CREATING A CHATBOT USING PYTHON

TEAM MEMBER

510521104048: S.SOUNDARIYA

PHASE 2: INNOVATION OF CHATBOT

INTRODUCTION:

Artificial Intelligence (AI) increasingly integrates our daily lives with the creation and analysis of intelligent software and hardware, called intelligent agents. Intelligent agents can do a variety of tasks ranging from labor work to sophisticated operations. A chatbot is a typical example of an AI system and one of the most elementary and widespread examples of intelligent Human-Computer Interaction (HCI).

It is a computer program, which responds like a smart entity when conversed with through text or voice and understands one or more human languages by Natural Language Processing (NLP). Chatbots are also known as smart bots, interactive agents, digital assistants, or artificial conversation entities.

HISTORY :

Alan Turing in 1950 proposed the Turing Test (“Can machines think?”), and it was at that time that the idea of a chatbot was popularized. The first known chatbot was Eliza, developed in 1966, whose purpose was to act as a psychotherapist

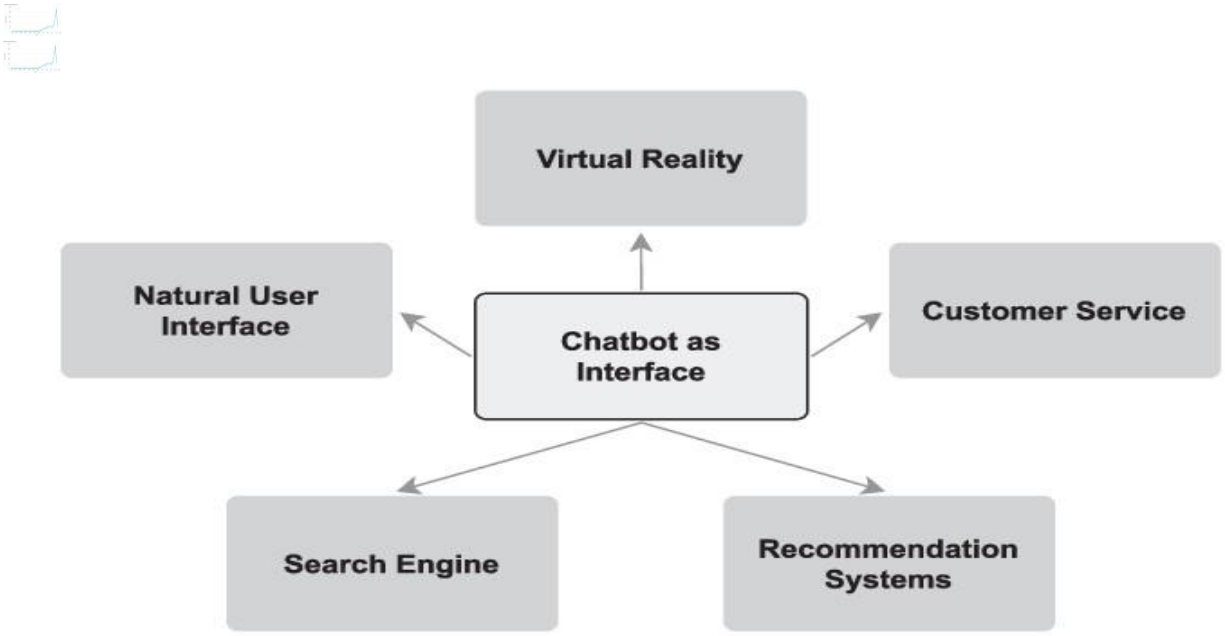
returning the user utterances in a question form. It used simple pattern matching and a template-based response mechanism.

Its conversational ability was not good, but it was enough to confuse people at a time when they were not used to interacting with computers and give them the impetus to start developing other chatbots. An improvement over ELIZA was a chatbot with a personality named PARRY developed in 1972. In 1995, the chatbot ALICE was developed which won the Loebnier Prize, an annual Turing Test, in years 2000, 2001, and 2004.

