NAME : JAISURYAH K P
REGNO: 727822TUIT063
DEPT : BTECH INFORMATION TECHNOLOGY
SECTION : IT - A
COURSE : ARTIFICIAL INTELLIGENCE
COURSE CODE : 22IT401

CHATBOT

Abstract:

ConvoBot is a Python-based conversational agent designed to engage in text-based interactions with users. Built using the NLTK library, ConvoBot employs pattern matching techniques to recognize user input and generate appropriate responses. The bot can handle a range of queries, including greetings, inquiries about its capabilities, requests for information about time and date, and offers of assistance. Implemented functionalities include retrieving current time and date in the Indian time zone and providing generic responses for unrecognized input. While ConvoBot demonstrates basic conversational abilities, there are opportunities for improvement in areas such as expanding functionality, enhancing natural language understanding, improving error handling, introducing personalization, and ensuring scalability. This abstract provides an overview of ConvoBot's features, implementation details, and potential areas for future development, highlighting its role as a simple yet promising conversational agent

Introduction:

ConvoBot is a simple conversational agent programmed to engage in text-based conversations with users. Built using Python, it utilizes the NLTK (Natural Language Toolkit) library to recognize patterns in user input and generate appropriate responses. This report will provide an overview of ConvoBot's functionalities, its implementation details, and potential areas for improvement.

Top Chatbots as per 2024

1. ChatSpot
2. ChatGPT
3. Bing Chat
4. Bard
5. Jasper Chat
6. Perplexity
7. Tidio Lyro
8. Kommunicate
9. HubSpot Chatbot Builder
10. Intercom
11. SmythOS
12. Watson Assistant
13. Drift
14. Infobip
15. Appy Pie Chatbot
16. Zendesk Answer Bot
17. Salesforce Einstein
18. LivePerson
19. Ada

Functionality:

ConvoBot is designed to handle a variety of user queries and interactions. It can respond to greetings, inquiries about its name and purpose, questions regarding time and date, requests for assistance, and general conversational prompts. Some key functionalities include:

Greetings: ConvoBot greets users with responses such as "Hello!", "Hey there!", and "Hi!" when greeted with salutations like "hi," "hello," or "hey."

Information Provision: It can provide information about itself, including its name ("ConvoBot") and its capabilities ("I can chat with you," "I can answer your questions").

Time and Date: ConvoBot can inform users about the current time and date in the Indian time zone.

Assistance: It offers assistance to users by responding affirmatively to queries like "can you help me?" and expressing willingness to assist.

Handling Unrecognized Input: When confronted with input it cannot understand, ConvoBot provides generic responses like "Sorry, I did not understand that" or "Could you please rephrase that?"

Implementation:

ConvoBot is implemented using Python, making use of the NLTK library for natural language processing tasks. Key components of its implementation include:

Patterns and Responses: User input is matched against predefined patterns using regular expressions. These patterns are associated with corresponding responses, which are selected randomly from a list of possible answers.

Time and Date Retrieval: ConvoBot retrieves the current time and date using Python's datetime module in conjunction with the pytz library to handle time zone information. The time and date are specifically obtained in the Indian time zone ("Asia/Kolkata").

User Interaction Loop: ConvoBot engages in a continuous interaction loop with the user, prompting for input, processing it, and providing responses until the user decides to quit the session.

Packages Used:

- **random:** This is a built-in Python module that provides functions for generating random numbers. It's used in this code to select random responses from predefined lists of responses to certain user inputs.

- **nltk (Natural Language Toolkit):** nltk is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

- **nltk.chat.util:** This submodule of nltk provides utilities for building simple chatbots, including the Chat class used to define patterns for matching user inputs and generating responses.

- **datetime:** This is a built-in Python module that provides classes for manipulating dates and times. It's used in this code to get the current date and time in the Indian time zone.

- **pytz (Python Time Zone):** pytz is a Python package that provides functions and classes for working with time zones. It's used in this code to get the current date and time in the Indian time zone

Functions:

get_current_indian_datetime(): This function retrieves the current date and time in the Indian time zone ("Asia/Kolkata"). It utilizes the datetime module along with the pytz library to obtain the current date and time information and formats it accordingly. The formatted date and time are then returned by the function.

chat(): The chat() function serves as the main interaction loop for ConvoBot. It prompts the user for input, processes the input, generates responses using the provided patterns, and continues the interaction until the user decides to quit the session by typing "quit." Within this function, the current date and time are displayed at the beginning and end of the session, and user input is processed using the chatbot.respond() method, which utilizes the predefined patterns to generate responses.

patterns: The patterns list contains tuples of regular expression patterns and corresponding lists of responses. These patterns are used by the Chat object from NLTK to match user input and generate appropriate responses. Each tuple consists of a regular expression pattern and a list of possible responses associated with that pattern. When a user input matches a pattern, one of the responses from the corresponding list is randomly selected and returned as the bot's response.

chatbot: The Chat object from NLTK is initialized with the predefined patterns and used to handle the conversation with the user. It utilizes the patterns to recognize user input and select appropriate responses. The respond() method of the Chat object is called within the chat() function to generate responses based on user input.

