CHATBOT CREATION USING PYTHON

TEAM MEMBER

211121205023:RAGHUL S

Phase 2 Submission Document

Project: Create Chatbot using Python

Introduction:

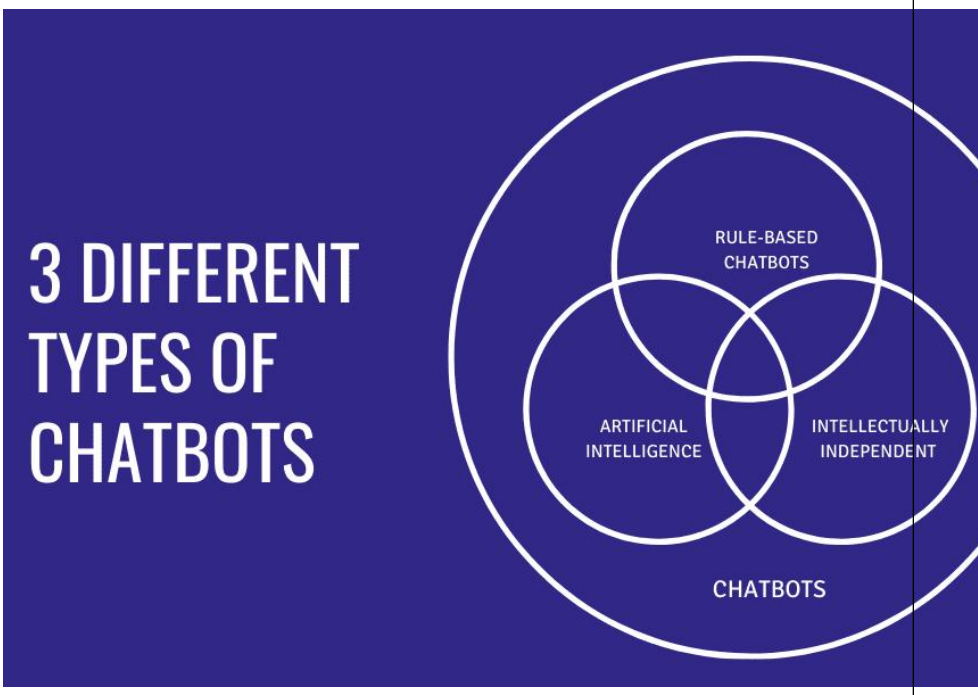
A **chatbot** *is "a computer program designed to simulate conversation with human users, especially over the internet."*

For example, let's say you wish to buy some shoes from a local retailer's website. However, you had some reservations when looking. If the business had an **online chatbot,** you could have gotten answers to all of your questions right away instead of sending long texts.

According to Mordor Intelligence, you can expect the worldwide chatbot industry to increase at a compound annual growth rate of 35 percent from 2021 to 2028, reaching $102 billion. Chatbots have several advantages, including that, unlike apps, they do not need to be downloaded. You don't need to update them, and they don't take up any memory on the phone. Another advantage is that we may have many bots in the same conversation. This way, we wouldn't have to go from one program to the next, depending on what we need.

Types of Chatbot:

There are three **types of chatbots** in today's digital realm in broad terms. These are:



We are going to use rule-based chatbot for handling customer services, decision-tree bots are another name for rule-based chatbots. As the name states, they follow a set of given rules. These guidelines serve as the foundation for the sorts of problems that the chatbot is familiar with and can solve.

Rule-based chatbots plot out talks like a flowchart. They already have a set of questions to ask the customers they are trained to answer. The customer can choose between those questions and keep moving ahead.

There are either simple or complex rules that are used in rule-based chatbots. However, they can't answer any inquiries that aren't in line with the established guidelines. Interactions do not teach these chatbots anything. Furthermore, they can only execute in specific circumstances for which they have been prepared.

While rule-based bots have a less flexible conversational flow, these safety nets are also beneficial. Chatbots that use machine learning are less predictable, so you can be more certain about the experience you'll get from them.

Advantages of Role-based chatbot:

- Easier to train in general (less expensive).
- It's simple to integrate with legacy systems.

- Streamline the transition from a computer to a human agent
- Extremely dependable and safe.
- Interactive components and media can be included.
- Aren't limited to text-based exchanges.

How do chatbot works:

Converting text or speech into structured data is a process for chatbots, especially when programmed to process natural language. Regardless, it's a process that has to be done to give users a proper response to their questions and concerns.

