HERA - AN OPERATING SYSTEM LEVEL VOICE RECOGNITION PACKAGE

PROJECT DESIGN REPORT

submitted in partial fulfilment of the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

of

APJ Abdul Kalam Technological University

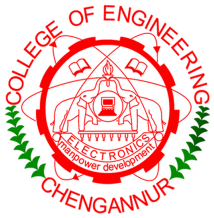
by

ARJUN VISHNU VARMA (CHN18CS027)

GOKUL MANOHAR (CHN18CS045)

JITHIN JAMES (CHN18CS066)

NANDU CHANDRAN (CHN18CS086)



DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ENGINEERING CHENGANNUR

Kerala

January 2022

COLLEGE OF ENGINEERING CHENGANNUR
CHENGANNUR - 689121

DEPARTMENT OF COMPUTER ENGINEERING

Certificate

This is to certify that this Design Project report entitled Hera - An Operating System Level Voice Recognition Package is a bonafide record of the design project presented by **Arjun Vishnu Varma, Gokul Manohar, Jithin James & Nandu Chandran** under our guidance towards the partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering of the KTU.

Co-Ordinator **Guide** **Head of the Department**

Abstract

Hera is an Operating System level voice recognition package that understands voice commands and moulds itself to suit the user's workflow. We propose a modernistic way of interacting with the system, where the latency of conventional physical inputs are minimized through speech recognition. Currently, voice assistant rely heavily on a primary server to perform most of the computation. Though this results in faster processing, this is still a privacy concern because we are not explicitly known how our data is used. Contrary to this, data processing in Hera is done locally, on the client-side. The voice data that is gathered, is sent to a speech recognition engine, and a natural language processing model is able to draw meaningful conclusions from it. Along with this, Hera is also capable of identifying usage patterns and is able to take intelligent decisions. The more the user interacts with Hera, the better it gets.

Acknowledgement

We are greatly indebted to God Almighty for being the guiding light throughout with his abundant grace and blessings that strengthened us to do this endeavour with confidence. I express our heartfelt gratitude towards our guide Ahammed Siraj K K. Also I express our sincere gratitude towards Dr. Smitha Dharan, Principal, College of Engineering Chengannur for extending all the facilities required for doing our project. We would also like to thank Dr. Manju S Nair, Head, Department of Computer Engineering, for providing constant support, encouragement and guiding us in doing this project. Now we extend our sincere thanks to our project co-ordinators Ms. Shiny B, Assistant Professor in Computer Engineering, and Ms. Sreelekshmi K R, Assistant Professor in Computer Engineering for guiding us in our work and providing timely advice and valuable suggestions. Last, but not the least, we extend our heartfelt gratitude to our parents and friends for their support and assistance.

Contents

Chapter 1

Introduction

Our project propose a new way of interacting with the operating system that prioritizes on improving the user experience via voice commands. It is able to recognize the spoken language and is able to draw meaningful conclusions from it and to provide responses accordingly. Unlike the traditional approach which rely heavily on the physical inputs, our proposed system can provide an alternative method through the means of voice interactions. Though we are developing a voice based system, the traditional physical input is still available, so the user can experience the best of both worlds.

1.1 Problem Statement

Enhancing the user experience of Linux Operating System by adding voice recognition.

1.2 Proposed Solution

For effective and efficient embedding of speech recognition into Linux Operating System, we employ a multimodule approach, namely Assistant, Coordinator and Skill modules. These modules determine how the voice data is collected, processed and evaluated. The entire working of the system is divided into two phases, Assistant-Coordinator (Primary) phase and Coordinator-Skill-Synthesis (Secondary) phase. The primary phase consist of transcribing the voice data to the corresponding intents. The secondary phase deals with mapping intents into corresponding skills and providing feedback in the form of speech or raw data.

Chapter 2

Report of Preparatory Work

During the research for our project, we came across some projects with similar ideas. They are as follows,

2.1 Airocorp

Airocorp is building the world's first Artificial Intelligence Operating System. To make our life easier and more efficient through, they are trying to convert the system as a companion, with the ability to perform tasks for us rather than being a mere utility to perform work. We can interact with our system the same way we interact with other people, through voice and gestures. Our System will learn from us, the user, which will help it grow and evolve over time, enhancing its own knowledge and unique personality.

