CREATING A CHATBOT USING PYTHON

Project Report

Team Member Details:

Dhanusri.G.R (au620121104020)

Dharshini.R (au620121104022)

Divya.B (620121104028)

DivyaDharshini.R (au620121104030)

Divya.S (au620121104029)

Gayathri.S (au620121104031)

Govarthini.P (au620121104034)

Habina Roja.N (au620121104035)

INTRODUCTION

- A chatbot is a computer designed to engage in text or voice-based conversations with users.
- Chatbots use Natural Language Processing (NLP) and artificial intelligence (AI) to understand and respond to user inputs, providing information, answering questions, or assisting tasks.
- They are used in various applications, from customer support and virtual assistants to automation and information retrieval.
- Chatbots can be rule-based, following predefined scripts, or machine learning-based, following predefined scripts, or machine learning-based, learning from interactions to improve their responses.

PROBLEM STATEMENT

We aim to develop a Python-based chatbot to enhance the customer support and streamline information retrieval for their Queries on a Website or a Application. Our goal is to improve the overall user experience by providing quick and efficient assistance to our customers, reducing response times to customer inquiries, and increasing user engagement. Here's some detailed abstract steps that are going to be used to implement a chatbot.

STEPS TO BE FOLLOWED FOR CREATING A CHATBOT USING PYTHON

1. Define its purpose ad goals.
2. Choose a development platform or framework.
3. Design the conversation flow.
4. Implement NLP and AI for understanding and responding.
5. Gather relevant data.
6. Train the chatbot.
7. Test it thoroughly.
8. Deploy it to your chosen platform.
9. Continuously improve it based on feedback.
10. Address security and privacy.
11, Consider the user interface.
12. Regularly maintain and update the chatbot as needed.

DESIGN THINKING

Design thinking is a user-centric approach to creating products or solutions. When applying design thinking to chatbot creation using python, we involved these following steps:

1.Empathize

- Understanding the needs and preferences of our target audience.
- Conducting user research, interviews and surveys to identify main point and goals.

2.Define

- Clearly articulate the problem that the chatbot will solve.
- Setting the specific objectives and constraints for our chat project.

3.Prototype

- Developing a low-fidelity prototype of out chatbot using python. This can be a basic text-based interface.
- Focus on functionality rather than aesthetics at this stage.

4.Ideate

- Brian storm possible solutions and features for the chatbot.
- Encourage creativity and generate wide range of ideas.

5.Test

- Gather feedback from potential users by testing your prototype.
- Analyse user interactions to identify what works and what needs improvement.

6.Build

- Develop the chatbot using python. We can use libraries like NTLK, spaCy, or frameworks like Rasa to build a conversational AI.

7.Refine

- Refine the chatbot based on user feedback and testing results. Continuously improving its performance and features.

8.Deploy

Deploy our chatbot, making it accessible to users.

Ensure it can handle different environments, like websites or messaging platforms.

9.Learn

- Collect data and analytics to monitor the chatbot's performance.
- Use this data to make data-driven improvements.

10.Maintain

- Regularly update and maintain the chatbot to keep it up-to-date and improve its capabilities.

By using these design thinking steps, we can create a chatbot using python that addresses the specific needs and preferences of our target audience, resulting in a more user-friendly and effective solution.

PHASES OF DEVELOPMENT

There are several approaches to developing chatbots, depending on goals and technical expertise. Here are some common approaches to chatbot development:

1.Feature Engineering for chatbot in python

Text processing: tokenization-split the text, lower casing-clean the text

Text vectorization: Bag of words (BOW)-convert the text, TF-IDF(Term Frequency-Inverse Document Frequency)- assign weights, word embeddings-use pre-trained word.

Intent recognition: Train a model

Entity recognition: Identify entities.

Dialog State tracking: Track of the conversation.

2.MODEL TRAINING

Training the chatbot using python to answer user queries.

