CHAPTER I

Introduction and Background of the Study

Introduction

As user demand scales for Intelligent Personal Assistants such as Apple’s Siri, Google’s Google Now, and Microsoft’s Cortana, humans are approaching the computational limits of current datacenter architectures. It is an open question how future server architectures should evolve to enable this emerging class of applications, and the lack of an Open-Source Intelligent Personal Assistant Development Platform workload is an obstacle in addressing this question.

Artificial Intelligence, as a subfield of computer science and human computer interaction may be provided via Natural Language Processing in order to combine human learning and machine reasoning (Goksel-Canbek & Mutlu, 2016). Researchers and Developers certainly believe that AI used to a certain level may offer a chance for humans to communicate without any restrictions. Natural Language Processing is the way for computers to analyze, understand, and derive meaning from human language in a smart and useful technique. (Kiser, 2016)

An Intelligent Personal Assistant is a software agent that can perform tasks or services for an individual. Many implementations of IPAs exist today and companies such as Apple, Google and Microsoft all have their implementations as a major feature in their operating systems and devices (Bahceci, 2016). The name, Intelligent Personal Assistant, is reflected to its three key defining features. First, is Intelligent, which conveys the conversational user interface that simulates an understanding of natural language and directly responds to the user with answers or actions. Second, is Personal which conveys that unlike search, these products adapt to the user in new and profound ways. Lastly, Assistant which conveys that it aims to offload routine tasks. (Buzzanga, 2015)

The actual challenge for Intelligent Personal Assistants is the process of recording, process of understanding, and effectively responding to the human speech. Given that it’s easy for most four-year-olds to understand a language, decoding human speech is a remarkably complex task. In fact, it requires more than a hundred times as much processing power when decoding a verbal request, as responding to a textual search request. (Lee, 2016)

This study focuses on the problems that the Intelligent Personal Assistants face today and can also be a problem in the future. The lack of modularity and flexibility of an IPA is one of the problem that the researchers found. Addressed that most of the IPAs today provide SDKs or Software Development Kit for the developers to use, this is a dilemma that needs to be given attention. The incorrect output when an IPA doesn’t understand the exact intent of a speech input, is also a huge problem. Given that wrong input was processed, therefore, it is highly possible that a wrong output will be produced. Also, the lack of useful clues and hints of an IPA when it gives feedbacks once the system detects an error due to not understanding the speech input of the user.

After thorough analyzation, the researchers designed an open end-to-end IPA Development Platform that can be used to build, code and extend a user’s own customized IPA with features such as Speech to Text Processing, Natural Language Processing, Text to Speech and, API and Web Services Compatibility. This is done by grouping the three main components and combining them to develop a core which will act as the central processing module and message bus. External and Add-On modules and services will be then incorporated by accessing the Platform's API. The General Architecture is illustrated in Appendix B. The platform is highly portable and can run on Linux systems from single board computers to desktops. It provides an API which allows developers to build their own Skills to access web services, control IoT devices, interact with media systems and more. The researchers aim to give developers and enthusiast alike, a platform that will enable them to build, customize and redistribute their own Intelligent Personal Assistant.

Significance of the Study

Intelligent Personal Assistant is the future of computing and further development of technologies in this area will surely be the trend for the future. It was one of the strongest fields of development during 2015 and will certainly be one of the sectors generating most profits for years or decades to come.

Smartphones are a great example of the application of artificial intelligence. In utilities like predicting what a user is going to type and correcting human errors in spelling, machine intelligence is at work. Artificial intelligence can be utilized in carrying out repetitive and time-consuming tasks efficiently.

This study seeks to benefit the following groups of people:

1. To the computer and smartphone users, to make computing a more interesting and worthwhile experience through the use and utilization of an A.I. Agent for automation and to ease them of the burden of executing certain computing tasks.

2. To the programmers and developers, this study will encourage them to partake in the improvement of Artificial Intelligence and to advance and develop applications and system that utilizes this technology.

3. To business owners and planners, A.I. is the current trend of today’s technology. It exists now in almost every device. This study can help them to take their business ideas to the next level through the use of A.I. agents to cater customer’s needs.

4. Lastly, to the future researchers, that they will further this study through investing more time in finding out the effectiveness and relevance and to what extent can A.I. help in today’s computing technologies.

