Chapter 1: Introduction

**Introduction To Chatbot**

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:

**1.1**

* **What is A Chatbot?**
* **Evolution Of Chatbots**
* **Limitations With A Chatbot**
* **How Does It Work?**

**1.2**

* **Chatterbot Library In Python**
* **How Does It Work?**

**1.3**

* **Trainer For Chatbot**

**1.4**

* **Use Case – Flask Chatbot**

**1.1 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.

**1.1.1**

**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. |

**1.1.2**

**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.

**1.1.3**

**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.**

**1.2**

**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.

**1.2.1**

**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.

**1.2.2**

**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.

**1.2.3**

**How To Install ChatterBot In Python?**

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

1. ***pip install chatterbot***
2. ***pip install chatterbot-corpus***
3. ***pip install flask***

**1.3**

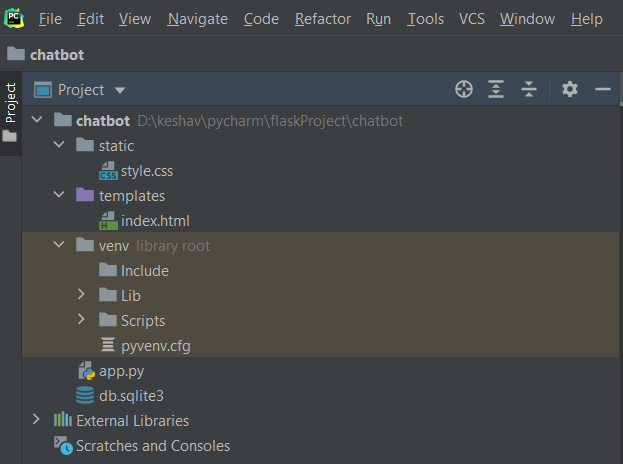
**Trainer For Chatbot**

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.

**1.4**

**Use Case – Flask ChatterBot**

After we are done setting up the flask app, we need to add two more directories static and templates for HTML and CSS files. Following is the code for the flask ChatterBot app.



**Chapter 2: Implementation**

**2.1**

**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()**

**2.2**

**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/3.5.1/jquery.min.js"></script>  
</head>  
<body>  
<h1>Mini Chatbot using Python</h1>  
<div>  
<div id="chatbox">  
<p class="botText"><span>Hi! I'm KIAT's Chatbot.</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>**