CODING:

```
import random
import nltk
from nltk.chat.util import Chat, reflections
from datetime import datetime
import pytz

patterns = [
    (r'hi|hello|hey', ['Hello!', 'Hey there!', 'Hi!']),
    (r'how are you', ['I am fine, thank you!', 'I am doing well, thanks!']),
    (r'what is your name', ['My name is ConvoBot.', 'I am a ConvoBot.', 'I go by
ConvoBot.']),
    (r'what can you do', ['I can chat with you!', 'I can answer your questions.', 'I can
provide information.']),
    (r'(.*) age (.*)', ['I do not have an age.', 'Age is just a number for me.']),
    (r'(.*) (location|located) (.*)', ['I exist in the virtual world.', 'You can find me wherever
you are!']),
    (r'(.*) (weather|temperature) (.*)', ['I do not have access to weather information.', 'You
can check the weather on your phone or computer.']),
    (r'what is the time', ['The current Indian time is ' +
datetime.now(pytz.timezone('Asia/Kolkata')).strftime('%H:%M:%S')]),
    (r'what is the date', ['Today\'s Indian date is ' +
datetime.now(pytz.timezone('Asia/Kolkata')).strftime('%Y-%m-%d')]),
    (r'how old are you', ['I am a computer program, so I do not age.', 'Age is not
applicable to me.']),
    (r'who created you', ['I was created by a team of developers.', 'My creators are human
programmers.']),
    (r'can you help me', ['Of course! What do you need help with?', 'I will do my best to
assist you.']),
    (r'(.*) (love|hate) (.*)', ['Feelings are complex and subjective.', 'Love and hate are
human emotions.']),
    (r'(.*)', ['Sorry, I did not understand that.', 'Could you please rephrase that?', 'I am still
learning!'])
]
```

```
chatbot = Chat(patterns, reflections)

def get_current_indian_datetime():
    current_time = datetime.now(pytz.timezone('Asia/Kolkata')).strftime('%H:%M:%S')
    current_date = datetime.now(pytz.timezone('Asia/Kolkata')).strftime('%Y-%m-%d')
    return current_date, current_time

def chat():

    indian_date, indian_time = get_current_indian_datetime()
    print(f"Welcome! Current Session date is {indian_date} and time is {indian_time}.
Type 'quit' to exit.")
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'quit':

            indian_date, indian_time = get_current_indian_datetime()
            print(f"Goodbye! Session end time is {indian_date} and time is {indian_time}.")
            break
        else:
            response = chatbot.respond(user_input)
            print("ConvoBot:", response)

if __name__ == "__main__":
    chat()
```

OUTPUT:

Welcome! Current Session is 2024-03-16 and time is 20:39:13. Type 'quit' to exit.
You: Hii
ConvoBot: Hi!
You: how are you
ConvoBot: I am doing well, thanks!
You: what is your name
ConvoBot: I am a ConvoBot.
You: what is the date
ConvoBot: Today's Indian date is 2024-03-16
You: i love you

ConvoBot: Feelings are complex and subjective.
You: jsancasihnc
ConvoBot: Could you please rephrase that?
You: quit
Goodbye! Session end time is 2024-03-16 and time is 20:41:06.

Potential Improvements:

While ConvoBot demonstrates basic conversational capabilities, there are several areas for potential improvement:

Expanded Functionality: ConvoBot could be enhanced with additional features such as weather information retrieval, basic arithmetic calculations, or integration with external APIs for more dynamic responses.

Natural Language Understanding: Improving the bot's ability to understand and respond to a wider range of user inputs could enhance its utility and user experience. This could involve incorporating more sophisticated natural language processing techniques or machine learning models.

Error Handling: Implementing more robust error handling mechanisms could help ConvoBot better handle unexpected or erroneous user inputs, providing clearer prompts for rephrasing or clarification.

Personalization: Introducing elements of personalization, such as remembering past interactions or preferences, could make the user experience more engaging and tailored to individual users.

Scalability: As the volume of interactions increases, ensuring that ConvoBot can handle concurrent users efficiently and scale gracefully would be important for maintaining performance and responsiveness.

Conclusion:

ConvoBot represents a basic yet functional conversational agent capable of engaging in text-based interactions with users. While it currently provides rudimentary responses to a limited set of queries, there is ample room for enhancement and refinement to make it more versatile, intuitive, and user-friendly. With ongoing development and iteration, ConvoBot has the potential to evolve into a more sophisticated and valuable conversational companion.