Statement of the Problem

1. Most intelligent Personal Assistant Platform focuses on extensibility, but lacks in modularity and flexibility.

2. Incorrect output when the exact intent of speech input is not obtained correctly because of language ambiguity.

3. Lack of useful clues and hints in giving feedbacks once an error occurs due to not understanding the speech input.

Objectives of the Study

General Objective:

To develop an integrated Intelligent Personal Assistant Development Platform that is openly accessible to the public by using Google Speech API, Adapt – Intent Parser base on Naïve Implementation and Greedy Algorithm and MIMIC base on CMU's Flite for its main components.

Specific Objectives:

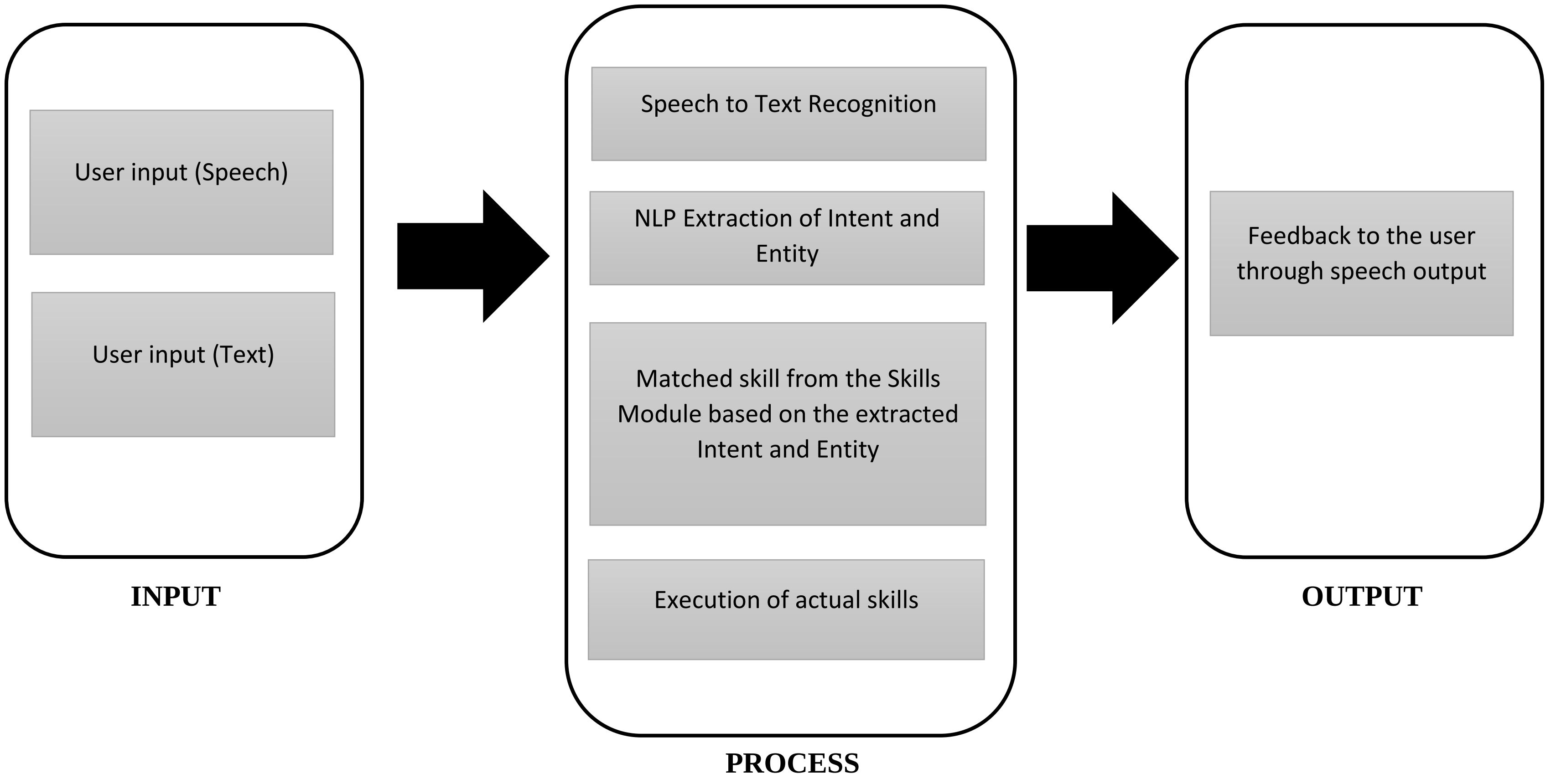
1. To provide an Intelligent Personal Assistant Platform that focuses on modularity and flexibility by making its three main components as modular for users to be able to change, extend and integrate its components.