Here is how natural language processing may work with chatbots:

- Some chatbots use tokenization to divide certain words into pieces – or, in this case, "tokens" – that can be pretty useful or significant for the application.
- Named entity recognition looks for categories of words, like the name of a product, a user's name, or an address, and the chatbot will know what those entities are when the user enters that information in the chatbox.
- Normalization processes text so that it finds common spelling or typographical errors that could happen whenever a user creates a typo or doesn't know how it spells a particular word.
- Speech tagging allows the chatbot to identify parts of speech such as nouns, verbs, etc. This is so that the chatbot can understand complex sentence structures and how they impact meaning.

- Dependency parsing helps chatbots look for subjects and objects in a given text, leading them to dependent phrases.
- Sentiment analysis lets chatbots watch and learn if a user has a good experience or if they still need help, as compared to a human agent.

Content for Project Phase 2:

1. Data Source:

Depending on the particular use case and the needs of the chatbot, many data sources can be used to train AI chatbots. Here is the link to the dataset that will be taken into account for building a Python chatbot that offers first-rate customer support and responds to user inquiries on a website or application.

Dataset Link: https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot

```
hi, how are you doing?  i'm fine. how about yourself?
i'm fine. how about yourself?   i'm pretty good. thanks for asking.
i'm pretty good. thanks for asking.     no problem. so how have you been?
no problem. so how have you been?       i've been great. what about you?
i've been great. what about you?        i've been good. i'm in school right now.
i've been good. i'm in school right now.        what school do you go to?
what school do you go to?       i go to pcc.
i go to pcc.    do you like it there?
do you like it there?   it's okay. it's a really big campus.
it's okay. it's a really big campus.    good luck with school.
good luck with school.  thank you very much.
how's it going? i'm doing well. how about you?
i'm doing well. how about you?  never better, thanks.
never better, thanks.   so how have you been lately?
so how have you been lately?    i've actually been pretty good. you?
i've actually been pretty good. you?    i'm actually in school right now.
i'm actually in school right now.       which school do you attend?
which school do you attend?     i'm attending pcc right now.
i'm attending pcc right now.    are you enjoying it there?
are you enjoying it there?      it's not bad. there are a lot of people there.
it's not bad. there are a lot of people there.  good luck with that.
good luck with that.    thanks.
```

2. Converting raw data into CSV file:

Program:

chatbot.py

```python
import csv

# Define the input and output file paths
input_file_path = r'C:\\Users\\dhaya\\Downloads\\archive\\dialogs.txt'
output_file_path = 'C:\\Users\\dhaya\\OneDrive\\Desktop\\dataset.csv'

# Open the input text file for reading and CSV file for writing
with open(input_file_path, 'r', encoding='utf-8') as input_file, open(output_file_path, 'w', newline='', encoding='utf-8') as output_file:
    csv_writer = csv.writer(output_file)
    csv_writer.writerow(['input', 'response'])  # Write the header row

    for line in input_file:
        line = line.strip()
        if line:  # Check if the line is not empty
            # Split the line into input and response using a separator (e.g., '\t' for tab)
            input_text, response_text = line.split('\t')

            # Write the input and response to the CSV file
            csv_writer.writerow([input_text, response_text])

print(f"Conversion completed. CSV file saved at {output_file_path}")

```

Hence required csv file is created using above python code.

3. Importing Dependencies:

Program:

simple chatbot

File Edit View Run Add-ons Help

Run All Code Draft Session (6m)

[1]:

```python
import csv
```

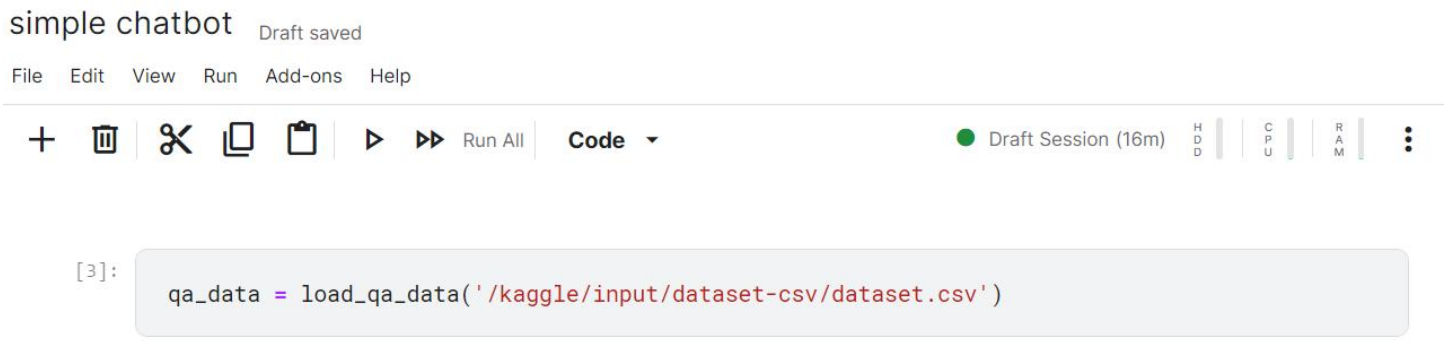
4. Define Functions in python:

1) The load_qa_data function is a Python function that reads a dataset containing questions and answers from a CSV (Comma-Separated Values) file and loads it into a list of tuples. Here's an explanation of what this function does:

- filename: This is a parameter that specifies the name (or path) of the CSV file that contains the question-answer data.
- qa_data: This is an empty list that will be used to store the loaded question-answer pairs.
- with open(filename, newline='', encoding='utf-8') as csvfile: This line opens the specified CSV file (filename) in a context manager. It ensures that the file is properly closed when the block of code inside the with statement is executed. The newline='' argument is used to handle newline characters, and encoding='utf-8' specifies the character encoding of the file
- reader = csv.reader(csvfile): This line creates a CSV reader object (reader) that is used to read the contents of the CSV file. The csv.reader function takes the opened file (csvfile) as an argument.
- for row in reader:: This is a loop that iterates through each row in the CSV file. In each iteration, row represents a row of data in the CSV file
- if len(row) == 2:: This line checks if the current row contains exactly two elements (columns). In a typical question-answer dataset, you would expect each row to have a question and an answer. If a row has more or fewer than two elements, it is skipped
- question, answer = row: If the row contains exactly two elements, this line unpacks the elements into the variables question and answer. It assumes that the first element is the question and the second element is the answer.

- qa_data.append((question.strip(), answer.strip())): This line appends a tuple containing the stripped (with leading and trailing whitespace removed) question and answer to the qa_data list. Each tuple represents a question-answer pair.

2) The line of code qa_data = load_qa_data('/kaggle/input/dataset-csv/dataset.csv') is an example of how to use the load_qa_data function to load a dataset of questions and answers from a CSV file named 'qa_data.csv'. Here's what this line of code does:



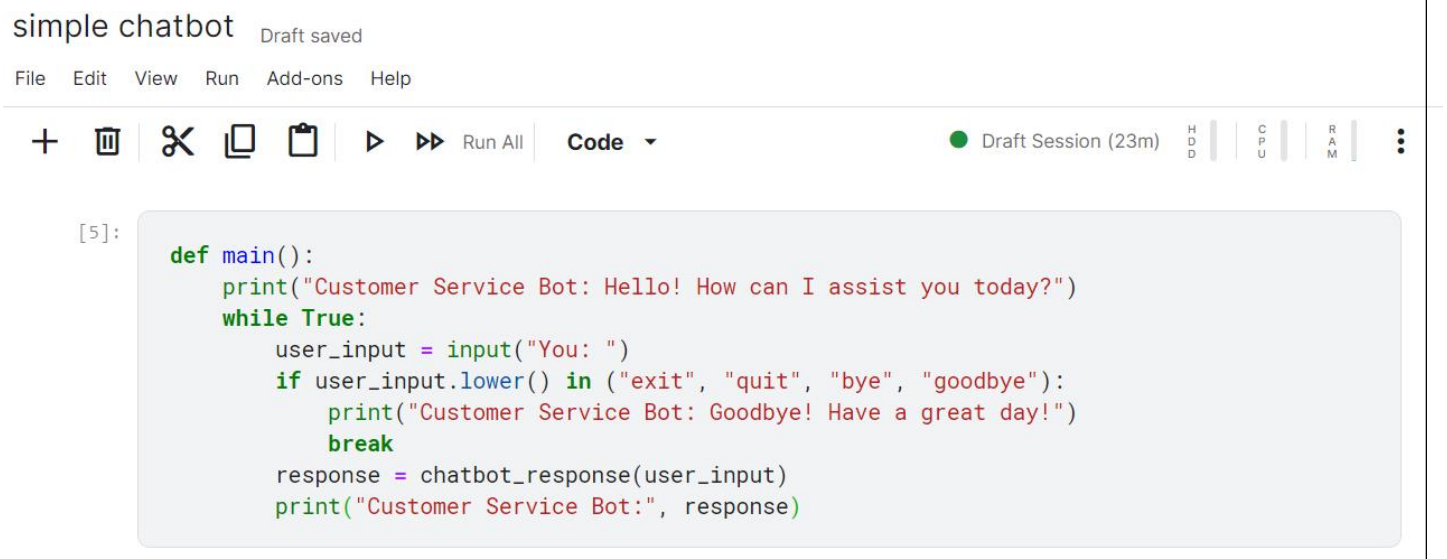
- load_qa_data('qa_data.csv'): This part of the code calls the load_qa_data function and passes the filename 'qa_data.csv' as an argument. The function will read and process the contents of the specified CSV file.