Data collection: Gather dataset of user queries and corresponding responses.

Processing: Clean and preprocess the data.

Text processing: tokenization-split the text, lower casing-clean the text

Natural Language Processing: Using NLP libraries.

Text processing: tokenization-split the text, lower casing-clean the text

Choose a framework: Chatterbot, Rasa, TensorFlow and pyTorch.

Text processing: tokenization-split the text, lower casing-clean the text

Model architecture: Designing the architecture.

Training the Model: We can use the ChatterBotCorpus Trainer.

Maintenance: Regularly updating and maintaining.

3.EVALUATION

Functional Testing of the chatbot. This involves following step.

Functional Testing, User experience and usability, Accuracy and intent Recognition, Dialog Flow, A/B testing, Load Testing, Bug and error monitoring, Compliance and Regulations and security.

Here are some steps for developing chatbot. They are

CHOOSE A PYTHON LIBRARY: Chatterbot libraries are Rasa and DialogFlow.

DESIGNING CHATBOT: Types of conversations that your chatbot will need to be able to handle.

TRAINING CHATBOT: This involves training the dataset that we have chosen.

DEPLOY CHATBOT: Deploy it to a production environment.

Step by Step Process

- ✓ **Install the required libraries:**
  Install required libraries like numpy, pandas, ntlk and scikit-learn.
- ✓ **Load the datasets:**
  Loading the datasets that we are going to use in out chatbot development.
- ✓ **Preprocess the datasets:**
  Pre-processing datasets like spelling corrections and converting into lowercase.
- ✓ **Train machine learning models:**
  Training a machine learning models using machine learning algorithms.
- ✓ **Build chatbot models:**
  Building the chatbot interface either it may be for a web application or a mobile or messaging application.

  Here are some additional elements for machine learning model:

- ✓ **Using a Pre-trained machine learning model:**
  This helps us to save lot of time.
- ✓ **Using a chatbot framework:**
  This involves using a framework like Rasa, DialogFlow, Amazon Lex.
- ✓ **Testing the chatbot:**
  This involves various testing techniques and gathering user feeback.

COMPLETE PYTHON CODE FOR CREATING A CHATBOT USING PYTHON

Pip install chatterbot

Code explanation:

- **This is code installs the python package called "chatterbot" using the pip package manager.**
- **Pip is a package that allows us to easily install libraries and packages.**
- **The "chatterbot" package is a python library that makes it easy to generate automated responses to a user's input, making it useful for building chatbots and conversational agents.**
- **By running this code, the "chatterbot" package gets installed and downloaded on the user's system.**

Pip install chatterbot_corpus

Code explanation:

- **This is code installs the python package called "chatterbot_corpus" using the pip package manager.**
- **The "chatterbot_corpus" package contains the pre-built training data for chatterbot library, which is a python library for creating chatbots.**
- **When this command is executed, pip will download the chatterbot_corpus package and its dependencies form PyPI and install them on a local machine.**

!Pip install chatterbot

Code explanation:

- **The exclamatory mark at the beginning indicates that this is a command to be executed in the command line interface (CLI) rather than in python code.**
- **The pip package manager python packages, and the "install" command is used to install a package.**
- **The chatterbot package is a python library that enables developers to build chatbots and conversational agents.**

!Pip install chatterbot_corpus

Code explanation:

- **The exclamatory mark at the beginning indicates that this is a command to be executed in the command line interface (CLI) rather than in python code.**
- **The "pip" command is used to install packages in python, and "chatterbot_corpus" is the name of the package being installed.**
- **This code installs all the python package called "chatterbot_corpus" using the pip package manager.**

#importing chatterbot from chatterbot import chatbot

Code explanation:

- **This code imports the chatbot class from the chatterbot module.**
- **The chatterbot module is a python library that makes it easy to generate automated responses to a user's input.**
- **By improving the chatbot class, we can create an instance of a chatbot that we can then train and use to generate responses to user input.**