2.2 Melissa

Melissa is a virtual assistant for OS X, Windows and Linux systems. It currently uses Google Chrome's speech-to-text engine, OS X's say command, Linux's espeak command or Ivona TTS along with some magical scripting which makes Melissa alive.

2.3 Jarvis

Jarvis is a simple command-line personal assistant for Linux, macOS, and Windows. If you enable his voice, he will be able to communicate with you. He can tell you the weather and locate restaurants and other locations near you. He can do some amazing things for you.

2.4 Monkey says, Monkey Does-Security and Privacy on Voice Assistants

[1] The introduction of smart mobile devices has radically redesigned user interaction, as these devices are equipped with numerous sensors, making applications context-aware. To further improve user experience, most mobile operating systems and service providers are gradually shipping smart devices with voice controlled intelligent personal assistants, reaching a new level of human and technology convergence.

2.5 Personal Assistant with Voice Recognition Intelligence

[2] The Most famous application of iPhone is "SIRI" which helps the end user to communicate end user mobile with voice, and it also responds to the voice commands of the user. Same kind of application is also developed by the Google that is "Google Voice Search" which is used for in Android Phones. But this Application mostly works with Internet Connections. But our Proposed System has capability to work with and without Internet Connectivity. It is named as Personal Assistant with Voice Recognition Intelligence, which takes the user input in form of voice or text and process it and returns the output in various forms like action to be performed, or the search result is dictated to the end user. In addition, this proposed system can change the way of interactions between end user and the mobile devices. The system is being designed in such a way that all the services provided by the mobile devices are accessible by the end user on the user's voice commands.

Chapter 3

Design

We employ a multimodule approach, namely Assistant, Coordinator and Skill modules. The entire working of the system is divided into two phases, Assistant-Coordinator (Primary) phase and Coordinator-Skill-Synthesis (Secondary) phase. The primary phase consist of transcribing the voice data to the corresponding intents. The secondary phase deals with mapping intents into corresponding skills and providing feedback in the form of speech or raw data.

