Wake-Word Activated Chatbot with Speech-to-Text and AI Integration

Using Python, Whisper, and OpenAI APIs

Introduction

What is the project about?

- - A wake-word-activated chatbot capable of transcribing speech, generating responses, and providing text-to-speech (TTS) feedback.

Why is it useful?

- - Hands-free interaction with an intelligent assistant.
- - Real-time query handling and spoken responses.

Key Features

- - Wake-word detection using Whisper.
- - Real-time speech-to-text transcription.
- - Natural language understanding with OpenAI's GPT model.
- - Spoken responses with Text-to-Speech integration.
- - Robust error handling and fallback mechanisms.

Technology Stack

Programming Language: Python

- - Libraries:
- - Audio Processing: Pydub, SpeechRecognition
- - Machine Learning: Whisper, Torch, OpenAI GPT-3.5

APIs:

- - OpenAI for language model responses.
- - TTS API for generating speech responses.
- - Frameworks: Click for command-line options.

Workflow Architecture

1. Audio Recording: Microphone input processed using SpeechRecognition.
2. Speech-to-Text: Whisper model transcribes recorded audio.
3. Wake-Word Detection: Filters user queries starting with a predefined wake word.
4. Language Model Response: Query sent to OpenAI API for a response.
5. Text-to-Speech (TTS): Response converted to audio using a TTS API.
6. Output: Audio played back to the user.

Code Highlights

Command-Line Options: Customizable parameters like model selection, energy threshold.

Key Functions:

- - record_audio: Captures and processes microphone input.
- - transcribe_forever: Uses Whisper to transcribe and detect wake words.
- - reply_with_tts: Generates responses and converts them to audio.
- - Error Handling: Graceful fallback for TTS or API issues.

Challenges Faced

- - Real-time Processing: Balancing transcription accuracy and speed.
- - Wake-Word Accuracy: Ensuring wake word detection doesn't produce false positives.
- - TTS Integration: Selecting a TTS API with high-quality voice output.
- - Error Handling: Managing API rate limits and connection issues.

Results

- - Successfully implemented a responsive wake-word activated chatbot.
- - High transcription accuracy using Whisper.
- - Real-time spoken responses enhance user experience.

Future Enhancements

- - Multilingual support for transcription and TTS.
- - Improved wake-word detection using machine learning models.
- - Offline capabilities with local model deployment.
- - Advanced query understanding using context-aware models.

Conclusion

- - Summarized the project’s key contributions.
- - Highlighted its potential for real-world applications, such as virtual assistants or accessibility tools.

- Thank You!
- Questions?