IBM NAAN MUDHALVAN

CHATBOT USING PYTHON

FINAL SUBMISSION :

How To Make A Chatbot In Python?

You may have this question in your mind, how to create a chatbot? We'll take a step by step approach and break down the process of building a Python chatbot.

To build a chatbot in Python, you have to import all the necessary packages and initialize the variables you want to use in your chatbot project. Also, remember that when working with text data, you need to perform data preprocessing on your dataset before designing an ML model.

This is where tokenizing helps with text data – it helps fragment the large text dataset into smaller, readable chunks (like words). Once that is done, you can also go for lemmatization that transforms a word into its lemma form. Then it creates a pickle file to store the python objects that are used for predicting the responses of the bot.

Another vital part of the chatbot development process is creating the training and testing datasets.

Now that we've covered the basics of chatbot development in Python, let's dive deeper into the actual process! It will help you understand how to create a chatbot.

1. Prepare the Dependencies

The first step in creating a chatbot in Python with the ChatterBot library is to install the library in your system. It is best if you create and use a new Python virtual environment for the installation. To do so, you have to write and execute this command in your Python terminal:

```
pip install chatterbot
pip install chatterbot_corpus
```

You can also install ChatterBot's latest development version directly from GitHub. For this, you will have to write and execute the following command:

pip install git+git://github.com/gunthercox/ChatterBot.git@master

If you wish to upgrade the command, you can do so as well:

```
pip install --upgrade chatterbot_corpus
pip install --upgrade chatterbot
```

Now that your setup is ready, we can move on to the next step to create chatbot using python.

2. Import Classes

Importing classes is the second step in the Python chatbot creation process. All you need to do is import two classes – ChatBot from chatterbot and ListTrainer from chatterbot.trainers. To do this, you can execute the following command:

```
from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
```

3. Create and Train the Chatbot

This is the third step on creating chatbot in python. The chatbot you are creating will be an instance of the class "ChatBot." After creating a new ChatterBot

instance, you can train the bot to improve its performance. Training ensures that the bot has enough knowledge to get started with specific responses to specific inputs. You have to execute the following command now:

```
my_bot = ChatBot(name='PyBot', read_only=True,
                 logic_adapters=
['chatterbot.logic.MathematicalEvaluation',
                                 'chatterbot.logic.BestMatch'])
```

Here, the argument (that corresponds to the parameter name) represents the name of your Python chatbot. If you wish to disable the bot's ability to learn after the training, you can include the "read_only=True" command. The command "logic_adapters" denotes the list of adapters used to train the chatbot.

While the "chatterbot.logic.MathematicalEvaluation" helps the bot to solve math problems, the "chatterbot.logic.BestMatch" helps it to choose the best match from the list of responses already provided.

Since you have to provide a list of responses, you can do it by specifying the lists of strings that can be later used to train your Python chatbot, and find the best match for each query. Here's an example of responses you can train your chatbot using python to learn:

```
small_talk = ['hi there!',
              'hi!',
              'how do you do?',
              'how are you?',
              'i\'m cool.',
              'fine, you?',
              'always cool.',
              'i\'m ok',
              'glad to hear that.',
              'i\'m fine',
              'glad to hear that.',
              'i feel awesome',
              'excellent, glad to hear that.',
              'not so good',
              'sorry to hear that.',
              'what\'s your name?',
              'i\'m pybot. ask me a math question, please.']

math_talk_1 = ['pythagorean theorem',
               'a squared plus b squared equals c squared.']

math_talk_2 = ['law of cosines',
               'c**2 = a**2 + b**2 - 2 * a * b * cos(gamma)']
```

You can also create and train the bot by writing an instance of “ListTrainer” and supplying it with a list of strings like so:

```
list_trainer = ListTrainer(my_bot)

for item in (small_talk, math_talk_1, math_talk_2):
    list_trainer.train(item)
```

4. Communicate with the Python Chatbot

To interact with your Python chatbot, you can use the .get_response() function. This is how it should look while communicating:

```
>>> print(my_bot.get_response("hi"))
how do you do?


>>> print(my_bot.get_response("i feel awesome today"))
excellent, glad to hear that.


 >>> print(my_bot.get_response("what's your name?"))
i'm pybot. ask me a math question, please.


>>> print(my_bot.get_response("show me the pythagorean
theorem"))
a squared plus b squared equals c squared.


>>> print(my_bot.get_response("do you know the law of
cosines?"))
c**2 = a**2 + b**2 - 2 * a * b * cos(gamma)
```

However, it is essential to understand that the chatbot using python might not know how to answer all your questions. Since its knowledge and training is still very limited, you have to give it time and provide more training data to train it further.

