iROBOTROCK: A Speech Recognition Mobile Application

Reema Pimpale
Prabhat Narayan
Anand Kamath
Outline

• Introduction
• Technologies
• Current Approaches
• Our Solution
• Users (Application Domain)
• Our Approach
• Pending Functionality
• Future Enhancements
• References
Introduction

Remotely accessing machine to do tasks such as
• Play music
• Type e-mail
• Take notes
• Browse web

Motivation
To increase user friendliness by using Speech Recognition.
Technologies

- **iOS** – Operating system developed by Apple in use on their mobile devices, such as the iPhone
- **Python** – Cross platform programming language
- **CMUSphinx** – A speech recognition toolkit which has a number of packages for different tasks and applications.
- **PocketSphinx** – A package of CMUSphinx
- **SphinxBase** – A package of CMUSphinx
- **Acoustic Model** – is used by a speech recognition engine to recognize speech.
- **LMTTool** – Web based tool for generating the language model
Current Approaches

Latest technologies
• iPhone – Siri
• Android – Iris

Features supported by the above technology:
• Send text messages
• Listen to music
• Call contacts
• Send e-mail
• View a map
• Visit websites
Our Solution

• Recognize the **speech** and **connect** to your computer.

• **Open a website** on your computer by giving a **voice input** on your phone.

• Type and **send an email** from your computer using voice input from your phone.

• In short, mobile device acts as a **remote control** to **connect** and carry out tasks on your **computer**.
Users (Application Domain)

• Extremely useful for handicapped people.

• Useful for people on the go.
Our Approach

- Researched the available speech recognition libraries
- Designed the high level architecture design
- Designed the Sequence diagram to understand the data flow
- Build the Application for the iPhone (Client Application)
- Build the server Application
- Test the application
• **Client side** - Uses the open source components of CMUSphinx to provide very complicated voice-recognition functionality.

• **Server side** – Uses Python to dynamically load additional classes at run-time and allow enhanced functionality without requiring future users to touch any previously developed code.
Researched the available speech recognition libraries

Designed the high-level architecture design

Designed the Sequence diagram to understand the data flow

Build the Application for the iPhone (Client Application)

Build the server Application

Test the application
Researched the available speech recognition libraries
Designing the high level architecture design
Designed the Sequence diagram to understand the data flow
Build the Application for the iPhone (Client Application)
Build the server Application
Test the application

User Interface       audioRecorder       pocketsphinx       dictionary       server       email server

1: voice i/p
2: audio file
3: fetch word request
4: respond query
5: word displayed
6: send string to server
7: send string to email client
8: ack
9: ack
Uses the **CMUSphinx** toolkit and implement the architecture for the client.

The main components are

- **PocketSphinx**: lightweight recognizer library written in C.

- **Sphinxbase**: support library required by Pocketsphinx. (Part of the PocketSphinx component).

- **Dictionary**: (Language Model and Acoustic model)
• Written in **Python**
• Basic **TCP server** that listens for messages from client on an assigned port
• Waits for keywords to activate specific functionality
• Ex. "**MESSAGE**" triggers e-mail composition component
  – All subsequent messages received will be appended to a message string
  – Waits for "**BYE**" keyword to end message and send email
Live Demonstration

- Researched the available speech recognition libraries
- Designing the high level architecture design
- Designed the Sequence diagram to understand the data flow
- Build the Application for the iPhone (Client Application)
- Build the server Application
- Test the application
Pending Functionality

• **Dynamically** load additional components enabled on the server.

• New **component** will be written as a Python class with a defined set of **callbacks**.

• Each class defines **trigger messages** and defines how following messages should be handled to perform any required task
Future Enhancements

• Currently uses unencrypted TCP connection.

• **Security** can be enabled using **TLS** (Transport Layer Security) to encrypt messages.

• **Plugin based** system can be **extended** in any way that the user desires.
RESULT

Successfully implemented a mobile application using speech recognition which connects to the computer remotely and is able to send an email via the voice input given through the phone.
References


Thank You

Questions?