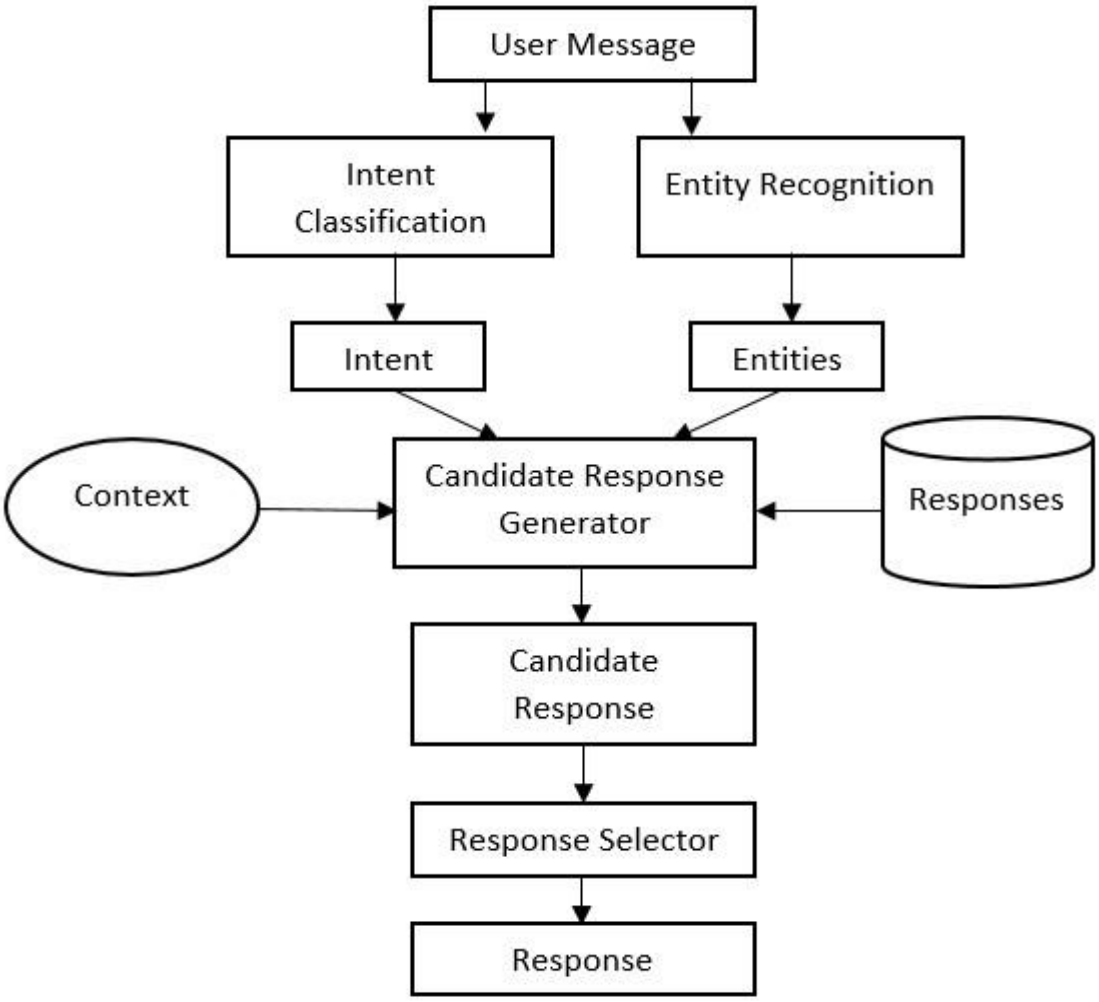
It was the first computer to gain the rank of the “most human computer”. ALICE relies on a simple pattern-matching algorithm with the underlying intelligence based on the Artificial Intelligence Markup Language (AIML), which makes it possible for developers to define the building blocks of the chatbot knowledge.

Chatbots, like Smarter Child in 2001, were developed and became available through messenger applications. The next step was the creation of virtual personal assistants like Apple Siri, Microsoft Cortana, Amazon Alexa, Google Assistant and IBM Watson.

PROGRESS IN CHATBOT:



ARCHITECTURE OF CHATBOT:



BUGS OR PROGRAMMING ERRORS:

some common issues or bugs found in chatbots are:

- Delayed response time.
- Inability to understand user queries or respond appropriately.
- Lack of personalization or customization.
- Over-reliance on scripted responses.
- Inconsistent user experience.
- Inability to handle complex queries or contexts.
- Integration issues with other systems or applications.
- Troubleshooting errors and maintaining codebase.
- Difficulty in handling multiple languages.

