

COLLEGE OF COMPUTING AND INFORMATICS TECHNOLOGY

Eliza- A virtual assistant for the Ubuntu platform

By

Group 38

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY

A Project Proposal Submitted to the School of Computing and Informatics Technology

For the Study Leading to a Project Report in Partial Fulfillment of the

Requirements for the Award of the Degree of Bachelor of Science in Computer Science

Of Makerere University

Supervisor

Ms. Rose Nakibuule

School of Computing and Informatics Technology, Makerere University

rnakibuule@cis.mak.ac.ug, +256-414-540-628

November, 2018

GROUP MEMBERSHIP

| SNo | Names | Registration Number | Signature |
| --- | --- | --- | --- |
| 1 | Lincoln Karuhanga | 16/U/5465/PS |  |

# Table of Contents

[Table of Contents](#_22s39mdzz4ar) 3

[Definitions](#_3icqq0p8sj52) 5

[Virtual Assistants](#_dpzsqu3bx86i) 5

[Introduction](#_3ua4z47revfl) 6

[Background](#_q3xd8syn8jul) 6

[Problem Statement](#_hhhgz5r24t3k) 6

[Objectives](#_n49c18hkkrwd) 7

[Main Objective](#_14o7s1pr46kh) 7

[Specific Objectives](#_cn8tgz5pqnny) 7

[Additional Objectives](#_g9x821s66y1m) 7

[Scope](#_n1irogvcdlrb) 7

[Operating System](#_1j1evrq7dzai) 7

[Primary Scope](#_m7hvsvh5jv8c) 7

[Secondary Scope](#_mwcumx8gp6cs) 7

[Languages](#_gfeq6nujvqvi) 8

[Primary Scope](#_d74pf18gp8oz) 8

[Secondary Scope](#_p312yhmgybpl) 8

[Supported Applications](#_qqwbywtcuvp4) 8

[Primary Scope](#_y0r2jq6wbuk7) 8

[Secondary Scope](#_68gwzk5wc2on) 8

[Method of control](#_nepvx9o9ari) 8

[Primary scope](#_f780kbsh4u99) 8

[Secondary scope](#_nawjveoc3de3) 8

[Significance](#_3we2ygaa5bm1) 8

[Literature Review](#_n4zrk4a367b) 10

[Approaches to Speech Recognition on Embedded Systems](#_fcnp3xx9cnw) 10

[Previous projects](#_pa6wpvuskbu4) 10

[Methodology](#_65q247tf098d) 12

[Understand speech recognition impediments for offline systems and find appropriate solutions to each](#_3emoch7dilep) 12

[Phoneme recognition based approach](#_8d6covlm7g3f) 12

[Model generation from training data](#_w654v6quchyq) 12

[Develop automated menu access methods and approaches](#_9uoourl2kxlt) 13

[Efficient action resolution and execution based on evaluated input](#_h73lkbsb0svu) 13

[References](#_z2xlqlr6x54a) 14

[Appendix](#_rcbf77oo2b3) 15

[Project tools](#_9mx3crgvcp9o) 15

[External Tools](#_ytpyabpips8l) 15

[Roadmap](#_4p5429lhmvg3) 15

[Phase 1](#_31oyi5vidq0q) 15

[Phase 2](#_t87gpdi3zo8i) 16

# 

# Definitions

## Virtual Assistants

A virtual assistant or intelligent personal assistant is a software agent that can perform tasks or services for an individual.

# 

# Introduction

## Background

Technology exists to make our lives easier and to this end, several tools are constantly being developed and experimented with. A key part of this ecosystem is variety. Variety gives consumers a chance to decide what works for them out of the range of available products. After decades of dominance on the personal computer scene, the Windows Operating System is increasingly seeing more competition from the previously little known Ubuntu Operating System, a Linux Operating System distribution.

A key factor in this growth has been the increasing availability of matching and often better and more open, decentralized tools that offer functionality at par with that offered by the Microsoft Operating System.

An increasingly useful tool on our smart devices is the Virtual Assistant which the Ubuntu Operating System conspicuously lacks.

The purpose of the project is therefore to start work on an open-source, community driven, open and transparent virtual assistant primarily targeting the Ubuntu Operating System, with the goal of further adding value to the platform and offering consumers variety.

## Problem Statement

Users of the increasingly popular Ubuntu Operating System find that the platform sometimes lacks convenient tools common with other, often proprietary Operating Systems. A key tool missing on the platform is a Virtual Assistant, a version of which all competing proprietary operating systems have. It is important that an open source alternative be available for a tool that is gradually playing a bigger role in how people interact with their devices, to allow transparency, inclusion, variety and collaboration in the design decisions and data collected by such a tool.

## Objectives

### Main Objective

To develop a proper structure for and build a Virtual Assistant Primarily targeting the Ubuntu Operating System.

### Specific Objectives

* Efficient action resolution and execution based on evaluated input.
* Develop automated menu access methods and approaches generally applicable to all applications in the Gnome desktop environment.
* Understand speech recognition impediments for offline systems and find appropriate solutions to each, ultimately implementing a scalable approach.