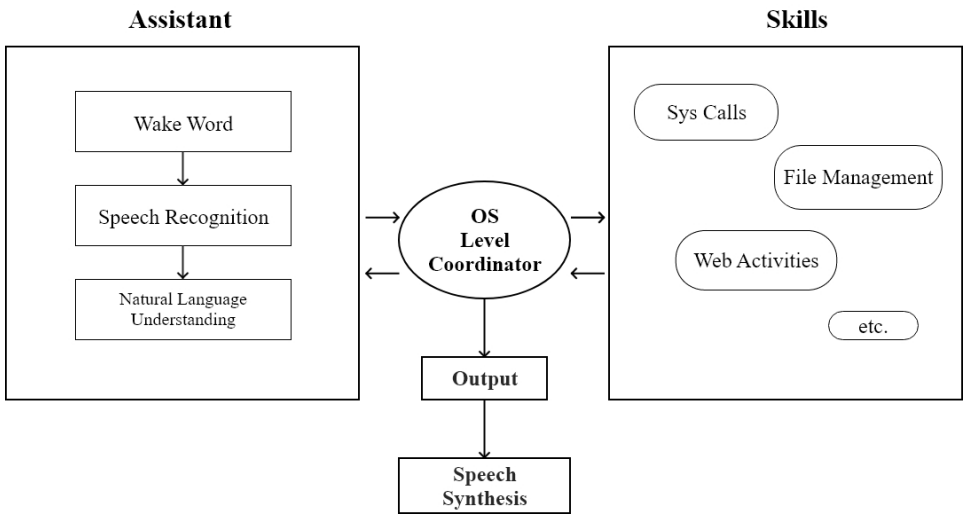


Figure 3.1: Block diagram

3.1 Assistant Module

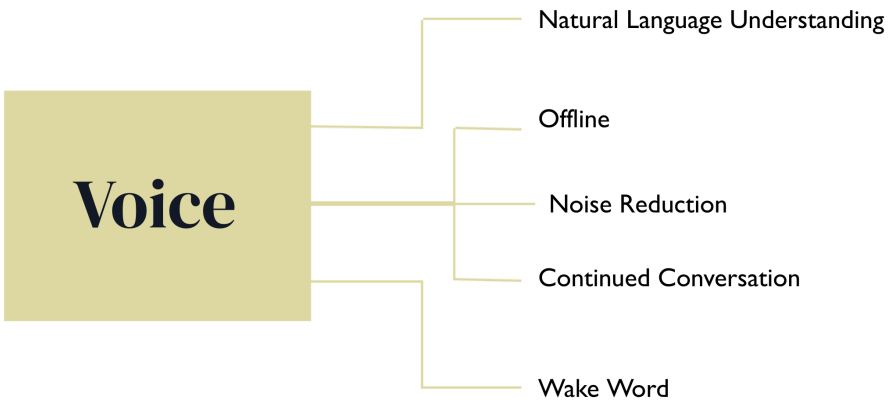


Figure 3.2: Voice Module Overview

In Assistant module, Hera is required to understand the basic 'human speech' and respond accordingly. And of course, this functionality should work offline. The way we tackle this problem is by something that can recognize the spoken voice - The spoken voice must be converted to text - Now we must look for the specific "keyword" or "skill word" (for e.g. weather is a skill) - Then we carry out a specific action as required by the user.

3.1.1 Wake Word System

We'll need to build two main components: the wake word listener and the wake word model. A wake word listener is code that constantly listens to the audio sequence and feeds that into the wake word model. This can be done in Python, a wake word model is a machine learning algorithm that takes an audio data as input and predicts if the wake word is present in that data, if the wake word is indeed present in that data then we'll wake up the Hera Server.

Our data is audio which is a naturally occurring time sequence, recurrent neural networks would be perfect for this issue, since they were invented to process sequential data we also know that the objective of the model is to predict from the audio the wake word is present in the data. We can think of this as a binary classification problem where there are only two outputs, awake were detected or awake were not.

So the model would have to learn how to classify the wake word in a sea of all other sounds, so we need to build a binary classification

recurrent neural network.

3.1.2 Offline Speech Recognition

Speech recognition would be our obvious first choice. But there is a catch to it. Most of the recognition systems that we use today require constant internet access (e.g. SpeechRecognition(Py), Google-Speech-API), so we won't be able to make the system standalone. Besides that speech recognition is computationally heavy, this would add a ton of latency and result in power drain, especially if the system is portable and runs off battery. In order to combat this problem, we will use the 'Wake Word' neural network.

To build an effective speech recognition system we have to have a strategy on how to tackle the physical properties of speech as well as the linguistic properties of it, let's start with the physical property to properly deal with the variations in nuances that comes with the physicality of speech like age, gender, microphone environmental condition etc. We'll build an acoustic model on a high level, our acoustic model will be a neural network that takes in speech waves as input and outputs to transcribe text in order for our neural network to know how to properly transcribe the speech waves to texts.

We need a neural network that can process the sequential data, the neural network also needs to be lightweight in terms of memory and compute because we want to run it real time on everyday consumer machines recurrent neural networks or RNN for short are a natural

fit for this task as it excels at processing sequential data even when we configure it to be a smaller network size, so we'll use that as our acoustic model.

To inject linguistic features into the transcriptions, we'll use something called a language model alongside a rescoring algorithm

3.1.3 NLU Model

For our implementation of a language model we can use an open source project called KenLM which is a rules-based language model. We want to use KenLM because it's lightweight and superfast, unlike the much heavier neural network based language models.

We'll use KenLM which works well enough for the rescoring algorithm. Furthermore, we'll use what's called a CTC beam search. The beam search combined with the language model is how we'll rescore the outputs for better transcriptions.

To sum it up, using a language model in the CTC beam search algorithm we can inject language information into the acoustic model's output which results in more accurate transcriptions.

3.2 Skills

Skills are a pre-programmed set of actions that Hera will perform once the intent of the user is identified. For example, skills of Hera can be Weather (To get weather information), File Management (To copy and

move files between different locations), Media Playback (Play required media files). Hera evolves as more skills are added.

