**NAAN MUDHALVAN**

**CHATBOT USING PYTHON**

**DEVELOPMENT PART 2:**

**Project Overview**

*The ChatterBot library combines language corpora, text processing, machine learning algorithms, and data storage and retrieval to allow you to build flexible chatbots.*

*You can build an industry-specific chatbot by training it with relevant data. Additionally, the chatbot will remember user responses and continue building its internal graph structure to improve the responses that it can give.*

*Begin Training Your Chatbot*

*In the previous step, you built a chatbot that you could interact with from your command line. The chatbot started from a clean slate and wasn’t very interesting to talk to.*

*In this step, you’ll train your chatbot using ListTrainer to make it a little smarter from the start. You’ll also learn about built-in trainers that come with ChatterBot, including their limitations.*

*Your chatbot doesn’t have to start from scratch, and ChatterBot provides you with a quick way to train your bot. You’ll use ChatterBot’s ListTrainer to provide some conversation samples that’ll give your chatbot more room to grow:*

*bot.py*

*from chatterbot import ChatBot*

*from chatterbot.trainers import ListTrainer*

*chatbot = ChatBot("Chatpot")*

*trainer = ListTrainer(chatbot)*

*trainer.train([*

*"Hi",*

*"Welcome, friend 🤗",*

*])*

*trainer.train([*

*"Are you a plant?",*

*"No, I'm the pot below the plant!",*

*])*

*exit\_conditions = (":q", "quit", "exit")*

*while True:*

*query = input(">")*

*if query in exit\_conditions:*

*break*

*else:*

*print(f"🪴 {chatbot.get\_response(query)}")*

*You can run more than one training session, so in lines 13 to 16, you add another statement and another reply to your chatbot’s database.*

*If you now run the interactive chatbot once again using python bot.py, you can elicit somewhat different responses from it than before:*

*> hi*

*🪴 Welcome, friend 🤗*

*> hello*

*🪴 are you a plant?*

*> me?*

*🪴 are you a plant?*

*> yes*

*🪴 hi*

*> are you a plant?*

*🪴 No, I'm the pot below the plant!*

*> cool*

*🪴 Welcome, friend 🤗*

*The conversation isn’t yet fluent enough that you’d like to go on a second date, but there’s additional context that you didn’t have before! When you train your chatbot with more data, it’ll get better at responding to user inputs.*

*The ChatterBot library comes with some corpora that you can use to train your chatbot. However, at the time of writing, there are some issues if you try to use these resources straight out of the box.*

**xplanation:**

In the above snippet of code, we have imported two classes - **ChatBot** from **chatterbot** and **ListTrainer** from **chatterbot.trainers**.

Creating and Training the Chatbot

The next step is to create a chatbot using an instance of the class "**ChatBot**" and train the bot in order to improve its performance. Training the bot ensures that it has enough knowledge, to begin with, particular replies to particular input statements.

Let us consider the following snippet of code for the same.

**File: my\_chatbot.py**

1. # creating a chatbot
2. myBot = ChatBot(
3. name = 'Sakura',
4. read\_only = True,
5. logic\_adapters = [
6. 'chatterbot.logic.MathematicalEvaluation',
7. 'chatterbot.logic.BestMatch'
8. ]
9. )

**Explanation:**

In the above snippet of code, we have defined a variable that is an instance of the class "**ChatBot**". We have included various parameters within the class. The first parameter, **'name'**, represents the name of the Python chatbot. Another parameter called **'read\_only'** accepts a Boolean value that disables (**TRUE**) or enables (**FALSE**) the ability of the bot to learn after the training. We have also included another parameter named **'logic\_adapters'** that specifies the adapters utilized to train the chatbot.

While the **'chatterbot.logic.MathematicalEvaluation'** helps the chatbot solve mathematics problems, the **`** helps it select the perfect match from the list of responses already provided.

Since we have to provide a list of responses, we can perform it by specifying the lists of strings that we can use to train the Python chatbot and find the perfect match for a certain query. Let us consider the following example of responses we can train the chatbot using Python to learn.

