Introduction

Artificial Intelligence (AI) has emerged as one of the most transformative technologies of our time, reshaping the way we interact with machines and access information. One important example of a technology that is gaining popularity in the field is an AI Chatbot. Chatbots, often hailed as the virtual assistants of tomorrow, have already found their place in diverse domains, from customer support and healthcare to education and entertainment. They are intelligent agents that simulate human-like conversations, enabling seamless interactions between humans and machines.

In this era of rapid technological advancement, the development of AI-powered chatbots stands as a remarkable testament to the convergence of human ingenuity and cutting-edge computing capabilities.

Content

The chatbot we have developed can answer 60 predefined questions. The questions that the user inputs are compared to the predefined questions using the **fuzzywuzzy library in python** which is used for comparing strings. This is to ensure that the chatbot provides a response even if there is a slight grammatical or spelling error. The chatbot answers the question but also provides a Wikipedia link if the user wants to know more.If a question asked by a user is not part of the predefined questions, then the chatbot will use Wikipedia API to search for the answer and provide the summary of the Wikipedia article as the answer. If both the search in the predefined questions and the Wikipedia search fail to produce a relevant answer (for example, if the user asks a highly specific or obscure question), the chatbot will provide a default fallback response, i.e. “I couldn’t find an answer. Can you please rephrase your question?” The Chatbot also recognizes both text and voice as an input. It uses Python’s speech recognition library to perform this feat. The spoken speech is then converted to text for the interpreter using ‘Google Web Speech API’.

Algorithm

1. Initialization:

- Initialize the text-to-speech engine (e.g., pyttsx3).
- Set the desired voice and speech rate.
- Initialize the speech recognition module (e.g., speech_recognition).
- Create an instance of the chatbot.

2. Chatbot Class:

- Initialize the chatbot class.
    1. Initialize the text-to-speech engine.
    2. Initialize the speech recognition module.
- speak(audio):
    1. Speak the provided audio using the text-to-speech engine.
    2. Print the spoken audio.

    3. Run and wait for the speech to finish.

- take_command():
    1. Use a microphone as the audio source.
    2. Adjust recognition settings (e.g., pause threshold and ambient
    3. noise).
    4. Listen for user input audio.
    5. Use Google's speech recognition to convert audio to text.
    6. Return the recognized query.
- clean_summary(summary):
    1. Remove any unwanted content from a Wikipedia summary.
    2. Return the cleaned summary.
- search_answers(question):
    1. Read questions and answers from text files (questions.txt and answers.txt).
    2. Calculate the similarity between the user's question and stored questions.
    3. If a similar question is found (similarity threshold = 75), return the corresponding answer.
- chatbot_response(user_input):
    1. Determine if the user is in 'text or 'voice' mode based on user input.
    2. Provide an initial response based on the mode.
    3. Handle user interactions:
        - ➢ In 'text' mode:
            - ▪ Allow the user to input questions.
            - ▪ Check for special commands like 'exit' or 'help'
            - ▪ Search for answers in the provided text files.
            - ▪ If not found, retrieve a Wikipedia summary and
            - ▪ provide a Wikipedia link
        - ➢ In 'voice' mode:
            - ▪ Allow the user to speak questions.
            - ▪ Listen for voice input and recognize it.
            - ▪ Search for answers in the provided text files.
            - ▪ If not found, retrieve a Wikipedia summary and
            - ▪ provide a Wikipedia link.