#create object of chatbot class bot=chatbot('Buddy')

Code explanation:

- **This code creates an object of the chatbot class and assign it to the variable bot.**
- **The chatbot class is likely defined in a library or module that has been imported into the current python script,**
- **The argument 'Buddy' is passed to the chatbot constructor, which is likely used to set the name of the chatbot.**

[ntlk_data]
Downloading package
Averaged_perception_tagger to [ntlk_data] /root/ntlk_data...
[ntlk_data] unzipping taggers/
averaged_perception_tagger.zip.[ntlk_data]
Downloading package stopwardes to /root/ ntlk_data....
[ntlk_data] unzipping corpora/stopward.zip.
[ntlk_data] downloading package wordnet to /root/ ntlk_data...
[ntlk-data] unzipping corpora/wordnet

Code explanation:

- **This code is using the Natural Language Toolkit (ntlk) library in python to download three packages: averaged_perception_tagger, stopword, wordnet.**

- These packages contain data and models for Natural Language Processing tasks such as part-of-speech tagging, stop word removal, word sense disambiguation.
- These code downloads these packages and unzips them into the specified directory (/root/ntlk_data) so that they can be used in the program.

#create object of chatbot class with Storage Adapter bot=chatbot('Buddy' storage _adapter='chatterbot.storage.SQLStorageAdapter', database_uri='sqlite://database.sqlite3'

Code explanation:

- This code creates an object of the chatbot class from the chatterbot library.
- The chatbot class is used to create chatbots that can engage in conversation with users.
- The first argument passes to the chatbot constructor is the name of the chatbot, which is set to 'Buddy' in this case.
- The second argument, storage_ adapter specifies the type of storage adapter to use for storing the chatbot's training data and conversation history.
- In this case, the SQLStorageAdapter is used, which stores the data in SQLite database.
- The third argument, database_uri, specifies the location of the SQLite database.
- In this case, the database is located in a file name database.sqlite3 in the current directory.

#create object of chatbot class with Logic Adapter bot=chatbot('Buddy', logic_adpters={

'chatterbot.logic.BestMatch',

'chatterbot.logic.TimeLogicAdapter'],)

Code explanation:

- This code creates an instance of a chatbot using the chatbot class from the chatterbot library.
- The chatbot is given the name "Buddy".
- The logic_adapters parameter is used to specify the logic adapters the chatbot will use to generate responses.
- In this case, the chatbot will use two logic adapters: BestMatch and TimeLogicAdapter.

- **The BestMatch adapter uses a combination of cosine similarity and levenshtein distance to find the closest matching response to a given input.**
- **The TimeLogicAdapter adapter allows the chatbot to respond to questions about the current time.**
- **Overall, this code creates a chatbot instance with two logic adapters that will be used to generate responses to user input.**

#import ListTrainer from chatterbot.trainers import ListTrainer

Trainer=ListTrainer(bot)

Trainer.train(['hi' ,'hello' ,'I need your assistance regarding my order id please, provide me with your order', ' I have a compliant', ' please elaborate your concern', ' how long it will take to receive an order?', ' an order takes 3-5 business days to ger delivered', . okay thanks',' no problem! Have a nice day'])

Trainer.train([{' what is your name?', ' I am a chatbot'},{'how are you?',' I don't have feelings, but I am here to help you'},{'tell me a joke' , 'why did bicycle fall over? because it was two-tired'},{'what's the weather today?','I don't have access to real-time weather information'},{'who won the world series in 2020?', ' los angels dodgers won the world series'}]

Trainer.train([{'what is your favourite color?', ' I don't have preferences but I think blue is a nice color'},{what is capital of France?','capital of France is paris'},{'what is photosynthesis?',' photosynthesis is the process by which plants convert the light energy into energy. They use sunlight to turn carbpn dioxide and water into glucose and oxygen'},{'who is albert Einstein?','Albert Einstein was a famous physicist know for his theory of relativity.he's one of the most influential scientist of the 20th century'}])

