PRAV: Personalized Responsive AI Voice Assistant

By Veerendra Amaravathi,Sri varsha tandra

Abstract

This paper presents "PRAV," an AI-based virtual voice assistant designed using Python, NLP, and deep learning techniques. It interprets voice or text commands and performs tasks like system control, web browsing, and personalized responses. Unlike traditional assistants, PRAV operates locally with a customtrained intent recognition model.

Keywords

Voice Assistant, Natural Language Processing, Speech Recognition, Deep Learning, Personal Assistant,
Python

I. Introduction

Voice-based digital assistants are transforming how humans interact with machines. This paper introduces PRAV, a locally operating AI assistant built using open-source libraries. PRAV offers customized functionality tailored for individual users, including spoken interaction, system utility execution, and intelligent response generation.

II. Problem Statement and Applications

Problem Statement

Most existing voice assistants rely on cloud services and lack the flexibility to operate locally or be customized for user-specific intents and workflows. Additionally, they pose privacy concerns and are often inaccessible offline.

Applications

- Personal productivity assistant
- Voice-activated desktop management
- Educational support tool
- Smart home control interface
- Accessible computing for individuals with disabilities

III. Workflow Description

1. **User Input**: User gives a voice or typed command.
2. **Speech Recognition**: Converts voice to text using Google API.
3. **Intent Classification**: Predicts intent via a trained deep learning model.
4. **Response Handling**: Fetches response from JSON intents.
5. **Command Execution**: Executes system-level tasks or provides verbal feedback.

IV. Modules

1. **Speech-to-Text Engine**
2. **Intent Classification Model (Keras)**

3. **Text-to-Speech (pyttsx3)**
4. **System & Web Integration**
5. **Logging & Personalization Layer**
6. **Training Module for intents.json**

V. System Requirements

Software:

- Python 3.8+
- pip packages: TensorFlow, SpeechRecognition, pyttsx3, numpy, nltk

Hardware:

- Dual-core CPU or higher
- Microphone
- Minimum 4 GB RAM

VI. Proposed Method

The proposed system introduces a hybrid assistant model using:

- Speech-to-text (STT) conversion using Google API
- Tokenization and intent classification using Keras
- Response generation and system task execution using Python

PRAV is trained on a custom dataset `intents.json` , storing common user commands and expected responses.

VII. System Architecture

1. **Training Phase**:

2. Tokenize user command patterns
3. Label encode intent classes
4. Train using a sequential neural network model

5. Save tokenizer, label encoder, and model

6. **Prediction Phase**:

7. Load trained model and encoders

8. Accept user input (typed/voice)
9. Convert to padded sequences 10. Predict intent and respond accordingly

11. **Execution Phase**:

12. Based on the response tag, execute corresponding commands (open apps, browse websites, speak text)

VIII. Tools and Technologies

- **Python Libraries**: TensorFlow, Keras, Pyttsx3, SpeechRecognition, Psutil
- **Speech Recognition**: Google Speech API
- **Text-to-Speech**: pyttsx3
- **Command Execution**: OS, subprocess
- **NLP Components**: Tokenizer, LabelEncoder

IX. Code Module Summary

train.py

- Loads intents
- Tokenizes patterns
- Trains a deep learning model
- Saves tokenizer, encoder, and model **test.py**

- Loads model and encoders
- Accepts typed command
- Outputs response based on predicted intent

main.py

- Integrates voice recognition and TTS
- Executes web and system tasks
- Custom schedule, personalized output

X. Features

- Voice Command Processing
- Intent Classification
- Command Execution (Open/Close apps, web browsing)
- Personalized Scheduling
- System Condition Monitoring

XI. Additional Functionalities

- **Error Handling**: Handles microphone/speech failures
- **Logging**: Extendable for tracking usage and command logs
- **User Personalization**: Custom greetings, saved routines
- **Extensibility**: Easy to scale new intents and responses
- **Code Explainability**: User can request explanations of selected code snippets or programming keywords using the assistant
- **Bookmark Commands**: Save frequently used or favorite voice commands for quick access

XII. Future Scope

- GUI Interface using Tkinter or PyQt
- Multilingual Support
- Cloud Sync for Preferences
- Contextual Awareness (memory of previous queries)
- Integration with Smart Home Devices
- Continuous Learning from User Behavior

XIII. Results and Observations

The model achieves consistent accuracy over 1000 training epochs. The assistant successfully recognized and responded to voice commands under typical indoor environments with acceptable latency.

XIV. Conclusion

PRAV successfully demonstrates a lightweight, customizable AI voice assistant capable of functioning locally. With future improvements, this model can compete with mainstream assistants in controlled domains like education, home automation, or personal productivity.

XV. References

1. TensorFlow & Keras Documentation
2. Pyttsx3 Python TTS
3. Google SpeechRecognition API
4. Scikit-learn LabelEncoder
5. GitHub, StackOverflow (community solutions)
6. Custom Dataset: intents.json

Author

Amaravathi Veerendra Nath \ Gmail: vasanthi.veeru123@gmail.com
GitHub Portfolio: https://veerendrasys.github.io/p1