**File: my\_chatbot.py**

1. # training the chatbot
2. small\_convo = [
3. 'Hi there!',
4. 'Hi',
5. 'How do you do?',
6. 'How are you?',
7. 'I\'m cool.',
8. 'Always cool.',
9. 'I\'m Okay',
10. 'Glad to hear that.',
11. 'I\'m fine',
12. 'I feel awesome',
13. 'Excellent, glad to hear that.',
14. 'Not so good',
15. 'Sorry to hear that.',
16. 'What\'s your name?',
17. ' I\'m Sakura. Ask me a math question, please.'
18. ]
20. math\_convo\_1 = [
21. 'Pythagorean theorem',
22. 'a squared plus b squared equals c squared.'
23. ]
25. math\_convo\_2 = [
26. 'Law of Cosines',
27. 'c\*\*2 = a\*\*2 + b\*\*2 - 2\*a\*b\*cos(gamma)'
28. ]

**Explanation:**

In the above snippet of code, we have defined some list of responses in order to train the chatbot. We can also create and train the chatbot by simple typing an instance of "**ListTrainer**" and providing it with a list of strings as shown below:

**File: my\_chatbot.py**

1. # using the ListTrainer **class**
2. list\_trainee = ListTrainer(myBot)
3. **for** i in (small\_convo, math\_convo\_1, math\_convo\_2):
4. list\_trainee.train(i)

**Explanation:**

In the above snippet of code, we have created an instance of the **ListTrainer** class and used the **for**-loop to iterate through each item present in the lists of responses.

Now, the Python chatbot is ready to communicate.

Communicating with the Python chatbot

We can use the **get\_response()** function in order to interact with the Python chatbot. Let us consider the following execution of the program to understand it.

**Output:**

# starting a conversation

>>> print(myBot.get\_response("Hi, there!"))

Hi

>>> print(myBot.get\_response("What's your name?"))

I'm Sakura. Ask me a math question, please.

>>> print(myBot.get\_response("Do you know Pythagorean theorem"))

a squared plus b squared equals c squared.

>>> print(myBot.get\_response("Tell me the formula of law of cosines"))

c\*\*2 = a\*\*2 + b\*\*2 - 2\*a\*b\*cos(gamma)

**Explanation:**

The above execution of the program tells us that we have successfully created a chatbot in Python using the **chatterbot** library. However, it is also necessary to understand that the chatbot using Python might not know how to answer all the queries. Since its knowledge and training are still very limited, we have to provide it time and give more training data to train it further.

Training the Python Chatbot using a Corpus of Data

As we move to the final step of creating a chatbot in Python, we can utilize a present corpus of data to train the Python chatbot even further.

Let us consider the following example of training the Python chatbot with a corpus of data given by the bot itself.

**File: my\_chatbot.py**

1. from chatterbot.trainers **import** ChatterBotCorpusTrainer
2. corpus\_trainee = ChatterBotCorpusTrainer(myBot)
3. corpus\_trainee.train('chatterbot.corpus.english')

**Explanation:**

In the above snippet of code, we have imported the **ChatterBotCorpusTrainer** class from the **chatterbot.trainers** module. We created an instance of the class for the chatbot and set the training language to English.

Moreover, from the last statement, we can observe that the **ChatterBot** library provides this functionality in multiple languages. Thus, we can also specify a subset of a corpus in a language we would prefer. Hence, our chatbot in Python has been created successfully.

A complete code for the Python chatbot project is shown below.

Complete Project Code

**File: my\_chatbot.py**

1. # importing the required modules
2. from chatterbot **import** ChatBot
3. from chatterbot.trainers **import** ListTrainer
4. from chatterbot.trainers **import** ChatterBotCorpusTrainer
6. # creating a chatbot
7. myBot = ChatBot(
8. name = 'Sakura',
9. read\_only = True,
10. logic\_adapters = [
11. 'chatterbot.logic.MathematicalEvaluation',
12. 'chatterbot.logic.BestMatch'
13. ]
14. )
16. # training the chatbot
17. small\_convo = [
18. 'Hi there!',
19. 'Hi',
20. 'How do you do?',
21. 'How are you?',
22. 'I\'m cool.',
23. 'Always cool.',
24. 'I\'m Okay',
25. 'Glad to hear that.',
26. 'I\'m fine',
27. 'I feel awesome',
28. 'Excellent, glad to hear that.',
29. 'Not so good',
30. 'Sorry to hear that.',
31. 'What\'s your name?',
32. ' I\'m Sakura. Ask me a math question, please.'
33. ]
35. math\_convo\_1 = [
36. 'Pythagorean theorem',
37. 'a squared plus b squared equals c squared.'
38. ]
40. math\_convo\_2 = [
41. 'Law of Cosines',
42. 'c\*\*2 = a\*\*2 + b\*\*2 - 2\*a\*b\*cos(gamma)'
43. ]
45. # using the ListTrainer **class**
46. list\_trainee = ListTrainer(myBot)
47. **for** i in (small\_convo, math\_convo\_1, math\_convo\_2):
48. list\_trainee.train(i)
50. # using the ChatterBotCorpusTrainer **class**
51. corpus\_trainee = ChatterBotCorpusTrainer(myBot)
52. corpus\_trainee.train('chatterbot.corpus.english')