### Additional Objectives

* Structure project in a way that allows multiple contributors to gradually add multiple application support
* Understand why previous projects have not achieved widespread use and remedy these issues

## Scope

### Operating System

#### Primary Scope

Ubuntu

#### Secondary Scope

Based on the success and applicability of the implementation approach chosen, this may expand to include other Linux Distributions.

### Languages

#### Primary Scope

English

#### Secondary Scope

Based on the approach to speech recognition chosen, other languages will subsequently be added, based on the community’s decision as well as ease of application of the knowledge and methods developed for English.Other languages based on phoneme dictionary creation

### Supported Applications

#### Primary Scope

Common system tasks

Popular Utility Applications

#### Secondary Scope

Additional applications will be added on demand and community contribution basis

### Method of control

#### Primary scope

Command based action triggers

#### Secondary scope

Natural language processing based techniques to action resolution and trigger

## Significance

This project will explore existing research into natural language processing and speech recognition on embedded and primarily offline systems and attempt to consolidate and layout the best performing strategies currently available. Most speech recognition and language processing approaches are currently dependent on online networks of connected processing power that renders using the same approaches on small devices with limited computing power infeasible.

Exploration and publication of the possible performance benefits of using a restricted language or search set on the best performing and widely available speech recognition approaches.

Build for community contributions- the project will be structured in a modular manner, increasing complexity with time, in order to allow for significant contribution from the Ubuntu open source community, as well as constant evaluation of the chosen methods to ensure parity with the best methods currently available.

Phoneme dictionary generation and update. One of the approaches to speech recognition taken will require work with language phoneme dictionaries. Efficient generation of correct and dependable phonemes to required word sets will also be a major area of exploration and findings will be promptly published.

Evaluating and documenting performance, adaptability, ease of use and customization of present speech recognition tools and libraries, including but not necessarily limited to kaldi and pocket-sphinx.

# 

# Literature Review

## Approaches to Speech Recognition on Embedded Systems

There are several approaches that have been experimented with when doing speech recognition under these conditions. The most promising though are Phoneme based approaches and Data collection and model training based approaches.

The most commonly used speech recognition engines currently use neural networks and one of the goals of the project will be to assess the feasibility of using this approach on an offline system, and whether enough data will be present to allow this approach.

The project takes an in depth look into two of the most commonly used open source speech recognition toolkits- CMU-sphinx and Kaldi. CMU-sphinx is a continuous-speech, speaker-independent recognition system making use of hidden Markov acoustic models (HMMs) and an n-gram statistical language model. Kaldi

More recently, progress has been made with offline speech recognition at better than real-time performance levels to warrant to make the approach feasible for general application. In particular the a paper by Ian McGraw et all[[1]](#footnote-0) demonstrates this progress, having achieved a 13.5% word error rate using a quantized Long Short-Term Memory (LSTM) acoustic model trained with connectionist temporal classification (CTC) to directly predict phoneme targets. Comparatively, google’s speech recognition engine has an error rate of 8% according to result from tests by Veton Këpuska and Gamal Bohouta at the Florida Institute of Technology[[2]](#footnote-1).

## Previous projects

There have been attempts at implementing such a project before, each of which has failed to achieve widespread adoption. The most promising of these were;

* Gnome voice control[3]
* Simon[4]
* Jasper[5]
* Speech control[6]

None of these projects has achieved widespread use for multiple reasons, the most common being the limited capability for speech recognition from inaccuracy in speech recognition, taking a generic approach to application control, attempting to tackle several different issues concurrently, especially in regards to attempting the generic control of multiple applications in a similar manner, and a distinct lack of focus on a single platform.

# 

# Methodology

## Understand speech recognition impediments for offline systems and find appropriate solutions to each

The primary data collection tool will be the ODK toolkit. Details of the server and application setup access are in the appendix. The project will also make use of publicly available datasets for use while training the models specifically the Mozilla Common Voice Dataset[11].

### Phoneme recognition based approach

The phoneme based approach takes a piece of sound, cleans it to remove unnecessary parts of it, for example, background noise as well as doing further preprocessing, like normalizing amplitude, before translating the recognizable speech into a sequence of phonemes which is subsequently compared to a previously built model based on a properly defined phoneme dictionary.

An implementation using this approach will have to find a scalable way of generating phonemes that accommodate the widely different accents and speech of different users, or adapt the dictionary to a particular user. Large scale phoneme dictionary generation is often done by experts and this will involve consulting one on the best approach to follow with generating this dictionary in an efficient scalable manner. This project seeks to build on top of progress made by the Kaldi speech recognition toolkit as well as the CMU-Sphinx toolkit- in particular pocket-sphinx from Carnegie Mellon University.

### Model generation from training data

The second approach is to simply collect speech data from multiple people, for example mentioning a set of words, cleaning the data by eliminating attributes of the data that will not be necessary for the model generation process, like volume levels, for example by minimizing background noise, normalizing amplitude and scaling the length of the sound and properly labeling each piece of input. Part of this data is then fed into algorithms that generate a model based on this data. The generated model is then tested with the remainder of the data to ensure its performance, recall and accuracy is within expected limits.