    4. Return the chatbot's response.

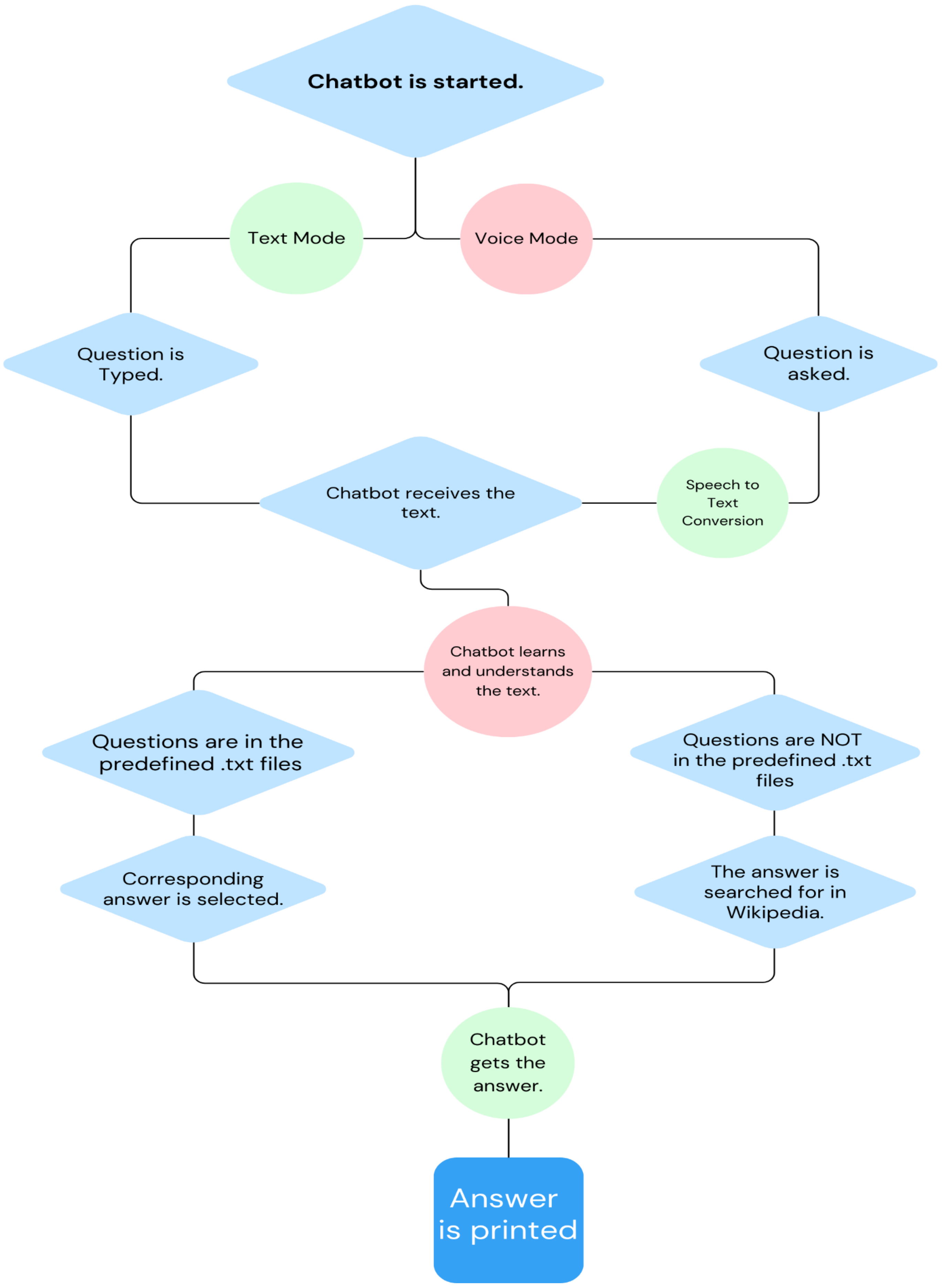
3. Main Function:

- Create an instance of the chatbot.
- Initialize the chatbot with a welcome message.
- Enter a loop to interact with the user:
    - ➢ Accept user input (either text or voice).
    - ➢ Pass the user input to the chatbot for processing.
    - ➢ Display the chatbot's response.

4. Execution:

- Execute the main function when the script is run.

Flow Chart


Chatbot is started.
Text Mode
Voice Mode
Question is Typed.
Question is asked.
Chatbot receives the text.
Speech to Text Conversion
Chatbot learns and understands the text.
Questions are in the predefined .txt files
Questions are NOT in the predefined .txt files
Corresponding answer is selected.
The answer is searched for in Wikipedia.
Chatbot gets the answer.
Answer is printed