2. To provide more accurate output by using the Adapt intent parser which has the capability to extract and parse the intent of the speech data, despite of language ambiguity.

3. To be able to give feedback with clues and hints on what a user should do when an error occurs due to misunderstanding of speech input by assigning a context specific fallback.

.

Conceptual Framework


User input (Speech)
User input (Text)
INPUT
Speech to Text Recognition
NLP Extraction of Intent and Entity
Matched skill from the Skills Module based on the extracted Intent and Entity
Execution of actual skills
PROCESS
Feedback to the user through speech output
OUTPUT

Technical Background

Hardware and Peripherals

The following are the hardware/peripherals that is used on the development of this platform. Similar hardware/peripherals may be used interchangeably.

1.) Laptop (HP Pavilion Notebook PC)

<u>Processor</u>
Intel Core i7-5500U (Intel Core i7)
<u>Graphics adapter</u>
NVIDIA GeForce 940M - 2048 MB, Core: 1072-1176 MHz, Memory: 900 MHz, DDR3, 64 bit interface, ForceWare 353.62 (10.18.13.5362), Optimus
<u>Memory</u>
12288 MB
DDR3-1600, dual-channel, two memory banks (both filled)
<u>Display</u>
15.6 inch 16:9, 1920x1080 pixel, BOE, TN LED, glossy: no
<u>Mainboard</u>
Intel Broadwell-U PCH-LP (Premium)

2.) Plug and Play External Desktop Microphone

Any generic brand of USB Desktop Microphones for audio capture.

3.) Plug and Play External Speakers

Any generic brand of 3.5mm speakers for audio output.

Software and Technologies

1. Google Speech API

Google Cloud Speech API enables developers to convert audio to text by applying powerful neural network models in an easy to use API. The API recognizes over 110 languages and variants, to support your global user base. You can transcribe the text of users dictating to an application's microphone, enable command-and-control through voice, or transcribe audio files, among many other use cases. Recognize audio uploaded in the request, and integrate with your audio storage on Google Cloud Storage, by using the same technology Google uses to power its own products.

Frame size

Streaming recognition recognizes live audio as it is captured from a microphone or other audio source. The audio stream is split into frames and sent in consecutive **StreamingRecognizeRequest** messages. Any frame size is acceptable. Larger frames are more efficient, but add latency. A 100-millisecond frame size is recommended as a good tradeoff between latency and efficiency.

Audio pre-processing

It's best to provide audio that is as clean as possible by using a good quality and well-positioned microphone. However, applying noise-reduction signal processing to the audio before sending it to the service typically reduces recognition accuracy. The service is designed to handle noisy audio.

Below is an example of speech to text transcription of Google Speech API:

- The audio level should be calibrated so that the input signal does not clip, and peak speech audio levels reach approximately -20 to -10 dBFS.
- The device should exhibit approximately "flat" amplitude versus frequency characteristics (+- 3 dB 100 Hz to 8000 Hz).
- Total harmonic distortion should be less than 1% from 100 Hz to 8000 Hz at 90 dB SPL input level.