**2.3**

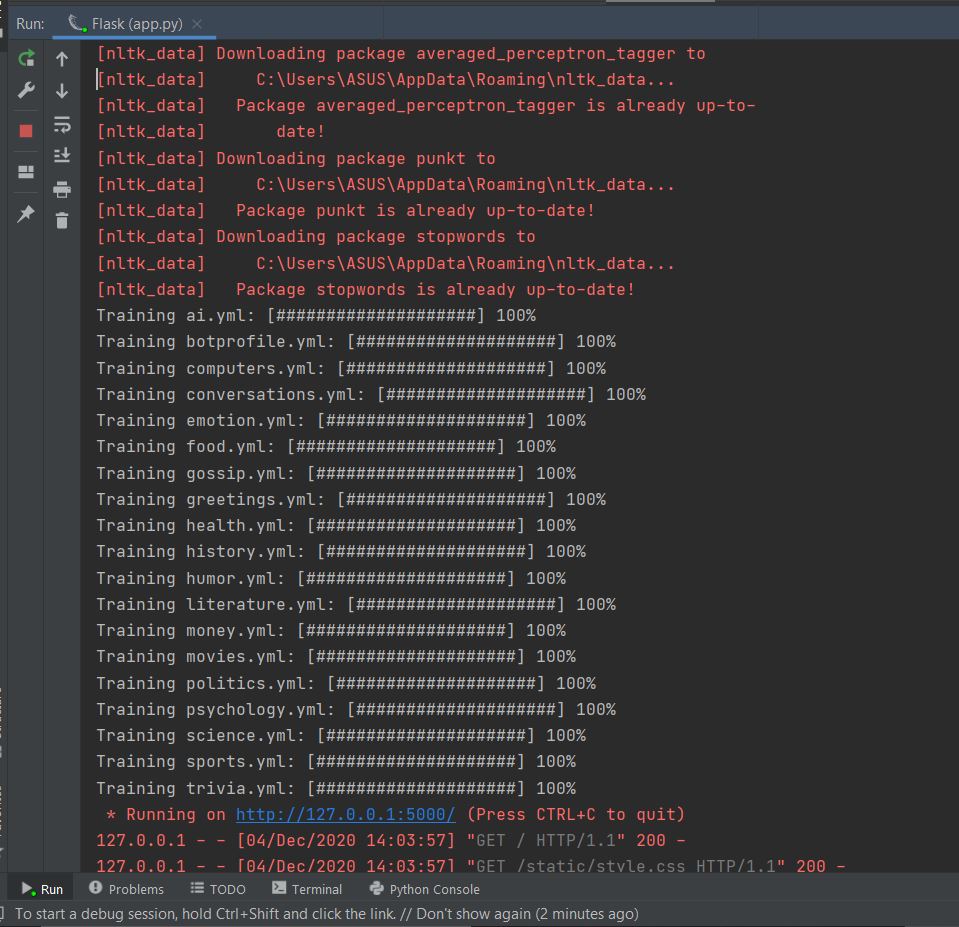
**Style.css**

**body  
{  
font-family: Garamond;  
background-color: black;  
}  
h1  
{  
color: purple;  
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;  
}**

**Chapter 3: Testing**

**3.1**

**Terminal:**

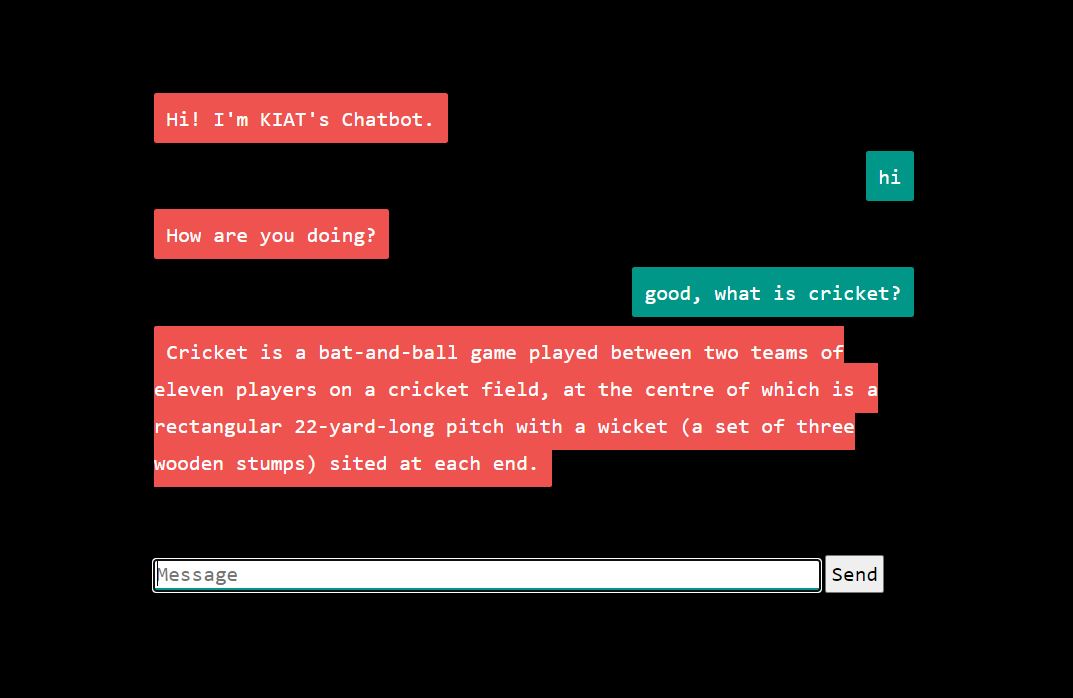
****

Go to the address shown in the output, and you will get the app with the chatbot in the browser.

**Chapter 4: Results And Discussion**

**4.1**

**Output:**

****

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.

**Conclusions**

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 .