3.3 OS-Level Coordinator

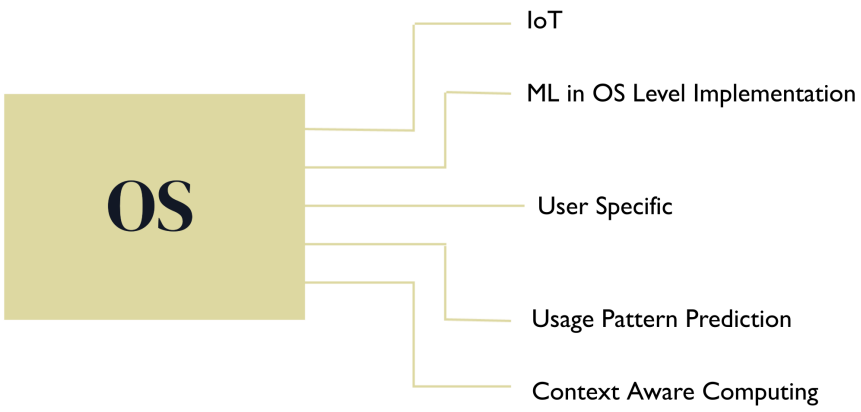


Figure 3.3: OS-Level Coordinator Overview

The Coordinating System (Hera Server) makes use of the transcribed meaning (intents) to map the requested service to the predefined skill and carry out the operation required by the user. The output is processed, and a speech synthesis system produces feedback in the form of speech or text.

3.4 Usage Pattern Prediction

Many computer users operate according to a daily routine. Taken together, over time, this use can be recognized as making up a pattern. To afford increased efficiency, the extent of the resources provided by an OS loaded during any particular time period may be determined by previously-established use patterns [3].

A daemon running as a superuser can hook into the regular (full-size) OS to make a list of applications that are executed by a particular processor. The daemon can populate a database with the identity of the applications, and the time of day in which they are used. Patterns can be determined by setting a lower use threshold against which actual use is compared, perhaps tailored for each application or action. Thus, by finding the usage pattern, Hera can be tailored for a particular user.

3.5 Hardware & Software Requirements

3.5.1 Hardware

- Microphone
- Speakers
- Computer System

3.5.2 Software

- Linux Operating System

- CMU Sphinx (Offline speech recognition package)
- PyQt5 (python package)
- BeautifulSoup (python package)
- Bert NLP (ML Model)
- Speech recognition (python package)
- TensorFlow (ML)
- PyTorch (ML)
- CDQA (python package)

3.6 Use Case

In our project, the two main entities are User and Hera Server. User has access over two main functionalities, i.e. Authentication and Service Request. While Hera Server has access to assistant, coordinator, skills, feedback system.