Source Code

```
import pyttsx3
import speech_recognition as sr
import wikipedia
from fuzzywuzzy import fuzz

class Chatbot:
    def __init__(self):
        self.engine = pyttsx3.init("sapi5")
        voices = self.engine.getProperty('voices')
        self.engine.setProperty('voices', voices[1].id)
        self.engine.setProperty('rate', 150)
        self.recognizer = sr.Recognizer()

    def speak(self, audio):
        self.engine.say(audio)
        print(audio)
        self.engine.runAndWait()

    def take_command(self):
        with sr.Microphone() as source:
            print("Listening...")
            self.recognizer.pause_threshold = 3
            self.recognizer.adjust_for_ambient_noise(source)
            audio = self.recognizer.listen(source, timeout=10)
            try:
                print("Recognizing...")
                query = self.recognizer.recognize_google(audio, language='en-in')
                print(f'You said: {query}\n')
                return query
            except Exception as e:
                print("Say that again please")
                return "None"

    def clean_summary(self, summary):
        cleaned_summary = summary.replace('== Content ==', '')
        return cleaned_summary

    def search_answers(self, question):
        try:
            with open(r''questions.txt', 'r') as q_file, open(r'answers.txt',
'r') as a_file:
                questions = q_file.readlines()
                answers = a_file.readlines()
                highest_match = -1
                best_answer = None
                for i, q in enumerate(questions):
                    similarity = fuzz.ratio(question.strip().lower(),
q.strip().lower())
                    if similarity > highest_match:
                        highest_match = similarity
                        best_answer = answers[i].strip()

                if highest_match >= 75:
                    return best_answer

            return None
        except FileNotFoundError:
            return None
    def chatbot_response(self, user_input):
        response = ""

        if "text" in user_input:
            response = "You are now in text mode. How can I assist you?"
            while True:
                question = input("You: ").lower()
                if question == "exit":
                    response = "Goodbye!"
                    break
                elif question == "help":
                    response = "You can ask me questions or give me commands."
                else:
                    answer = self.search_answers(question)
                    if answer:
                        response = answer
                    else:
                        try:
                            result = wikipedia.summary(question, sentences=2)
                            cleaned_result = self.clean_summary(result)
                            response = f"{cleaned_result}\n\nWikipedia Link:
{wikipedia.page(question).url}"
                        except (wikipedia.exceptions.DisambiguationError,
wikipedia.exceptions.PageError):
                            response = "I couldn't find an answer. Can you please
rephrase your question?"
                print("Chatbot:", response)

        elif "voice" in user_input:
            response = "You are now in voice mode. How can I assist you?"
            self.speak(response)
            while True:
                voice_input = self.take_command().lower()
                if "exit" in voice_input:
                    response = "Goodbye!"
                    break
                answer = self.search_answers(voice_input)
                if answer:
                    response = answer
                else:
                    try:
                        result = wikipedia.summary(voice_input, sentences=2)
                        cleaned_result = self.clean_summary(result)
                        response = f"{cleaned_result}\n\nWikipedia Link:
{wikipedia.page(voice_input).url}"
                    except (wikipedia.exceptions.DisambiguationError,
wikipedia.exceptions.PageError):
                        response = "I couldn't find an answer. Can you please
rephrase your question?"
                self.speak("Chatbot: " + response)
                print("Next question")

        return response

def main():
    chatbot = Chatbot()
    chatbot.speak("Hello. My name is CHATBOT. Please type text or voice mode to
proceed.")

    while True:
        user_input = input("You: ").lower()

        if "exit" in user_input:
            break

        response = chatbot.chatbot_response(user_input)
        print("Chatbot:", response)

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

Conclusion

In the ever-evolving landscape of artificial intelligence (AI), our project embarked on a journey into the realm of chatbot development. With a primary focus on enhancing user interactions and harnessing the power of natural language processing, our chatbot has illuminated the potential of AI-driven conversational agents.

Throughout the course of this project, we ventured into the intricacies of chatbot design, leveraging the robust capabilities of libraries like pyttsx3, speech_recognition, and fuzzywuzzy. Our chatbot gracefully transitioned between text and voice modes, seamlessly accommodating user preferences and providing an immersive experience.

The significance of our project extends beyond the lines of code and algorithmic intricacies. It has opened a window into the profound impact of AI on diverse domains. From simplifying information retrieval through Wikipedia integration to offering solutions in text or voice, our chatbot epitomizes the synergy between technology and user-centric design.

Furthermore, as AI continues to redefine the boundaries of human-machine interaction, we are reminded of the ethical considerations that accompany this progress. Our project underscores the importance of responsible AI deployment, ensuring that these intelligent agents are not just efficient but also empathetic and considerate.

In conclusion, our journey through the development of a chatbot has been a testament to the limitless potential of AI and its ability to enhance user experiences. As the field of AI continues to advance, we remain committed to exploring new horizons and harnessing technology for the betterment of society, one conversation at a time.