3) The chatbot_response function is responsible for providing responses based on user input by searching for a matching question in the qa_data dataset. Here's an explanation of how this function works:

- user_input: This is the input provided by the user, typically a question or statement they want the chatbot to respond to.

- qa_data: This is assumed to be a variable containing the dataset of question-answer pairs, which was loaded using the load_qa_data function.
- for question, response in qa_data:: This line sets up a loop that iterates through each question-answer pair in the qa_data dataset. For each pair, it unpacks the question and response values.
- if user_input.lower() == question.lower():: Inside the loop, it compares the lowercase version of the user_input (converted to lowercase with lower()) with the lowercase version of the question from the dataset. This comparison is case-insensitive, so it will match regardless of the letter case used by the user.
- return response: If a match is found (i.e., the user's input matches a question in the dataset), the function returns the corresponding response. This means that the chatbot will provide theresponse associated with the matched question.If no match is found after iterating through all the questions in the dataset, the function returns a default response: "I'm sorry, I don't have an answer to that question." This is a way to handle cases where the user's input doesn't match any of the questions in the dataset.
- print("Customer Service Bot: Hello! How can I assist you today?"): This line displays an initial greeting message to the user when the chatbot starts running. It informs the user that the chatbot is ready to assist them.

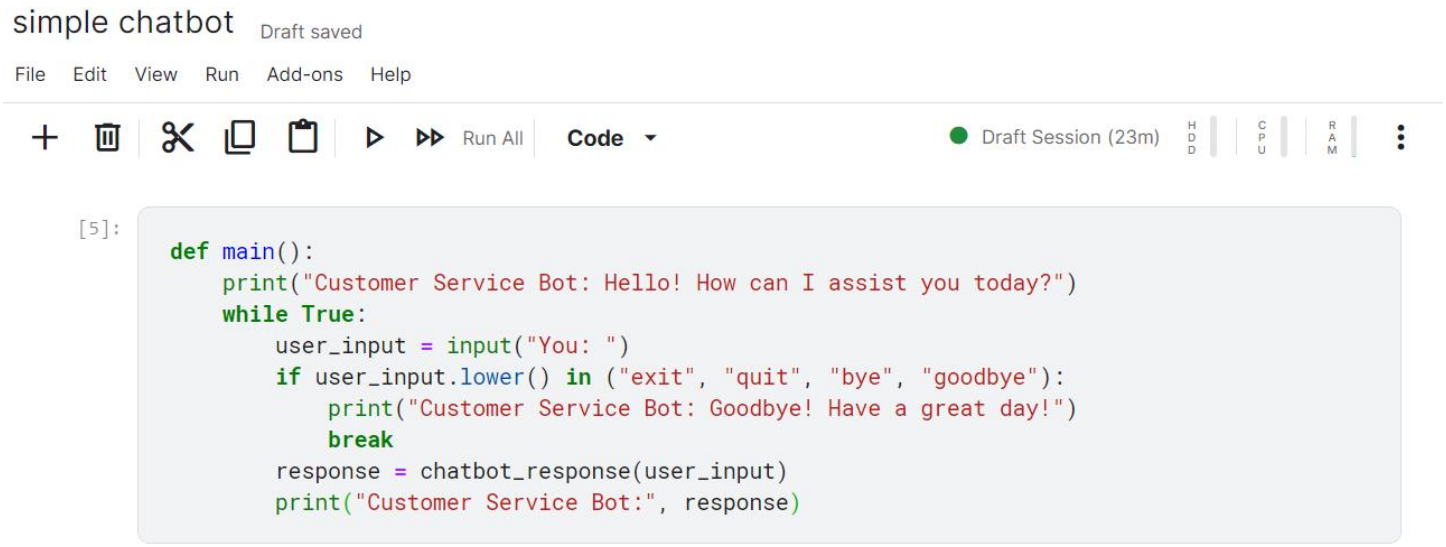
4) The main function is the core of your customer service chatbot program. It serves as the entry point where the interaction with the user takes place. Here's an explanation of how the main function works:



- while True:: This starts an infinite loop, which means the chatbot will continue to run until a specific exit condition is met.