5. Train your Python Chatbot with a Corpus of Data

In this last step of how to make a chatbot in Python, for training your python chatbot even further, you can use an existing corpus of data. Here's an example of how to train your Python chatbot with a corpus of data provided by the bot itself:

```
from chatterbot.trainers import ChatterBotCorpusTrainer


corpus_trainer = ChatterBotCorpusTrainer(my_bot)
corpus_trainer.train('chatterbot.corpus.english')
```

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:

```
1# bot.py
2
3from chatterbot import ChatBot
4from chatterbot.trainers import ListTrainer
5
6chatbot = ChatBot("Chatpot")
7
8trainer = ListTrainer(chatbot)
9trainer.train([
10    "Hi",
11    "Welcome, friend ",
12])
13trainer.train([
14    "Are you a plant?",
15    "No, I'm the pot below the plant!",
16])
17
18exit_conditions = (":q", "quit", "exit")
19while True:
20    query = input("> ")
21    if query in exit_conditions:
22        break
23    else:
24        print(f" {chatbot.get_response(query)}")
```

In line 4, you import ListTrainer, to which you pass your chatbot on line 8 to create trainer.

In lines 9 to 12, you set up the first training round, where you pass a list of two strings to trainer.train(). Using .train() injects entries into your database to build upon the graph structure that ChatterBot uses to choose possible replies.

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.

While the provided corpora might be enough for you, in this tutorial you'll skip them entirely and instead learn how to adapt your own conversational input data for training with ChatterBot's ListTrainer.

To train your chatbot to respond to industry-relevant questions, you'll probably need to work with custom data, for example from existing support requests or chat logs from your company.

Moving forward, you'll work through the steps of converting chat data from a WhatsApp conversation into a format that you can use to train your chatbot. If your own resource is WhatsApp conversation data, then you can use these steps directly. If your data comes from elsewhere, then you can adapt the steps to fit your specific text format.

To start off, you'll learn how to export data from a WhatsApp chat conversation.

```
import spacy

from chatterbot import ChatBot

from chatterbot.trainers import ChatterBotCorpusTrainer


# Load the spaCy English model
```

```
nlp = spacy.load('en_core_web_sm')


# Create a new chatbot

chatbot = ChatBot('MyAdvancedBot')


# Create a new trainer for the chatbot

trainer = ChatterBotCorpusTrainer(chatbot)


# Train the chatbot on English language data

trainer.train('chatterbot.corpus.english')


# Custom function using spaCy for more advanced processing

def get_response(input_text):

    doc = nlp(input_text)

    # Perform custom processing or analysis here

    # ...


# Get a response using the custom function

response = get_response('Tell me about artificial intelligence')

print(response)
```

What Is A Chatbot?

A chatbot also known as a chatterbot, bot, artificial agent, etc is basically software program driven by artificial intelligence which serves the purpose of making a conversation with the user by texts or by speech. Famous examples include Siri, Alexa, etc.

These chatbots are inclined towards performing a specific task for the user. Chatbots often perform tasks like making a transaction, booking a hotel, form submissions, etc. The possibilities with a chatbot are endless with the technological advancements in the domain of artificial intelligence.

Almost 30 percent of the tasks are performed by the chatbots in any company. Companies employ these chatbots for services like customer support, to deliver information, etc. Although the chatbots have come so far down the line, the journey started from a very basic performance. Let's take a look at the evolution of chatbots over the last few decades.

Evolution Of Chatbots

It started in 1966 when Joseph Weizenbaum made a natural language conversational program that featured a dialog between a user and a computer program. With this great breakthrough came the new age chatbot technology that has taken an enormous leap throughout the decades.

Limitations With A Chatbot

With increasing advancements, there also comes a point where it becomes fairly difficult to work with the chatbots. Following are a few limitations we face with the chatbots.

Domain Knowledge – Since true artificial intelligence is still out of reach, it becomes difficult for any chatbot to completely fathom the conversational boundaries when it comes to conversing with a human.

Personality – Not being able to respond correctly and fairly poor comprehension skills has been more than frequent errors of any chatbot, adding a personality to a chatbot is still a benchmark that seems far far away. But we are more than hopeful with the existing innovations and progress-driven approaches.

How Does It Work?

We can define the chatbots into two categories, following are the two categories of chatbots:

Rule-Based Approach – In this approach, a bot is trained according to rules. Based on this a bot can answer simple queries but sometimes fails to answer complex queries.

Self-Learning Approach – These bots follow the machine learning approach which is rather more efficient and is further divided into two more categories.

Retrieval-Based Models – In this approach, the bot retrieves the best response from a list of responses according to the user input.

Generative Models – These models often come up with answers than searching from a set of answers which makes them intelligent bots as well.

Let us try to make a chatbot from scratch using the chatterbot library in python.

Subscribe

How To Make A Chatbot In Python?

Last updated on Apr 13,2023128.3K Views

Mohammad Waseem

Mohammad Waseem

image not found!

image not found!