Developers need to continuously improve the chatbot's performance and make sure it caters to user needs to provide a seamless experience.

SOLVING THE BUGS:

Delayed Response:

```
customMessageDelay: (message) => {
                let delay = message.length * 30;
if (delay > 6000) delay = 6000;
if (delay < 100) delay = 1000;
return delay;},
```

That is creating quite a nice delay, but I would prefer having a delay of the messages depending on the length of the previously posted message:
So, let's say the first message (m1) that gets posted by the bot is always delayed by a constant value t1 and the second message posted message (m2) gets delayed by m1.length*t2. How can I achieve this?

SOLUTION:

replace this code with us and I will get the fast reply (idle response scenario for chatbot):

```
customMessageDelay: (message) => {
let delay = message.length * 30;          if
(delay > 3 * 200) delay = 3 * 200;
if (delay < 100) delay = 100;
return delay;
```

- **Improve server speed**: If the chatbot's delay is due to slow server response, you can upgrade your server infrastructure to handle a larger volume of requests and process them faster. ○ **Optimize code efficiency**: Review the code of your chatbot to identify any bottlenecks or inefficient processes. Optimize the code to make it more efficient and reduce unnecessary delays.
- **Monitor and analyze performance:** Continuously monitor the chatbot's performance and identify patterns or issues related to response time. Analyze logs and metrics to pinpoint areas where improvements can be made.

- **Apply machine learning techniques:** Utilize machine learning algorithms to improve the chatbot's performance over time. Ensure the chatbot learns from user interactions and continuously improves its response time and accuracy.

TROUBLESHOOTING ERROS AND MAINTAINING CODEBASE:

- **Debugging**: The first step in troubleshooting errors is to identify and locate the issue. Use debugging tools and techniques to understand the flow of the code, examine variable values, and find any errors or exceptions. This can involve stepping through the code, adding log statements, or using a debugger. ○ **Error Handling**: Implement robust error handling mechanisms in your code. Use try-catch blocks to catch and handle exceptions gracefully. Provide meaningful error messages to help identify and resolve issues more effectively. Consider logging errors to keep track of recurring issues. ○ **Unit Testing:** Write unit tests to ensure that each component of your chatbot is functioning correctly. Test different scenarios and inputs to cover as many edge cases as possible. Regularly run these tests to catch errors early and ensure stable performance.

DIFFICULTY IN HANDLING MULTIPLE LANGUAGES:

- **Language detection**: Implement a language detection mechanism to identify the language of the user's input. There are various language detection libraries available like Google's language-detection library or Natural Language Toolkit's (NLTK) language detection module. ○ **Data collection**: Gather a diverse dataset for each language, including user inputs and corresponding responses. This will help train the languagespecific models and improve the chatbot's performance. ○ **Multilingual chatbot:** If you have the resources and capability, you can develop a multilingual chatbot that can handle multiple languages seamlessly. This involves training and fine-tuning models for multiple languages and implementing language-specific processing pipelines.

KEY FEATURES:

- **Natural Language Understanding**: Implement Natural Language Processing (NLP) techniques to understand user input, including intent recognition and entity extraction.
- **Task Automation**: Allow users to perform tasks such as setting reminders, sending emails, searching the web, or providing weather updates.
- **Information Retrieval**: Enable the chatbot to answer general knowledge questions, provide news updates, or fetch data from APIs (e.g., weather, Wikipedia).
- **Personalization**: Implement user profiles to provide personalized responses and recommendations based on user preferences and history.
- **Multi-Platform Integration**: Deploy the chatbot on multiple platforms, such as a web app, mobile app, or integrate it with messaging platforms like Telegram or WhatsApp.
- **Voice Interaction (Optional):** If you want to take it a step further, integrate speech recognition and synthesis to allow users to interact with the chatbot through voice commands.

CONCLUSION :

Chatbots or smart assistants with artificial intelligence, in my opinion, are revolutionizing the world. In comparison to larger chatbots developing a simple chatbot is not a difficult effort, and developers need understand and address concerns such as reliability, scalability, and adaptability, as well as high level of intent on human language. Chatbots are more effective than people in reaching out to a big audience via messaging apps. They have the potential to become a useful information gathering tool in the near future.