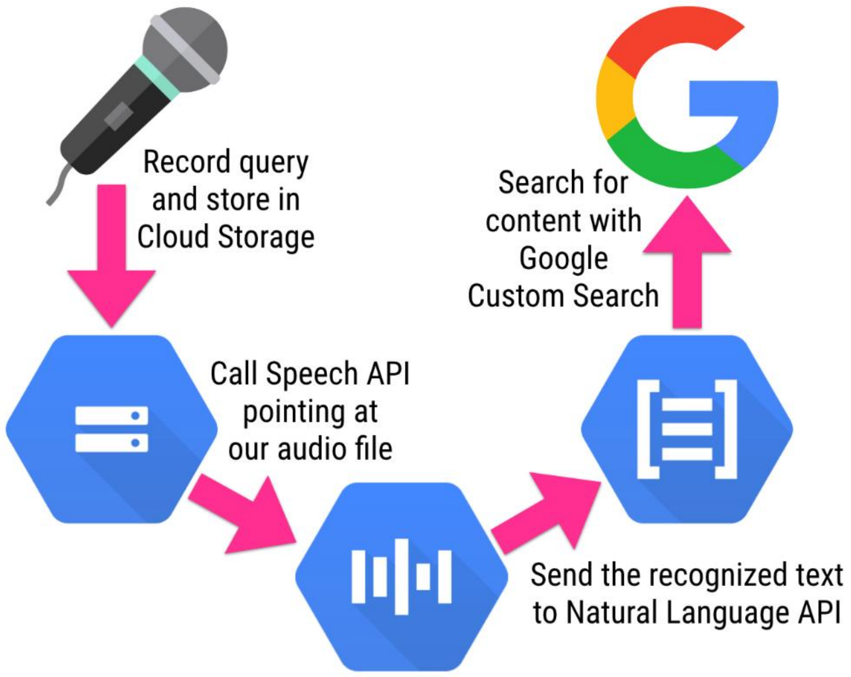

Record query and store in Cloud Storage
Call Speech API pointing at our audio file
Send the recognized text to Natural Language API
Search for content with Google Custom Search

Fig 1.0 Google Speech API

2. Adapt Pattern Based Text Intent and Entity Extraction

The **Adapt Intent Parser** is a flexible and extensible intent definition and determination framework. It is intended to parse natural language text into a structured intent that can then be invoked programatically. It is an **open source software library** for converting natural language into machine readable data structures.

Adapt is lightweight and streamlined and is designed to run on devices with limited computing resources. Adapt takes in natural language and outputs a data structure that includes the intent, a match probability, a tagged list of entities.

Adapt is a rules-based artificial intelligence library that is useful for interpreting natural language input. For example, a user might want to create a natural language interface that allows them to play a Pandora station. The user might say "Turn on Pandora", or "Play Pandora", or "Put on my Joan Jett Pandora station."

Adapt has many different parts to it, but the core engine uses a greedy and naive implementation for intent determination and domain determination.

Naive Implementation just means that it uses a brute force approach to find the best intent.

An algorithm is said to be naive when it is simple and straightforward but does not exhibit a desirable level of efficiency (usually in terms of time, but also possibly memory) despite finding a correct solution or it does not find an optimal solution to an optimization problem. Naive algorithms are easy to discover, often easy to prove correct, and often immediately obvious to the problem solver. They are often based on simple simulation or on brute force generation of candidate solutions with little or no attempt at optimization.

Naive implementation pattern matching

- No pre-processing is done
- Comparison is done in left to right order. On the characters in two arrays
- When mismatch occurs, pattern is moved one position to the right with respect to the text.

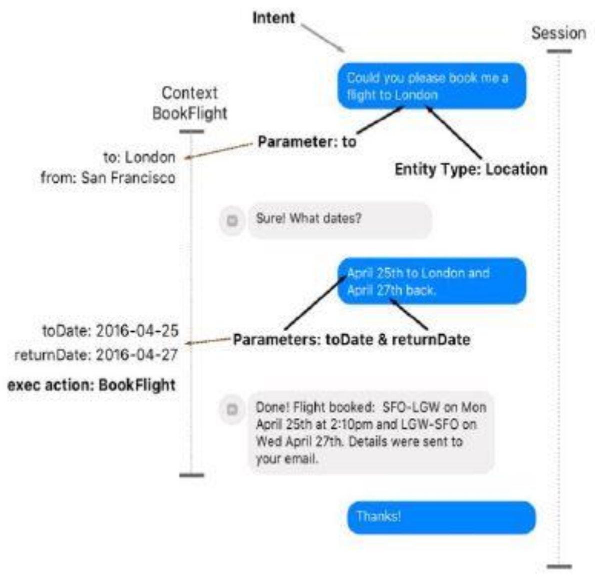
Few applications

- In text editing programs where a particular word is supplied by the user and is matched in a given text using pattern matching process
- In DNA Sequence to match the strands of DNA of two organisms and find how similar the two organisms are.

Greedy Algorithm means it only looks at the choices locally (or closest to it) and then chooses the best choice it sees.

A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment. This means that it makes a locally-optimal choice in the hope that this choice will lead to a globally-optimal solution.

Overview of Adapt Intent and Entity Extraction Algorithm:



Could you please book me a flight to London

Sure! What dates?

April 25th to London and April 27th back.

Done! Flight booked: SFO-LGW on Mon April 25th at 2:10pm and LGW-SFO on Wed April 27th. Details were sent to your email.