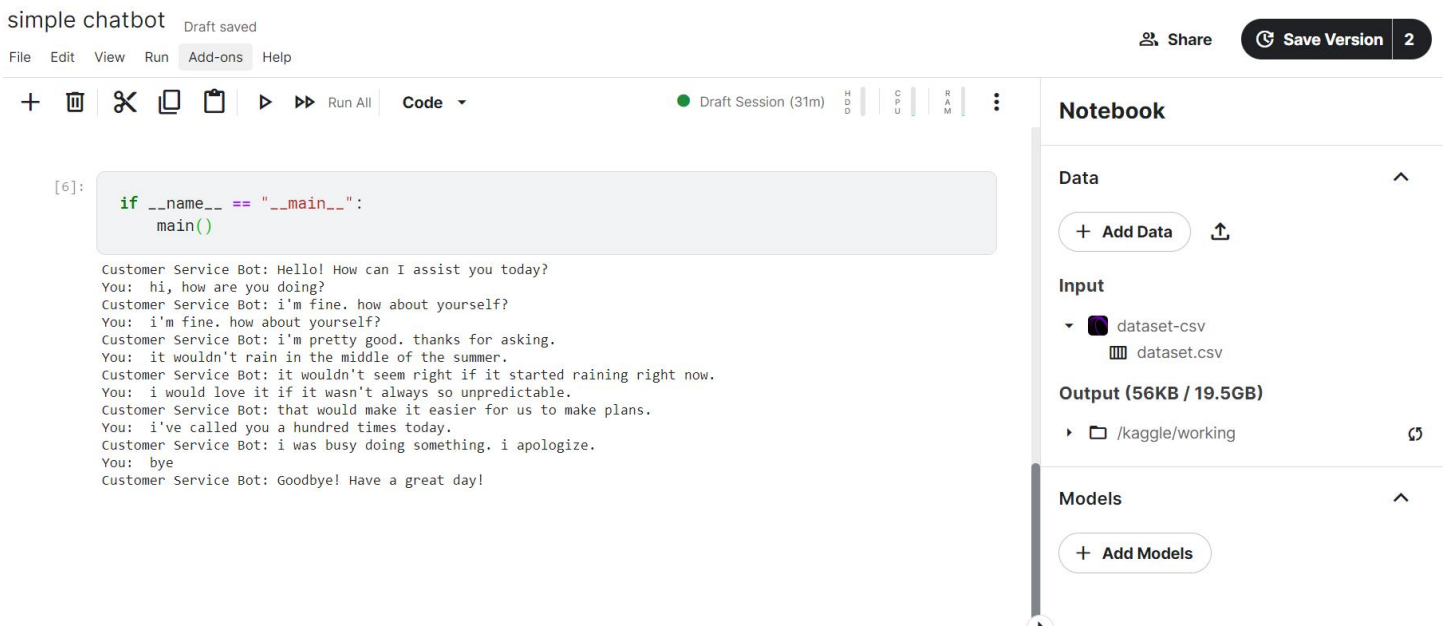
- user_input = input("You: "): This line prompts the user to enter their input by displaying "You: " and then waits for the user to type something. The user's input is stored in the user_input variable.
- if user_input.lower() in ("exit", "quit", "bye", "goodbye"):: This condition checks if the user's input (converted to lowercase using lower()) matches any of the specified exit phrases ("exit," "quit," "bye," or "goodbye"). If the user enters any of these phrases, the chatbot prints a farewell message and exits the loop, effectively ending the chatbot session.
- response = chatbot_response(user_input): This line calls the chatbot_response function and passes the user_input as an argument. It retrieves a response from the chatbot based on the user's input and stores it in the response variable.
- print("Customer Service Bot:", response): After generating a response, this line prints the response to the screen, prefaced by "Customer Service Bot:". This way, the user can easily distinguish the bot's responses from their own input.

5) The line if __name__ == "__main__": is a common Python idiom that checks whether the script is being run as the main program or if it's being imported as a module into another script. Let me explain how it works:



- __name__: In Python, __name__ is a built-in variable that is automatically set by the Python interpreter. When a Python script is executed, Python sets the __name__ variable to "__main__" if the script is the main program being run. If the script is imported as a module into another script, __name__ is set to the name of the script/module.
- "__main__": This is a string representing the name of the main script or program.

5. Output:



Project Conclusion:

There are various opportunities to develop and enhance our chatbot as we move to the future. We can improve its user interface, add more data sources, and polish its conversational skills. In order to offer helpful assistance and automation, the chatbot can also be linked into a variety of applications, including customer support, e-commerce, and healthcare.