Creating a Chatbot in python

How To Make A Chatbot In Python?

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.

Creating a chatbot in Python involves several steps. Here's a high-level overview of the process:

Define the Purpose: *Determine the purpose and functionality of your chatbot. What will it do? What problems will it solve?*

Choose a Framework: *Select a framework or library for building your chatbot. Popular choices include:*

NLTK*: Natural Language Toolkit*

spaCy*: Industrial-strength natural language processing*

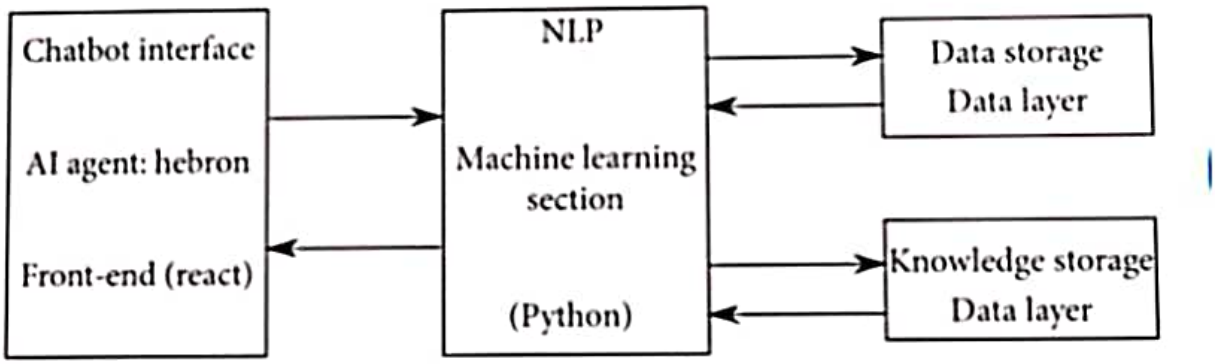
ChatterBot: *A machine learning-based chatbot framework*

Dialogflow: *A cloud-based chatbot development platform*

Collect and Prepare Data: *If your chatbot needs to understand and respond to specific topics, you'll need to collect and prepare training data. This might involve creating a dataset of questions and responses.*

