PROJECT : CREATE A CHATBOT IN PYTHON

INTRODUCTION:

A chatbot is computer language that can understand human language and converse with a user via a website or a messaging app. Chatpot can handle tasks online – from answering simple questions and scheduling calls to gathering customers feedback

8ABSTRACT :

To built a artificial intelligence chatbots through python, you will require an AIML package. First we need to create a standard setup file without any pattern and load aiml b in the kernel . Add a response pattern that would makes dialogue interesting

Top of Form

Bottom of Form

In summary, understanding NLP and how it is implemented in Python is crucial in your journey to creating a Python AI chatbot. It equips you with the tools to ensure that your chatbot can understand and respond to your users in a way that is both efficient and human-like.

## Types of Chatbots



Before we dive into the technicalities of building your very own Python[AI chatbot](https://blog.hubspot.com/marketing/best-ai-chatbot?hubs_content=blog.hubspot.com/website/python-ai-chat-bot&hubs_content-cta=%20AI%20chatbot), it‘s essential to understand the different types of chatbots that exist. This understanding will allow you to choose the kind of chatbot that best suits your needs. Let’s take a closer look at the three primary types of chatbots: rule-based, self-learning, and hybrid.

* **Rule-Based Chatbots:** These chatbots operate based on pre-determined rules on which they are initially programmed. They are best for scenarios that require simple query-response conversations. Their downside is they can't handle complex queries because their intelligence is limited to their programmed rules.
* **Self-Learning Chatbots:** Powered by Machine Learning and Artificial Intelligence, these chatbots learn from their mistakes and the inputs they receive. The more data they are exposed to, the better their responses become. These chatbots are suited for complex tasks but their implementation is more challenging.
* **Hybrid Chatbots:**As the name suggests, these chatbots combine the best of both worlds. They operate on pre-defined rules for simple queries and use machine learning capabilities for complex queries. Hybrid chatbots offer flexibility and can adapt to a wide array of situations, making them a popular choice.

Let‘s consider a practical scenario. Suppose you run an e-commerce website. A rule-based chatbot might suffice if you want to answer FAQs. But, if you want the chatbot to recommend products based on customers’ past purchases or preferences, a self-learning or hybrid chatbot would be more suitable.

Understanding the types of chatbots and their uses helps you determine the best fit for your needs. The choice ultimately depends on your chatbot’s purpose, the complexity of tasks it needs to perform, and the resources at your disposal.

### **Building Your First Python AI Chatbot**

Now that we have a solid understanding of NLP and the different types of chatbots, it‘s time to get our hands dirty. In this section, we’ll walk you through a simple step-by-step guide to creating your first Python AI chatbot. We'll be using the ChatterBot library in Python, which makes building [AI-based](https://blog.hubspot.com/website/ai-website-builder?hubs_content=blog.hubspot.com/website/python-ai-chat-bot&hubs_content-cta=AI-based) chatbots a breeze.

### **Step 1: Install Required Libraries**

Install the ChatterBot library using pip to get started on your chatbot journey.

Python

Pip install chatterbot

### **Step 2: Import Necessary Libraries**

Import ChatterBot and its corpus trainer to set up and train the chatbot.

Python

From chatterpot Import chatbot

From chatterpot. Trainer Import ChatterBot

### **Step 3: Create and Name Your Chatbot**

Create your chatbot instance and name it something memorable.

Python

Chatbot =chatpot (my chatbot )

### **Step 4: Train Your Chatbot with a Predefined Corpus**

Use the ChatterBotCorpusTrainer to train your chatbot using an English language corpus.

Trainer = chatbot corpus trainer

Trainer.train(‘chatterpot. Corpus )

### **Step 5: Test Your Chatbot**

Interact with your chatbot by requesting a response to a greeting.

Response = chatpot .get response (chatterpot)

Print (response)

### **Step 6: Train Your Chatbot with Custom Data**

Make your chatbot more specific by training it with a list of your custom responses.

From chatterbot.trainer import chatterpot

Trainer = list trainer (chatbot)

“How are you”? ,

“I am good”,

“That is good to hear .”

“You’re welcome”

] )

### **Step 7: Integrate Your Chatbot into a Web Application**

Use Flask to create a web interface for your chatbot, allowing users to interact with it through a browser.

From flash import flash , render\_ chatbot

App = ( --flash --)

@app.route = (“/”)

Return render template (‘index chatbot’)

App run ()

By following these steps, you'll have a functional Python AI chatbot that you can integrate into a web application. This lays down the foundation for more complex and customized chatbots, where your imagination is the limit. Experiment with different training sets, algorithms, and integrations to create a chatbot that fits your unique needs and demands