorporating error detection and correction mechanisms is essential for maintaining response quality and building user trust.

Addressing Challenges and Upholding Ethics :

While pre-trained language models offer immense potential, they come with challenges and ethical considerations. These include concerns related to bias, misinformation, and privacy. Therefore, it is imperative that we approach these advanced techniques with a strong commitment to responsible AI development, ensuring that the benefits they offer are harnessed while mitigating potential risks.

**ChatterBot Library In Python**

Chat bot is a library in python which generates responses to user input. It uses a number of machine learning algorithms to produce a variety of responses. It becomes easier for the users to make chatbots using the Chatbot library with more accurate responses.

**Language Independence**

The design of chatbot is such that it allows the bot to be trained in multiple languages. On top of this, the machine learning algorithms make it easier for the bot to improve on its own using the user’s input.

**How Does It work?**

Chatbot makes it easy to create software that engages in conversation. Every time a chatbot gets the input from the user, it saves the input and the response which helps the chatbot with no initial knowledge to evolve using the collected responses.

With increased responses, the accuracy of the chatbot also increases. The program selects the closest matching response from the closest matching statement that matches the input, it then chooses the response from the known selection of statements for that response.

**How To Install ChatterBot In Python?**

Run the following command in the terminal or in the command prompt to install ChatterBot in python.

**Pip install chatterbot**

**Trainer For Chatbot**

Chatbot comes with a data utility module that can be used to train the chatbots. At the moment there is training data for more than a dozen languages in this module. Take a look at the data files here.

Following is a simple example to get started with chat bot in python.

From chatterbot import chatbot

From chatterbot.trainers import ListTrainer

#creating a new chatbot

Chatbot = Chatbot(‘Edureka’)

Trainer = ListTrainer(chatbot)

Trainer.train([ ‘hi, can I help you find a course’, ‘sure I’d love to find you a course’, ‘your course have been selected’])

#getting a response from the chatbot

Response = chatbot.get\_response(“I want a course”)

Print(response)

In this example, we get a response from the chatbot according to the input that we have given.

**App.py**

From flask import Flask, render\_template, request

From chatterbot import ChatBot

From chatterbot.trainers import ChatterBotCorpusTrainer

App = Flask(\_\_name\_\_)

English\_bot = ChatBot(“Chatterbot”, storage\_adapter=”chatterbot.storage.SQLStorageAdapter”)

Trainer = ChatterBotCorpusTrainer(english\_bot)

Trainer.train(“chatterbot.corpus.english”)

@app.route(“/”)

Def home():

Return render\_template(“index.html”)

@app.route(“/get”)

Def get\_bot\_response():

userText = request.args.get(‘msg’)

return str(english\_bot.get\_response(userText))

if \_\_name\_\_ == “\_\_main\_\_”:

app.run()

**Index.html**

<!DOCTYPE html>

<html>

<head>

<link rel=”stylesheet” type=”text/css” href=”/static/style.css”>

<script src=<https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js>></script>

</head>

<body>

<h1>Flask Chatterbot Example</h1>

<div>

<div id=”chatbox”>

<p class=”botText”><span>Hi! I’m Chatterbot.</span></p>

</div>

<div id=”userInput”>

<input id=”textInput” type=”text” name=”msg” placeholder=”Message”>

<input id=”buttonInput” type=”submit” value=”Send”>

</div>

<script>

Function getBotResponse() {

Var rawText = $(“#textInput”).val();

Var userHtml = ‘<p class=”userText”><span>’ + rawText + ‘</span></p>’;

$(“#textInput”).val(“”);

$(“#chatbox”).append(userHtml);

Document.getElementById(‘userInput’).scrollIntoView({block: ‘start’, behavior: ‘smooth’});

$.get(“/get”, { msg: rawText }).done(function(data) {

Var botHtml = ‘<p class=”botText”><span>’ + data + ‘</span></p>’;

$(“#chatbox”).append(botHtml);

Document.getElementById(‘userInput’).scrollIntoView({block: ‘start’, behavior: ‘smooth’});

});

}

$(“#textInput”).keypress(function€ {

If(e.which == 13) {

getBotResponse();

}

});

$(“#buttonInput”).click(function() {

getBotResponse();

