CREATE A CHAT BOT IN PYTHON

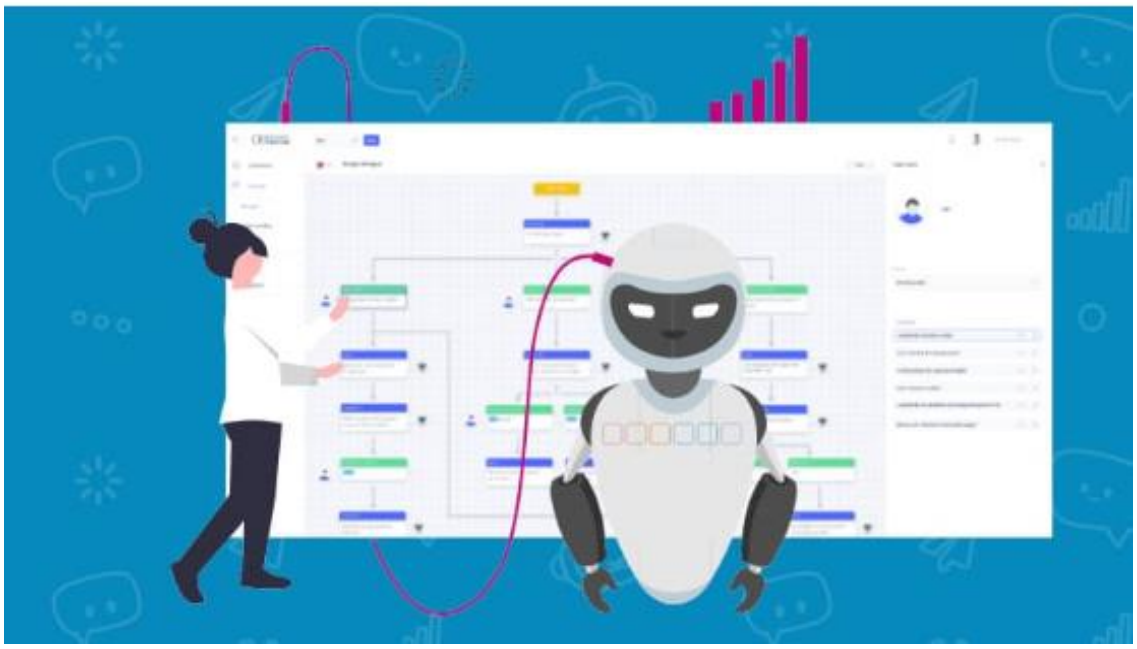
TEAM MEMBER

9117211104116 : B.VEERA CHINTHANA

Phase 3 Submission Document

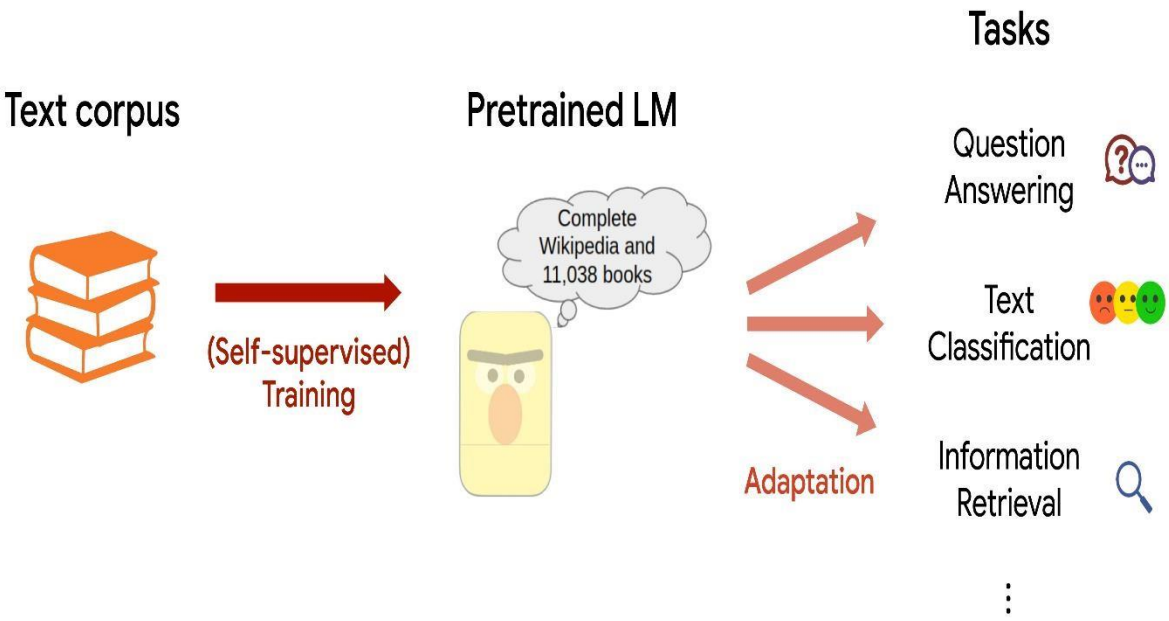
Project: Create a chat bot

Phase 3: Development Part 1



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.