Thanks!

Fig 2.0 Intent and Entity Extraction

Input acceptance

1. Natural Language Text Input

Fig 3.0 Text Input

Text analysis

1. Tokenize text to separate segments

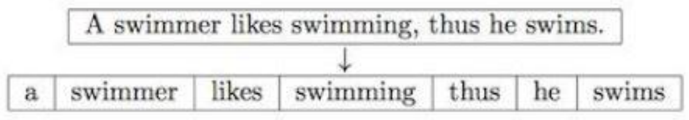


Fig 4.0 Text Tokenization

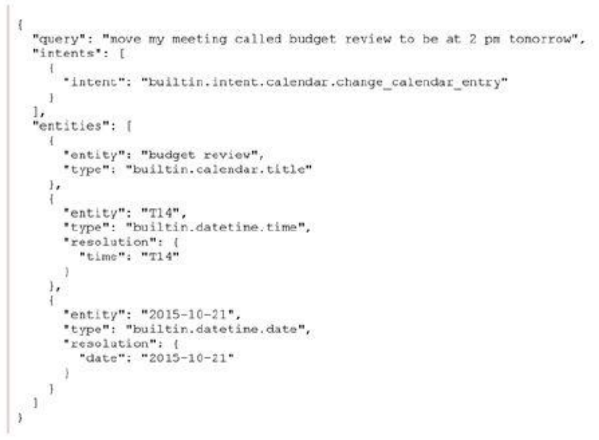
Intent recognition

1. Identify and extract entities based from provided pattern

2. Match text input to a defined pattern in database and recognize intent

Export to machine readable format

1. Intent and Entities will be formatted and exported in a machine-readable format such as XML or JSON



```
{
  "query": "move my meeting called budget review to be at 2 pm tomorrow",
  "intents": [
    {
      "intent": "builtin.intent.calendar.change_calendar_entry"
    }
  ],
  "entities": [
    {
      "entity": "budget review",
      "type": "builtin.calendar.title"
    },
    {
      "entity": "T14",
      "type": "builtin.datetime.time",
      "resolution": {
        "time": "T14"
      }
    },
    {
      "entity": "2015-10-21",
      "type": "builtin.datetime.date",
      "resolution": {
        "date": "2015-10-21"
      }
    }
  ]
}
```

Fig 5.0 Exporting to machine readable format

3. Mimic Text to Speech Synthesis

Mimic is a fast, lightweight Text-to-speech engine based on Carnegie Mellon University's FLITE software. Mimic takes in text and reads it out loud to create a high-quality voice. Mimic's low-latency, small resource footprint, and good quality voices set it apart from other open source text-to-speech projects.

Mimic is a powerful tool that can also help solve other important problems. Mimic works on Linux, Android & Windows and we are working on iOS support. We are also adding more languages to enable many people to access realistic voices for the first time.

It is a speech synthesizer algorithm that uses a formant synthesis method, providing many languages in a small size. In addition, it can be used as a front-end, providing text-to-phoneme translation.

Mimic Text to Speech Synthesis Algorithm:

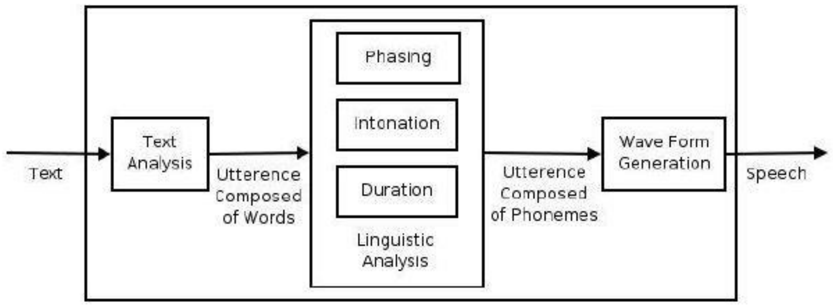


Fig 6.0 eSpeak Speech Synthesis

Text analysis

1. Text and symbols will be tokenized

2. After tokenization, words will be syllabicated and symbols will be evaluated to extract the phonemes

3. Phonemes will be given indices to allow programmability and matching

Phoneme matching

1. Phonemes extracted from the text will be individually matched to existing sinusoidal phones in the database

Phone concatenation

1. Matched phones will then be concatenated to form the sound synthesis

Programming Language

Python

Python is a widely used high-level programming language for general-purpose programming. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The design of Python offers some support for functional programming in the Lisp tradition. The language has filter(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

Software Packages and Libraries:

The following packages are required for setting up the development environment:

git	**openssl**
python 2	**autoconf**
python-setuptools	**bison**
python-virtualenv	**swig**
pygobject	**glib2.0**
virtualenvwrapper	**s3cmd**
libtool	**portaudio19**
libffi	**mpg123**

The above-mentioned software packages are all specifically for Linux Based Operating System.