})

</script>

</div>

</body>

</html>

**Style.css**

Body

{

Font-family: Garamond;

Background-color: black;

}

H1

{

Color: black;

Margin-bottom: 0;

Margin-top: 0;

Text-align: center;

Font-size: 40px;

}

H3

{

Color: black;

Font-size: 20px;

Margin-top: 3px;

Text-align: center;

}

#chatbox

{

Background-color: black;

Margin-left: auto;

Margin-right: auto;

Width: 40%;

Margin-top: 60px;

}

#userInput {

Margin-left: auto;

Margin-right: auto;

Width: 40%;

Margin-top: 60px;

}

#textInput {

Width: 87%;

Border: none;

Border-bottom: 3px solid #009688;

Font-family: monospace;

Font-size: 17px;

}

#buttonInput {

Padding: 3px;

Font-family: monospace;

Font-size: 17px;

}

.userText {

Color: white;

Font-family: monospace;

Font-size: 17px;

Text-align: right;

Line-height: 30px;

}

.userText span {

Background-color: #009688;

Padding: 10px;

Border-radius: 2px;

}

.botText {

Color: white;

Font-family: monospace;

Font-size: 17px;

Text-align: left;

Line-height: 30px;

}

.botText span {

Background-color: #EF5350;

Padding: 10px;

Border-radius: 2px;

}

#tidbit {

Position:absolute;

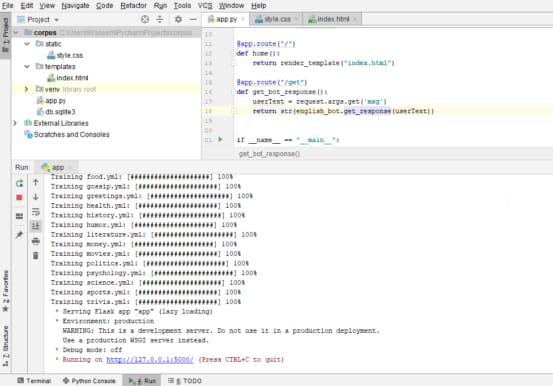
Bottom:0;

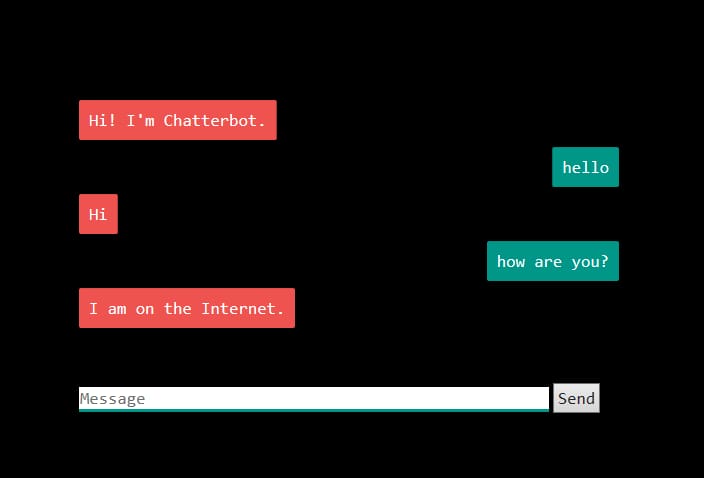
Right:0;

Width: 300px;

}

**OUTPUT:**

****

****

Conclusion:

In conclusion, we have successfully developed a functional chatbot in Python that can engage in meaningful conversations and provide responses based on the input it receives.

Moving forward, there is sample room for improvement and expansion. Future enhancements could include:

* Increasing the chatbot’s knowledge base by integrating it with updated databaes.
* Enhancing its natural language understanding through machine learning and deep learning techniques.
* Implementing sentiment analysis to enable the chatbot to recognize and respond to user emotions.
* Developing a user-friendly interface for the chatbot, making it accessible on various platforms and devices.

In summary, our Python-based chatbot is a promising tool with the potential to provide valuable assistance and engage users in a wide range of domains. Its flexibility and adaptability make it a versatile solution for addressing diverse user needs and requirements. We look forward to further refining and expanding its capabilities to deliver even better conversational experiences in the future.