

USE OF PYTHON LIBRARIES IN BUILDING AI ASSISTANT

Speech Recognition and Text-to-Speech

Table of Contents

- 01 Entry Point Speech Recognition
- O2 Exit Point pyttsx3
- O3 Processing In-Between
- 04 Example Workflow



Entry Point - Speech Recognition

Speech Library Basics

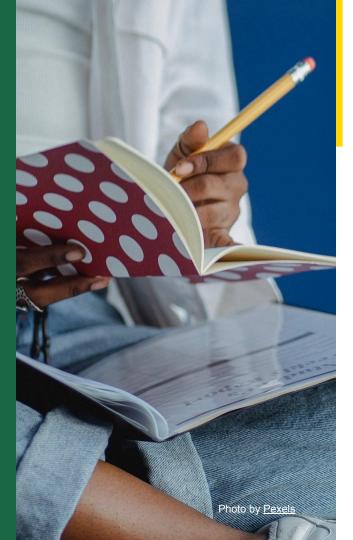
- Explore the `speech_recognition` library, providing classes and methods for speech recognition like Google Web Speech API and CMU Sphinx (offline).
- Key components include Recognizer Class with methods like recognize_google(), recognize_sphinx(), and listen() for speech recognition.
- Enhance applications with accurate speech recognition abilities,
 enabling user interaction through speech commands.



Exit Point - pyttsx3

Text-to-Speech

- Discover the `pyttsx3` library for converting text to speech effortlessly,
 offering features like voice control and speech output functionalities.
- Initiate the text-to-speech engine using init() method and manage properties like 'rate', 'volume', and 'voice' with setProperty() and getProperty().
- Enhance user experience by adding speech output functionality with say() and runAndWait() methods, providing seamless audio feedback for applications.



Processing In-Between

Text Manipulation & Enhancement

- When we recieve the string input after speech recognition, we can perform any operation of our choice.
- We can perform web search, we can do OS operations or play music on Spotify.
- The features we have implemented are- user greeting, Chrome and Spotify use, camera opening, wikipedia search and much more.



Steps in Workflow

Step 1 Step 2 Step 3 Step 4 Step 5 **Audio Input** Speech **Functions** Text-to-Speech Output Capture audio Recognition Conversion Provide the do the required input fr Convert the processing such as Convert the final speech captured audio om the user. processed text back web search. output to the to text using to speech using application user. speech_recog pyttsx3 handling, playing nition. music, etc



Future Enhancements

Advancing Voice Applications

- Implement NLP techniques to enhance text comprehension for improved responses.
- Define custom voice commands for opening apps, sending emails,
 and executing specific tasks through voice input.
- Enrich text processing by integrating advanced techniques like sentiment analysis for better interactions.
- Explore the realm of Al-powered voice applications with personalized voice commands and responses.



Documentation

//GitHub

https://github.com/KumarShresth/JARVIS_ KES_PROJECT