



USE OF PYTHON LIBRARIES IN BUILDING AI ASSISTANT

**Speech Recognition and
Text-to-Speech**

Table of Contents

- 01 Entry Point - Speech Recognition
- 02 Exit Point - pyttsx3
- 03 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 - pytttsx3



Text-to-Speech

- Discover the `pytttsx3` 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 receive 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

Audio Input

Capture audio input from the user.

Step 2

Speech Recognition

Convert the captured audio to text using `speech_recognition`.

Step 3

Functions

do the required processing such as web search, application handling, playing music, etc

Step 4

Text-to-Speech Conversion

Convert the processed text back to speech using `pyttsx3`

Step 5

Output
Provide the final speech output to the user.

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 AI-powered voice applications with personalized voice commands and responses.



Documentation

//GitHub

https://github.com/KumarShresth/JARVIS_KES_PROJECT