How To Make A Chatbot In Python?

edureka.co

image not found!

Whatsapp

image not found!

Linkedin

image not found!

Twitter

image not found!

Facebook

image not found!

Reddit

Copy Link

image not found!

Nowadays almost 30 percent of the tasks are fulfilled by chatbots. Companies use the chatbots to provide services like customer support, generating information, etc. With examples like Siri, Alexa it becomes clear how a chatbot can make a difference in our daily lives. In this article, we will learn

how to make a chatbot in python using the ChatterBot library which implements various machine learning algorithms to generate responses. Following are the topics discussed in this blog:

What Is A Chatbot?

Evolution Of Chatbots

Limitations With A Chatbot

How Does It Work?

Chatterbot Library In Python

How Does It work?

Trainer For Chatbot

Use Case – Flask Chatbot

What Is A Chatbot?

A chatbot also known as a chatterbot, bot, artificial agent, etc is basically software program driven by artificial intelligence which serves the purpose of making a conversation with the user by texts or by speech. Famous examples include Siri, Alexa, etc.

These chatbots are inclined towards performing a specific task for the user. Chatbots often perform tasks like making a transaction, booking a hotel, form submissions, etc. The possibilities with a chatbot are endless with the technological advancements in the domain of artificial intelligence.

Almost 30 percent of the tasks are performed by the chatbots in any company. Companies employ these chatbots for services like customer support, to deliver information, etc. Although the chatbots have come so far down the line, the journey started from a very basic performance. Let’s take a look at the evolution of chatbots over the last few decades.

Evolution Of Chatbots

It started in 1966 when Joseph Weizenbaum made a natural language conversational program that featured a dialog between a user and a computer program. With this great breakthrough came the new age chatbot technology that has taken an enormous leap throughout the decades.

Traditional Bots	Current Bots	Future Bots
System Driven	Driven by back-and-forth communication	Communication at multiple-levels
Automation based	The automation is at the task level	Automation at the service level
Minimal Functionality	Maintains system context	Ability to maintain task, system and people context
Maintained only system context	Maintains task context as well	Introduction to master bots and eventually a bot OS as well.

Limitations With A Chatbot

With increasing advancements, there also comes a point where it becomes fairly difficult to work with the chatbots. Following are a few limitations we face with the chatbots.

Domain Knowledge – Since true artificial intelligence is still out of reach, it becomes difficult for any chatbot to completely fathom the conversational boundaries when it comes to conversing with a human.

Personality – Not being able to respond correctly and fairly poor comprehension skills has been more than frequent errors of any chatbot, adding a personality to a chatbot is still a benchmark that seems far far away. But we are more than hopeful with the existing innovations and progress-driven approaches.

How Does It Work?

We can define the chatbots into two categories, following are the two categories of chatbots:

Rule-Based Approach – In this approach, a bot is trained according to rules. Based on this a bot can answer simple queries but sometimes fails to answer complex queries.

Self-Learning Approach – These bots follow the machine learning approach which is rather more efficient and is further divided into two more categories.

Retrieval-Based Models – In this approach, the bot retrieves the best response from a list of responses according to the user input.

Generative Models – These models often come up with answers than searching from a set of answers which makes them intelligent bots as well.

Let us try to make a chatbot from scratch using the chatterbot library in python.

ChatterBot Library In Python

ChatterBot 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 ChatterBot library with more accurate responses.

Language Independence

The design of ChatterBot 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?

ChatterBot 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

Chatterbot 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 ChatterBot 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. Let us try to build a rather complex flask-chatbot using the chatterbot-corpus to generate a response in a flask application.

```
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(e) {
if(e.which == 13) {
getBotResponse();
}
});
$("#buttonInput").click(function() {
getBotResponse();
})
</script>
</div>
</body>
</html>

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;

}
```

The chatbot will look something like this, which will have a textbox where we can give the user input, and the bot will generate a response for that statement.

In this article, we have learned how to make a chatbot in python using the ChatterBot library using the flask framework. With new-age technological advancements in the artificial intelligence and machine learning domain, we are only so far away from creating the best version of the chatbot available to mankind. Don't be in the sidelines when that happens, to master your skills enroll in Edureka's Python certification program and become a leader.

Also, If you wish to learn more about ChatGPT, Edureka is offering a great and informative ChatGPT Certification Training Course which will help to upskill your knowledge in the IT sector.

Conclusion

What we've illustrated here is just one among the many ways of **how to make a chatbot in Python.** You can also use NLTK, another resourceful Python library to create a Python chatbot. And although what you learned here is a very basic **chatbot in Python** having hardly any cognitive skills, it should be enough to help you understand the anatomy of chatbots.

Once you understand the design of a **chatbot using python** fully well, you can experiment with it using different tools and commands to make it even smarter.