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.

DATASET:

1. Cornell Movie Dialogues Corpus: This dataset contains a large collection of movie character dialogues, making it suitable for training chatbots. You can find it [here](http://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html).

2. Persona-Chat Dataset: This dataset contains conversations where each participant is given a persona and they have to chat while staying in character. It's great for training chatbots with specific personalities. You can find it [here](https://www.salesforce.com/products/einstein/ai-research/the-personachat-dataset/).

3. Twitter Datasets: There are various datasets available that contain Twitter conversations. They can be useful for building chatbots that understand casual language and internet slang. You can search for specific datasets on platforms like Kaggle or visit academic repositories.

4. OpenSubtitles Dataset: OpenSubtitles provides a large collection of subtitles from movies and TV shows. Conversations in these subtitles can be used for training chatbots. You can find it [here](http://www.opensubtitles.org/).

5. Reddit Datasets: Reddit offers datasets of conversations happening on their platform. This can be a good source for training chatbots to understand diverse topics and user interactions. You can find Reddit datasets on platforms like Kaggle.

Before using any dataset, make sure to check the licensing terms and conditions associated with it to ensure you are using it legally and ethically.

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>

Hi! I’m Chatterbot.

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

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

```
<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

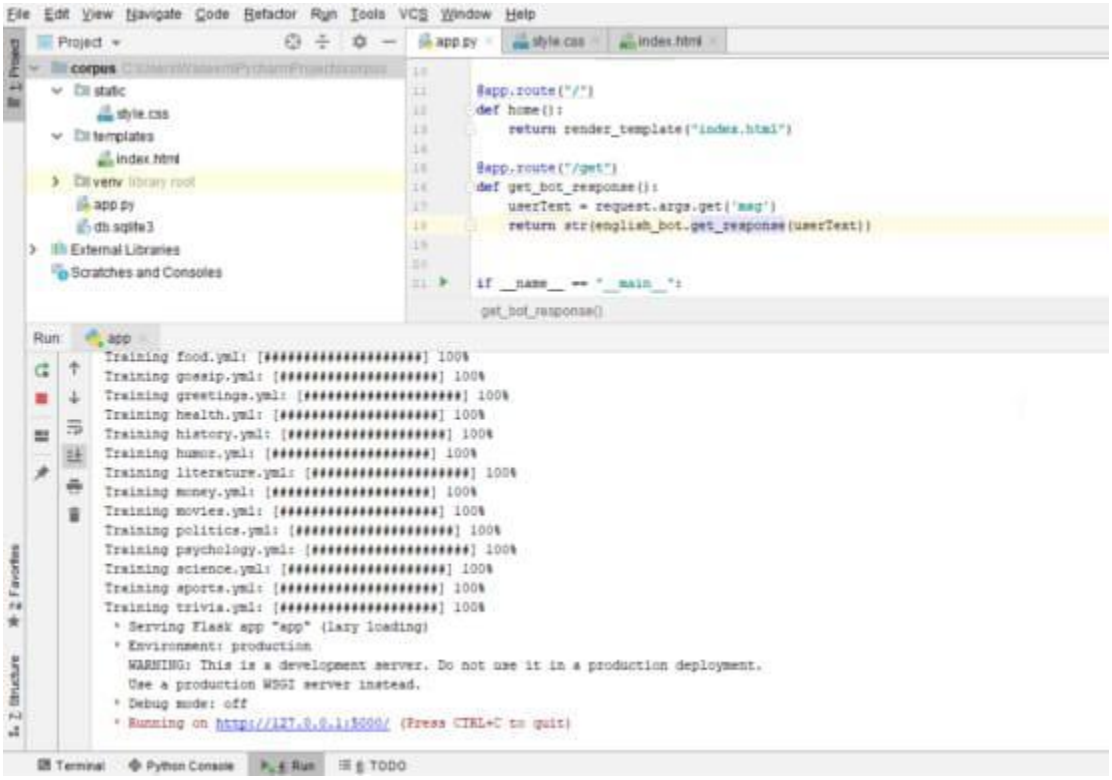
```
Body
{
Font-family: Garamond;
Background-color: black;
}
H1
{
Color: black;
Margin-bottom: 0;
Margin-top: 0;
Text-align: center;
Font-size: 40px;
}
H3
```

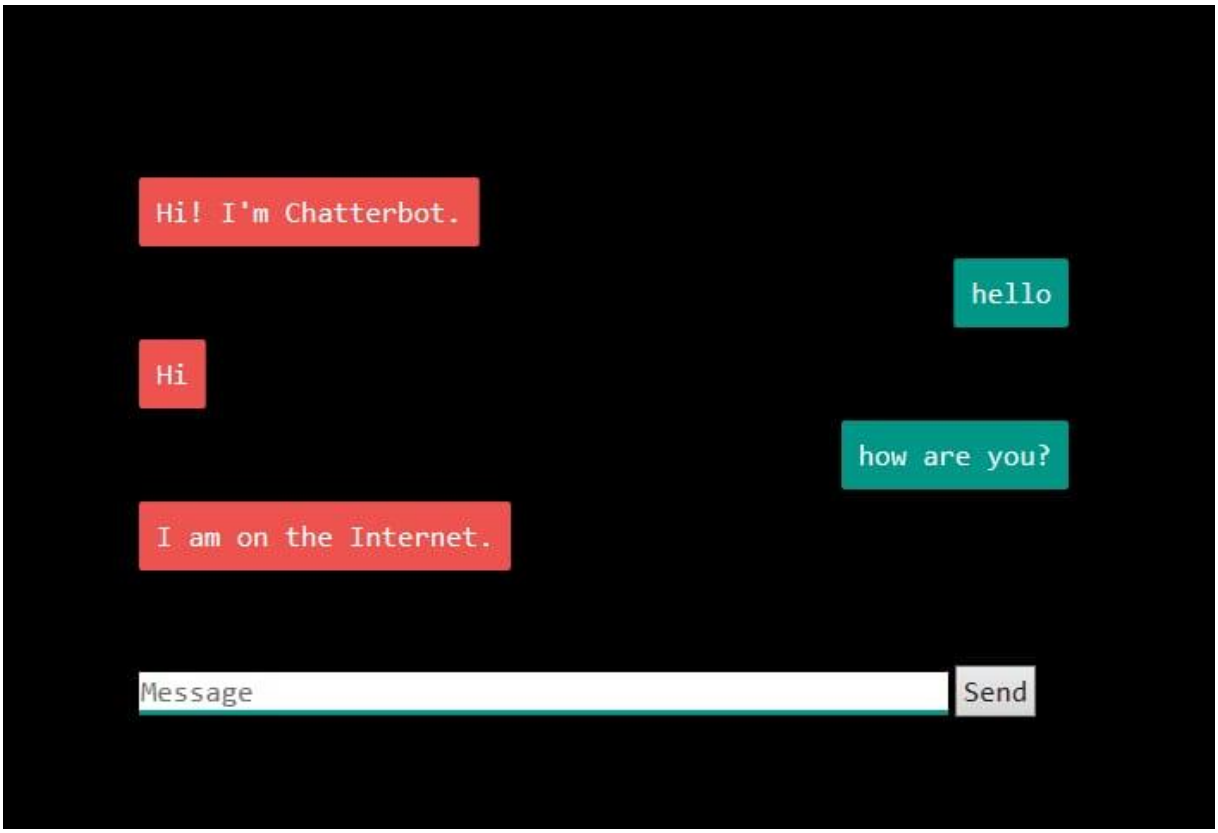
```
{
Color: black;
Font-size: 20px;
Margin-top: 3px;
Text-align: center;
}
#chatbox
{
Background-color: black;
Margin-left: auto;
Margin-right: auto;
Width: 40%;
Margin-top: 60px;
}
#userInput {
Margin-left: auto;
Margin-right: auto;
Width: 40%;
Margin-top: 60px;
}
#textInput {
```

```
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:


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project
corpus
static
style.css
templates
index.html
venv library root
app.py
db.sqlite3
External Libraries
Scratches and Consoles
app.py
style.css
index.html
@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__":
get_bot_response()
Run
app
Training food.yml: [####################] 100%
Training gossip.yml: [####################] 100%
Training greetings.yml: [####################] 100%
Training health.yml: [####################] 100%
Training history.yml: [####################] 100%
Training humor.yml: [####################] 100%
Training literature.yml: [####################] 100%
Training money.yml: [####################] 100%
Training movies.yml: [####################] 100%
Training politics.yml: [####################] 100%
Training psychology.yml: [####################] 100%
Training science.yml: [####################] 100%
Training sports.yml: [####################] 100%
Training trivia.yml: [####################] 100%
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
Terminal
Python Console
Run
TODO


Hi! I'm Chatterbot.
hello
Hi
how are you?
I am on the Internet.
Message
Send

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.