Operating System (Linux Based)

Ubuntu 17.04 (Zesty Zapus)

Ubuntu (/ʊˈbʊntuː/ uu-BUUN-too, stylized as **ubuntu**) is an open source operating system software for computers. It is one of the distribution systems of Linux, and is based on the Debian architecture. It is usually run on personal computers, and is also popular on network servers, usually running the Ubuntu Server variant, with enterprise-class features. Ubuntu runs on the most popular architectures, including Intel, AMD, and ARM-based machines. Ubuntu is also available for tablets and smartphones, with the Ubuntu Touch edition.

Ubuntu is published by Canonical Ltd, who offer commercial support. It is based on free software and named after the Southern African philosophy of ubuntu (literally, 'human-ness'), which Canonical Ltd. suggests can be loosely translated as "humanity to others" or "I am what I am because of who we all are".

Ubuntu is the most popular operating system running in hosted environments, so–called "clouds", as it is the most popular server Linux distribution.

Development of Ubuntu is led by UK-based Canonical Ltd., a company founded by South African entrepreneur Mark Shuttleworth. Canonical generates revenue through the sale of technical support and other services related to Ubuntu. The Ubuntu project is publicly committed to the principles of open-source software development; people are encouraged to use free software, study how it works, improve upon it, and distribute it.

Scope and Limitations

Scope

This study focuses on the development of a Development Platform for building Intelligent Personal Assistants based from three core components in the field of artificial intelligence, namely: Speech-to-Text, Natural Language Processing, and Text-to-Speech. With this platform, IPAs can be extended to include Machine Learning, Computer and Home Automation, Problem Solving and Queries. Software, APIs and SDKs that will be used in the development of this project are either open-source or free-to-use technologies.

Delimitations

This study will not cover or in any way will involve proprietary technologies and features of similar Intelligent Personal Assistants such as Microsoft 's Cortana, Apple's Siri, IBM's Watson, Amazon's Alexa and Google's Google Now, etc. Any similarity among these systems within this study is not intended and may be incidental.

The development platform targets to run on Linux-based Operating Systems but users that might want to use it in other Operating Systems can use and run it thru Virtualization Software such as VMware and Virtual box. Given that there might be enthusiasts alike that wants to try on Windows and MacOS.

Functions and features of this project only covers topic Artificial Intelligence, Computational Knowledge, Machine Learning and Cognitive Perceptions in the field of Computer Programming and Development and not topics directly related to Medicine, Engineering and other subject areas not mentioned or related above.

Definition of Terms

Acoustic Model - is a set of words or sentences a Speech to Text Engine can understand.

Actuator - is a Module or Code that executes a specific Intent.

Application Program Interface (A.P.I) - is a set of routines, protocols, and tools for building software applications.

Artificial Intelligence (A.I.) - is the simulation of human intelligence processes by machines.

Assistant - is the term used for the Artificial Intelligence Entity, can be interchanged to Agent.

Belief - is the knowledge base of the Agent about the world.

Desire - is the current goal of the Agent in relation to the Belief.

Entity - represent a class of object or a data type that is relevant to a user's purpose.

Environment - means the real world where the Agent takes input from.

Intelligent Personal Assistant (I.P.A.) - is a software agent that can perform tasks or services for an individual.

Intent - the code generated proposed “idea” of a natural language entry

Intention - is the method of action or the Agent will execute based from the Belief and Desire.

Internet of Things - is a proposed development of the internet in which everyday objects have network connectivity, allowing them to send and receive data.

Natural Language Processing (N.L.P.) - is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human.

Phoneme - any of the perceptually distinct units of sound in a specified language that distinguish one word from another

Phonetic Frame - is one unit of phoneme in a given word (tokenized and syllabicated)

Software Development Kit - a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform.

Speech Recognition - is the process of translating speech into human readable text

Speech Synthesis - the process of transcribing text into audible computer-generated voice