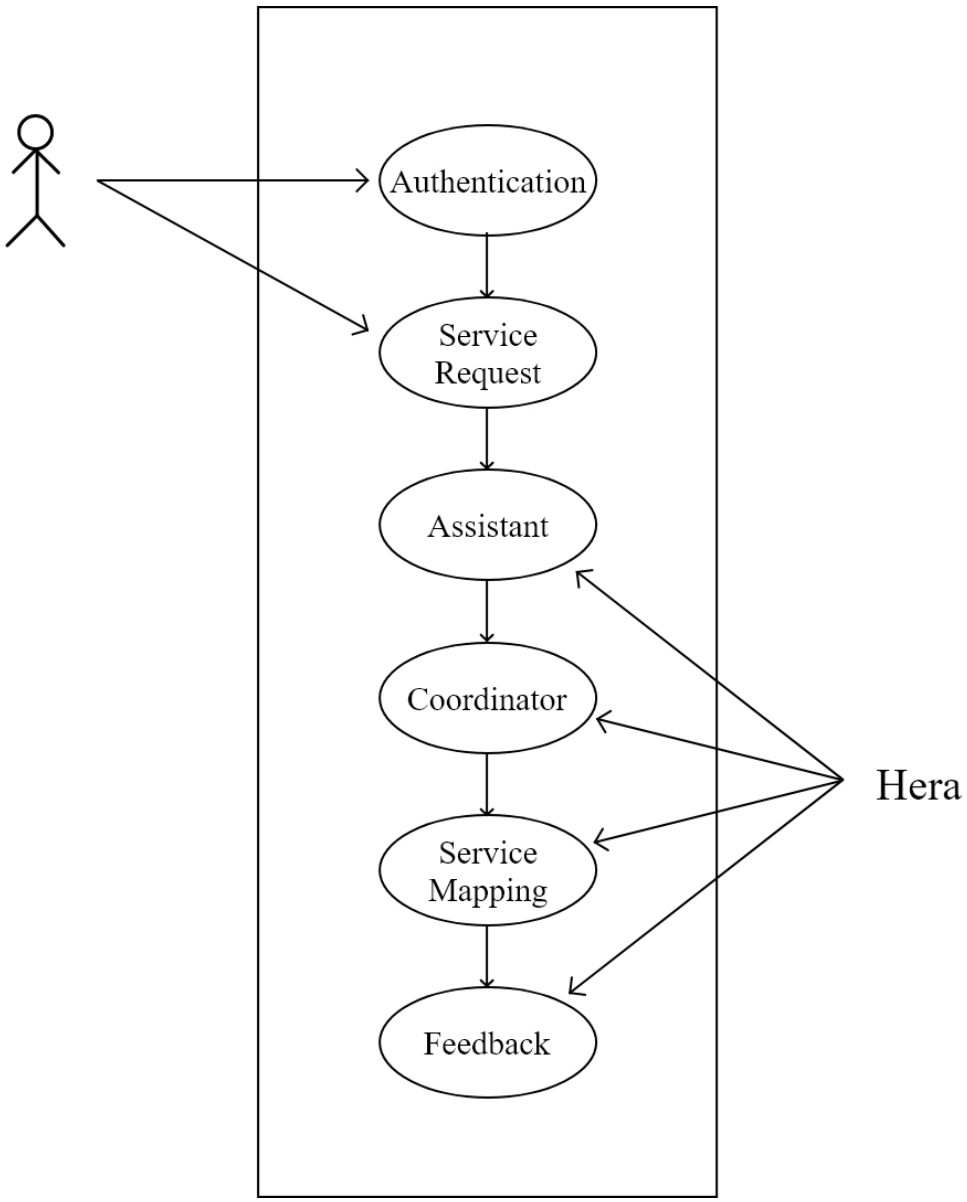


Figure 3.4: Use case diagram

3.7 Data Privacy

This is how we plan to handle the data.

1. We wish to operate mainly on offline mode. So the data rarely gets transmitted to the servers.[1]
2. The database files should also be stored locally.
3. No personal information should be sent to mainstream servers.
4. No speech data should be transmitted over the network.

3.8 Limitations

These are some limitations we identified in our project,

- Since the system works in offline mode, there will be some issues.
- Support of local language is not enabled as the training data is not available.
- The speech recognition software algorithms are not 100% accurate.
- Cyber-attack chances are still there.
- Sometimes it works slowly as AI is based on some complex calculations and predictions which will find hard to digest for some users
- Since the system self-contained, it may run into some errors

3.9 Schedule

Collect voice sample	March 25 - March 29
Implement wake word listener	March 30 - April 03
Speech recognition	April 04 - April 09
Natural Language Understanding model	April 13 - April 17
Adding skills to Hera	April 18 - April 27
Develop Coordinator module	April 28 - May 03
Testing and debugging	May 05 - May 12
$*$ Implementation on OS level	May 13 - May 18
$*$ Front end application	May 13 - May 24

$*$ Add-ons to Hera (if time permits).

Chapter 4

Conclusions

In the project preliminary phase, we decided on the way which our project is planned. Modern systems require large servers for speech recognition, and the voice data is sent to those systems for processing. Though these systems are faster in processing the data, privacy is still a major concern. Keeping this in mind, we have decided to create a package called Hera that helps in eliminating all the hassles we discussed above to some extent. Using this package, we aim to reduce the time latency in using the traditional OS. In the era of modern assistants, we hope our project Hera stands out by providing a secure and hassle-free user-experience.

Bibliography

[1] Efthimios Alepis and C. Patsakis. Monkey says, monkey does: Security and privacy on voice assistants. *IEEE Access*, 5:17841–17851, 2017.

[2] Tushar Bansal, Ritik Karnwal, Vishal Singh, and Hardik Bansal. Genesis-the digital assistant (python). *International Journal of Engineering Applied Sciences and Technology*, 5:644–648, 05 2020.

[3] Dhairesh Oza. Generating and automatically loading reduced operating system based on usage pattern of applications. *Field of Classification Search*, pages 1–12, 2009.