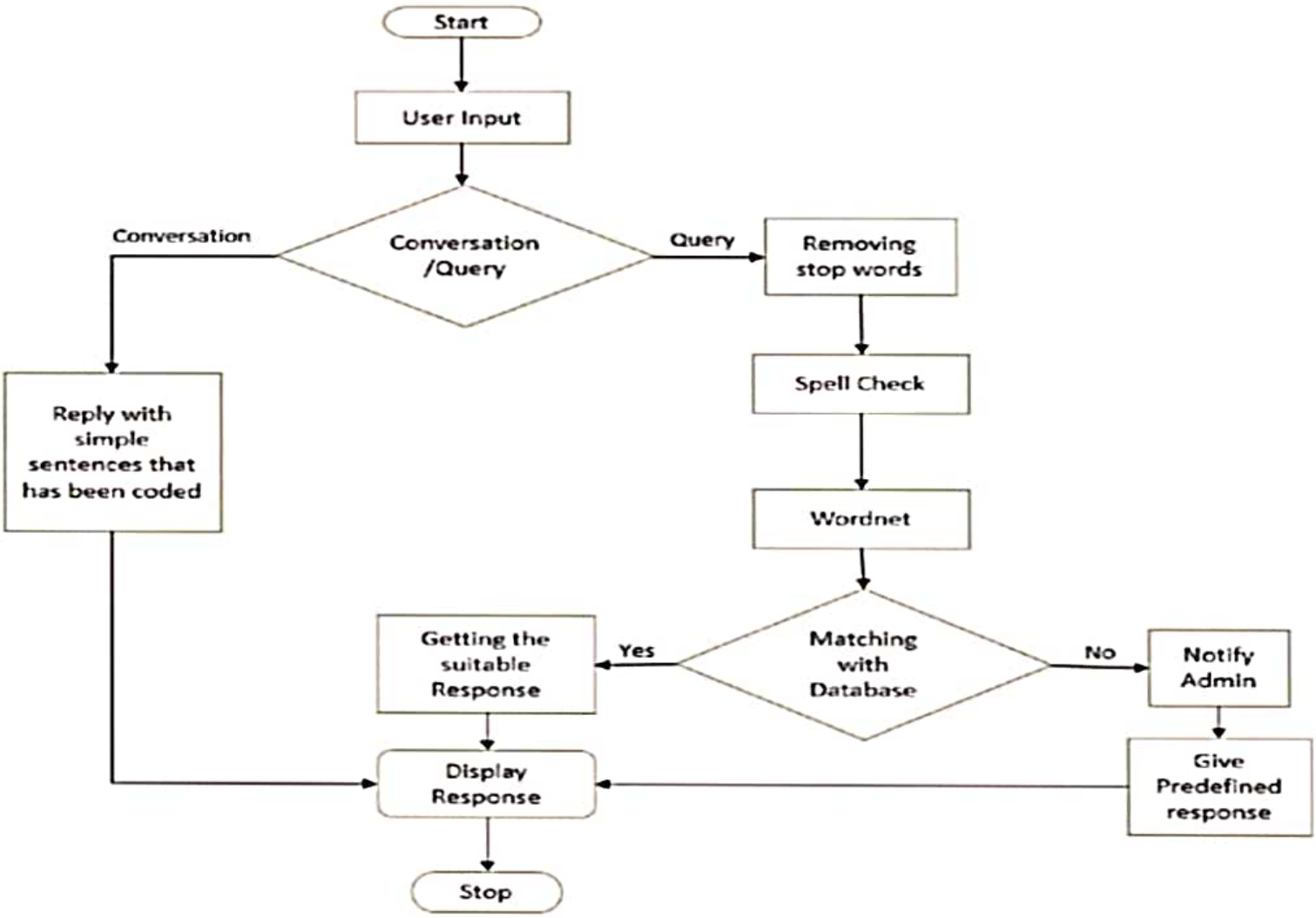
Design the Conversation Flow: *Plan how the conversation will flow. Define what questions or statements your chatbot should respond to and how it should respond. This often involves creating a decision tree or flowchart.*

Block diagram:



Block diagram for the University Chatbot System

Flow chart:



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
```

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'])
```

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 pythYou can also

```
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)']
```

create and train the bot by writing an instance of "ListTrainer" and supplying it with a list of strings like so:on to learn:

```
list_trainer = ListTrainer(my_bot)

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

Now, your Python chatbot is ready to communicate.

You: What is your name?

Chatbot: sofia

You: hii sofia

Chatbot: how r you?

You:I'm good

.....

HOW DOES CHATBOT IN PYTHON WORKS:

- *RULE BASED APPROACH*
- *SELF LEARNING APPROACH*

ADVANTAGES OF A CHATBOT IN PYTHON:

PYTHON CHATBOT IS AN ARTIFICIAL INTELLIGENCEBASED PROGRAM THAT MIMICS HUMAN SPEECH. PYTHON IS AN EFFECTIVE AND SIMPLE PROGRAMMING LANGUAGE FOR BUILDING CHATBOTS AND FRAMEWORKS LIKE CHATTERBOT. 3

Conclusion:

Chatbot Python development may be rewarding and exciting. Using the ChatterBot library and the right strategy, you can create chatbots for consumers that are natural and relevant. By mastering the power of Python's chatbot-building capabilities, it is possible to realize the full potential of this artificial intelligence technology and enhance user experiences across a variety of domains. Simplilearn's postgraduate program in Machine Learning and AI, in collaboration with Purdue University and IBM will help you learn in-demand skills such as deep learning, reinforcement learning, NLP, computer vision, generative AI, explainable AI, and many more.

Team members:

Jabez

Antony Raja

Manimaran

Raja

Abiman