The approach used in the initial prototype experimented with speech data from a single user and used algorithms, out of the box from the sklearn toolkit to perform this training and validation. The initial results indicate an improving accuracy rate, but more data and better data cleaning, and a better understanding of the sound data is required to improve the model’s accuracy, as well as the need to use cross validation during training. These changes will determine suitability of these algorithms for this task, in particular, the Support Vector Machine, Decision Tree and Naive Bayes based classifiers.

## Develop automated menu access methods and approaches

The latest versions of the Ubuntu operating system use the Gnome desktop environment, which had previously been abandoned for the Unity desktop environment. The challenge here is most applications do not follow conventions in development, making accessing and manually parsing application menus in a single manner for all applications infeasible.

The current approach is to register specific actions for each application that will then map to a trigger word. This makes it easy to scale the project horizontally by adding more applications, while at the same time vertically, by adding functionality and actions to applications already added to the project.

## Efficient action resolution and execution based on evaluated input

Related to the previously mentioned approach, executing actions is also impeded by the same hurdles that complicate menu access.

The current approach is to trigger actions via key shortcuts for example triggering ‘ctrl+s’ when a save action is requested. This fits well into the scaling approach and is application specific, which handles the issue of these actions and commands being unique for each application.

Another approach to action resolution will be passive search and information structure build up. This will allow the application to return context specific results, for example, requesting a music track to be played will trigger a search in only areas known to have music.

# References

[1]Ian McGraw et all (10 Mar. 2016). Personalized Speech recognition on mobile devices [online], <https://arxiv.org/abs/1603.03185>. Accessed 22 Nov. 2018.

[2]Veton Këpuska and Gamal Bohouta (14 Mar. 2017). Comparing Speech Recognition Systems (Microsoft API, Google API and CMU Sphinx) [online], <https://www.researchgate.net/publication/314938892_Comparing_Speech_Recognition_Systems_Microsoft_API_Google_API_And_CMU_Sphinx>. Accessed 22 Nov. 2018.

[3]<https://wiki.gnome.org/Projects/GnomeVoiceControl>

[4]<https://simon.kde.org/>

[5]<https://jasperproject.github.io/>

[6]<https://wiki.ubuntu.com/SpeechControl>

[7]<http://kaldi-asr.org/doc/about.html>

[8]<http://publications.idiap.ch/downloads/papers/2012/Povey_ASRU2011_2011.pdf>

[9]<https://cmusphinx.github.io/wiki/>

[10]<https://cmusphinx.github.io/doc/pocketsphinx/>

[11]<https://voice.mozilla.org/>

[12]Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition [online], <https://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf>. Accessed 22 Nov. 2018.

# 

# Appendix

## Project tools

* ODK Server-<http://cs2-216003532-karuhanga-odk.appspot.com/>
* Github Repository-<https://github.com/Karuhanga/eliza>

## External Tools

* Kaldi- speech recognition
* PocketSphinx- speech recognition
* pydub- audio processing
* pyaudio- audio device access
* sklearn- a bit of machine learning

## Roadmap

### Phase 1

* Setup ODK collection server and endpoints
* Figure out a way of finding out which program is currently running
* Evaluation of speech recognition alternatives (will make use of Kaldi for the more mature parts and PocketSphinx and sklearn for quick demos)
* Setup Kaldi
* Figure out a way of generating language phonemes that represent our accent
* Demo keyboard control and automation
* Solve compatibility problems between the keyboard and pyaudio tools
* Demo sound based routine activation (this will be done by sound levels e.g clapping at this point) on select applications
* Collect keyword data based on single user (Using odk)
* Perform data cleaning and normalisation
  + Normalise amplitude(related to volume)
  + Remove leading and trailing silences
  + Normalise length
* Build speech recognition model for hotwords based on one user
* Demo hotword recognition, based on single user
* Fully automate keyboard input to allow us use hotkeys and shortcuts
* Integrate hotkey activation to hotword triggers
* Complete v1 build with three application hotword controls

### Phase 2

* Expand range of words that can be recognised by model
* Remove single user dependency- start collecting hotword data from multiple individuals
* Improve data collection methods- make these more professional and increase scale
* Improve data cleaning methodology
* Train model on new acquired data set
* Expand targeted applications
* Evaluate possibility of shifting from a control based system to a more intelligent assistant
* Passive search and trigger readiness- look into possibility of expanding scope to continuously scan the pc and build a structure that allows efficient search based on voice input/ grep it

1. "Personalized Speech recognition on mobile devices." 10 Mar. 2016, <https://arxiv.org/abs/1603.03185>. Accessed 22 Nov. 2018. [↑](#footnote-ref-0)
2. "(PDF) Comparing Speech Recognition Systems (Microsoft API ...." 14 Mar. 2017, <https://www.researchgate.net/publication/314938892_Comparing_Speech_Recognition_Systems_Microsoft_API_Google_API_And_CMU_Sphinx>. Accessed 22 Nov. 2018. [↑](#footnote-ref-1)