Code explanation:

- **This code is used to train a chatbot using ListTrainer.**
- **The chatbot will use this training data to learn how to answer to similar inputs in the future.**
- **The trainer.train() method is called with a list of conversation pairs as an argument.**
- **Then, a trainer object is created by passing in the chatbot object as an argument.**

List trainer
[##################]100%

Code explanation:

- In this case, it is indicating that the 'ListTrainer' task completed with 100% progress.
- The # symbols represent the progress bar filling up as the task is completed.
- No specific programming language is specified In this snippet.

#get response to the input text ' I would like to book a flight'.

Response='bot.get_response('I have a complaint')

Print("bot response:",response)

Code explanation:

- First, the get_response() method of the bot object is called with input text as its argument.
- This method uses a natural language processing algorithm to analyse the input text and generate a response based on the chatbot's programming.
- The print () function is used to display the generated response in the console with the message "Bot Response: preceding it.

Bot Response: please elaborate, your concern

Code explanation:

- This code snippet is not actually code, but rather a message from a chatbot asking the user to provide more information about their concern.
- It is not written in any specific programming language, but rather in natural language.

Print("welcome to the bot service! Let me know how can I help you?")

While True:

Request=input(name+':')

if request=='bye' or request=='Bye':

Print("bye")

Break

else:

response=bot.get_response(request)

print('Bot:'. reponse)

Code explanation:

- **This code first prints the welcome message. Next, enter into while loop that will continue to run until the user enter "bye" or "Bye".**
- **Within the loop, the code prompts the user to enter a request variable.**
- **The code passes to the users request to a chatbot using the bot.get_response() function and stores the response in the response variable.**
- **Finally, the code prints the bot's response using the print() function.**

SAMPLE OUTPUT:

Here is a sample output for the above code execution.

Bot: welcome to the Bot Service! Let me know how can I help you?

User: who is Albert Einstein?

Bot: Albert Einstein was a famous physicist known for his theory of relativity. He's one of the most influential scientists of the 20th century.

User: what's the capital of the France?

Bot: The capital of France is Paris.

User: how are you?

Bot: I don't have feelings, but I'm here to help!

User: Bye

Bot: Bye

INNOVATIVE TECHNIQUES AND APPROACHES

1.Rule-based chatbots:

Static rules: chatbots follow predefined rules and keywords or patterns in user input.

2.Machine Learning-based chatbots:

this involves NLP, Supervised Learning, Reinforcement Learning and Deep Learning.

3.Hybrid chatbots:

Combines rule-based and machine learning-based.

4.Generative pre-trained Transformers (GPT) -based chatbots:

Building chatbots using large models like GPT-3 or GPT-4.

5.API-based chatbots:

Building chatbots and integrate with existing API's and services.

6.Voice-Enabled chatbots:

Develop chatbots that can interact with users.

7.Converational UI designs:

Focus on creating User-friendly design.

8.Platform specific chatbots:

Build chatbots for specific messaging platforms like Facebook Messenger, slack, or WhatsApp.

9.Multimodal chatbots:

Extend chatbot to support multiple.

10.Continuous Development:

Implementing mechanism for continuous learning.

11.Educational chatbots:

It can assist in learning new skills, tutoring and providing personalized educational content.

And there are more innovative techniques in chatbot.

The choice of development approach depends on your project requirements, available resources, and the level of sophistications.

CONCLUSION

Chatbots are AI-driven programs that engage in conversations with users, offering benefits such as 24/7 availability, efficiency, scalability, and cost savings.

They are used in various sectors, including customer support, e-commerce, healthcare and marketing.

However, challenges include Natural Language understanding and the ongoing improvement.

Technologies like NLP and machine learning power chatbots, and various libraries and platforms facilitate their development.

Overall, chatbots enhance user experiences and provide valuable solutions for business and organization.