TITLE: Develop an elementary chatbot for any suitable customer interaction application

AIM: To create Chatbot for customer interaction

OBJECTIVES: Based on above main aim following are the objectives

1. To study theory of Chatbot
2. To create Chatbot

Theory:

Chatbots:

A chatbot can be defined as a computer program that converses with users over the internet as any human would with natural language and sentiments. It is a program or piece of software that automatically responds to messages sent over a website chatbox, email, social media messaging app, or text.

For a deeper understanding of Chatbot, we can define it as a computer program that impersonates human conversations in its natural format, which may include text (since the advent of bots) or spoken language using artificial intelligence (AI) techniques such as Natural Language Processing (NLP) and audio analysis. One of the primary aspects of an AI-based bot is that it is dynamic.

A chatbot is a smart application that reduces human work and helps an organization to solve basic queries of the customer. Today most of the companies, business from different sector makes use of chatbot in a different way to reply their customer as fast as possible. chatbots also help in increasing traffic of site which is top reason of business to use chatbots.

Chatbot asks for basic information of customers like name, email address, and the query. If a query is simple like product fault, booking mistake, need some information then without any human connection it can solve it automatically and If some problem is high then It passes the details to the human head and helps customer to connect with organization manager easily. And most of the customers like to deal and talk with a chatbot.

How do the Chatbots function?

The main technology that lies behind chatbots is NLP and Machine Learning.

When a question is presented to a chatbot, a series or complex algorithms process the received input, understand what the user is asking, and based on that, determines the answer suitable to the question.

Chatbots have to rely on the ability of the algorithms to detect the complexity of both text and spoken words. Some chatbots perform very well to the point it becomes difficult to differentiate whether the user is a machine or a human.

However, handling complex conversations is a huge challenge; where there is a usage of various figures of speech, it may be difficult for machines to understand.

Why we need Chatbots?

Cost and Time Effective: Humans cannot be active on-site 24/7 but chatbots can and the replying power of chatbots is much fast than humans.

Cheap Development cost: with the advancement in technology many tools are developed that help easy development and integration of chatbots with little investment.

Human Resource: Today Chatbots can also talk with text o speech technology so it gives the feel as a human is talking on another side.

Business Branding: Businesses are changing with technology and chatbot is one out of them. Chatbot also helps in advertising, branding of organization product and services and give daily updates to users.

Types of Chatbots:

1. **Rule-based chatbots:** Chatbots follow a set of established rules or flows to respond to questions posted by a user. All your simple applications contain rule-based chatbots, which respond to queries based on the rules they are trained on. For instance, a weather application, where you ask for weather forecast and it fetches the data from different sources and responds with the information.

   Rule-based chatbots may not be able to hold complex conversations. It can only accomplish the tasks it is programmed to perform unless more improvements are made by the developer.

2. **Self-learning chatbots:** Self-learning bots are highly efficient because they are capable to grab and identify the user's intent on their own. They are build using advanced tools and techniques of Machine Learning, Deep Learning, and NLP. Self-learning bots are further divided into 2 subcategories.

   a. **Retrieval-based chatbots**:- Retrieval-based it is somewhat the same as Rule- based where predefined input patterns and responses are embedded.

b. **Generative-Based chatbots**:- It is based on the same phenomenon as Machine Translation build using sequence 2 sequences neural network.

Most of the organization uses self-learning chatbot along with embedding some rules like Hybrid version of both methods which makes chatbot powerful to handle each situation during a conversation with a customer.

Chatbot Development Platforms:

IBM Watson: Watson is one of the most preferred platforms when it comes to building AI chatbots. The advantage of Watson is its capability to serve different verticals and manage complex interactions with ease.

Microsoft Azure Bot Service: The Azure bot service provides the developer with SDK and portal, along with a bot connector service that will allow the developer to connect to any social media platform. The SDK also helps with debugging your bot and provides a large selection of sample bots that can be used as building blocks for your bot. This Cloud-based service is accessible from almost anywhere and provides multiple language support.

QnA Maker: This is another bot from Microsoft, which is exactly as the name suggests. It can be of great help to any business that is asked frequent questions from their customers regarding their products. QnA Maker allows you to develop and train your bots for answering simple questions, based on your FAQ URLs, any structured documents, and manuals for the product within a matter of minutes.

Chatbot Deployment Platforms:

Once chatbots are developed, they need to be deployed to a deployment platform. You will have to choose a deployment platform based on your customer base. However, the use of chatbots revolves mostly around social media platforms or virtual assistant features in various devices. Let us look at some of the emerging bot platform ecosystems.

Facebook Messenger: With over 1 billion users, there is no denying that Facebook has a wide reach around the world. For developers who are developing bots, this is a great platform to reach out to a bigger audience. Facebook has been investing in bot development and has provided tools for users to create bots for their specific needs without writing a single line of code. Fast food joints like Burger King has leveraged the use of bots to serve their customers by taking their order

via Facebook. Many businesses have used Facebook to their advantage and improved ways of serving their customer base.

Skype for Business: This is another popular instant messaging platform utilized by many businesses around the world for their internal or external communication. Bots like Skyscanner allow you to make travel arrangements right in your Skype window. In addition, it helps you to find the most affordable travel options. Bots like Bing Image Preview and Getty Images allow you to search for images right from your Skype search bar.

Kik: It is an instant messaging platform, used for internal communication in businesses. One of the most popular bots on this platform is The Weather Channel. It forecasts the weather for you and lets you know if there is going to be any change in the weather. This is great for traveling professionals as they can plan their schedules accordingly.

There are other messaging platforms with interesting bots that have been used to make business operations smoother and easier.

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.

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

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:

```python
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:

```python
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.

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')
```

The good thing is that ChatterBot offers this functionality in many different languages. So, you can also specify a subset of a corpus in a language you would prefer.

Conclusion: Thus we have successfully implemented elementary Chatbot for Customer interaction application.

```
    print('..................................')
    print('..................................')
    input()


greet('TE-Chatbot', '2022')  # change it as you need
remind_name()
guess_age()
count()
test()
end()
```

```
Hello! My name is TE-Chatbot.
I was created in 2022.
Please, remind me your name.
Tejas
What a great name you have, Tejas!
Let me guess your age.
Enter remainders of dividing your age by 3, 5 and 7.
21
3
5
Your age is 33; that's a good time to start programming!
Now I will prove to you that I can count to any number you want.
2
0 !
1 !
2 !
Let's test your programming knowledge.
Why do we use methods?
1. To repeat a statement multiple times.
2. To decompose a program into several small subroutines.
3. To determine the execution time of a program.
4. To interrupt the execution of a program.
```