A Project Report

on

Enhanced Virtual A.I. Assistant

Submitted in partial fulfillment of the requirement for the award of the degree of

B.Tech Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision:
Mr. Dhruv Kumar
Assistant Professor

Submitted By
Anshul Jain
18021140052/18SCSE1180079
Prakhar Misra
19021180105/19SCSE1180112

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA INDIA

December, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"Enhanced Virtual A.I. Assistant"** in partial fulfillment of the requirements for the award of the degree of B.Tech Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of Mr. Dhruv Kumar Assistant Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

ANSHUL JAIN, 18SCSE1180079
PRAKHAR MISRA, 19SCSE1180112

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Dhruv Kumar
Assistant Professor

CERTIFICATE

The Final Project Viva-Voce examination of ANSHUL JAIN,18SCSE1180079 PRAKHAR MISRA,19SCSE1180112 has been held on__________________and his/her work is recommended for the award of B.Tech Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December,2021
Place: Greater Noida

Acknowledgement

In completing this project report on project titled **ENHANCED VIRTUAL ASSISTANT**, we had to take the help and guideline of a few respected people, who deserve our greatest gratitude. The completion of this project report gives me much Pleasure. We would like to show our gratitude to **Mr. Dhruv Kumar (Guide)** for giving us a good guideline for project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this project report. Many people, especially our classmates and friends themselves, have made valuable comments and suggestions on this proposal which gave us inspiration to improve our project. Here we thank all the people for their help directly and indirectly to complete this project report.

Date: December 18, 2021

Anshul Jain (18SCSE1180079)
Prakhar Misra (19SCSE1180112)

ABSTRACT

Virtual assistants are software programs that helps to ease our day-to-day tasks. Voice-based intelligent assistants need an invoking word to activate the listener, followed by the command. Personal assistants improve user productivity by managing routine tasks of the user and by providing information from an online source to the user.

This project features a live voice-based input mechanism processed Virtual Assistant, trained by various NLP and Deep Learning algorithms technique to perform various operations on the screen by voice commands by the user.

The main agenda of our virtual assistant is to give implement several tasks which are presented as features of a virtual assistant in the modern world. The voice assistance takes the voice input through the microphone and it converts the voice into computer understandable language which further gives the required solutions and answers asked for by the user. Natural Language Processing algorithm helps computer machines to engage in communication using natural human language in many forms.

Table of Contents

List of Figures

Acronyms

B.Tech.	Bachelor of Technology
SCSE	School of Computing Science and Engineering

Chapter-1

INTRODUCTION

By definition ~ “Artificial Intelligence (or AI in short) is the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.”. AI has become a popular field of study in the ever-growing cyberworld of machines and computers. The birth of the artificial intelligence is said to in "*Computing Machinery and Intelligence*" by Alan Turing, which was published in 1950. In this paper, Turing devised a question - “Can machines think?”. As to answer this question, he devised a series of tests where the machine is supposed to be ‘Intelligent’, if a human interrogator is not able to find any difference in the response of a human and the machine, famously naming it as “*The Turing Test”*.

Virtual Assistants, sometimes being referred to as Intelligent Virtual Assistant (IVA) or Intelligent Personal Assistants (IPA), is a software agent that performs tasks or provide various services to an individual based on commands or queries. Sometimes the term ‘Chatbot’ is also referred to as a virtual assistant due to its similarities. These agents are able to interpret human speech (either by voice or text input) and provide a response to the input. From being featured in movies like *HAL-9000 from 2001: A Space Odyssey* and *Jarvis, Friday in Iron Man series* to being publicly available in mobile phones, they are the most favorite subject of the modern world of Artificial Intelligence as most of the big tech companies are investing to provide more advanced virtual assistants.

1.1 Background

There already exist a number of desktop virtual assistants. A few examples of current virtual assistants available in market are discussed in this section along with the tasks they can provide Some of the Most Common Virtual Assistants are:

Siri:

Siri is still one of the best virtual assistants one could ask for. Featured all common Apple devices, Siri laid the foundation of virtual assistants. The veteran app has gotten updated numerous times, improving and adding more to what it can already do. Siri can type texts for you, search your queries on search engines, dial a contact, warn about the weather, and read headlines from the selected newspapers (many newspapers have audio reports now, such as The Washington Post). This smart virtual assistant can also act as an interpreter helping you understand different foreign languages.

Cortana:

Cortana is another great option for a virtual assistant. The ambitious virtual assistant has been around for quite a while and has picked up numerous features along the way. This virtual assistant helps keep you organized by handling your calendar. Cortana will take notes, set up reminders and alarms, including many other features you would find in virtual assistants. The best part about this virtual assistant is that it learns and can adapt a number of features. It is also compatible with a number of apps (you can ask your assistant to startup apps). You can ask your queries, and Cortana will search for it on Binge Engine.

Amazon Echo (Alexa):

Amazon’s take on Virtual Assistant, Alexa, is one of the users’ favorites. The phrase ‘Alexa…play [insert a song]’ has been very popular this year. The virtual assistant initially ran on Amazon devices. The company has now extended its support to iOS and Android devices as well. The assistant can do a range of things other than playing music. You can take notes, make To-Do lists, set up alarms, can keep you informed about the weather and news, and more. The virtual assistant comes with this exciting feature of waking up with a word of your choice (not every device offers this feature, though).

Google Assistant:

Just as Siri is to Apple; Google Assistant is powered by voice commands on Android. Most of the Android phones come with a shortcut that can take the user directly to the virtual assistant. Google Assistant types messages, emails, dial contacts, play songs, search queries, read notifications, open apps, to name a few basic functions. The Assistant supports the text commands as well. You can turn the Assistant on and then decide whether you want to continue with the voice command or type instead. Additionally, you can also control your smart home devices using the assistant.

1.2 History of the Virtual Assistant

Communication between people and machines started back in the early 1960s. The first natural language processing computer program, ELIZA, was developed by MIT professor Joseph Weizenbaum. Not much later, another advancement in digital speech recognition was produced by IBM with its Shoebox voice-activated calculator, presented to the general public during the 1962 Seattle World's Fair. The 1970s was the decade of voice recognition where companies and academia including IBM, Carnegie Mellon University, and Stanford Research Institute collaborated. The result was "Harpy," a machine that mastered about 1,000 words, with the vocabulary of a three-year-old that could understand sentences. The 1980s produced products like IBM's voice recognizing typewriter, named Tangora, after the world's fastest typist. It had a vocabulary of 20,000 words. The birth of the first virtual assistant; however, began with IBM Simon in the early 1990s. It was a digital speech recognition technology that became a feature of the personal computer with IBM and Philips. In the early 2000s, the first chatbot, familiar to most today, was technically invented by Colloquis, which launched Smarter Child on platforms like AIM and MSN Messenger. It was entirely text-based and was able to play games, check the weather, look up facts, and converse with users. It is also considered the precursor to Apple's Siri.

We have tried to build a system with similar features known as Enhanced Virtual Assistant (E.V.A). E.V.A is fully implemented with a Chatbot with an API for Robotic Process Automation (RPA) functionality. The main purpose of EVA is to provide an interface for the user to operate the entire operating system with minimal to no physical interaction for basic tasks like creating and saving a text file, using voice input to type in a text document, search for some information online, record cam videos, search web for specific information, and many more. The functionality of this Virtual Assistant can be accessed on various platforms such as Windows and some distribution of Linux. E.V.A was built to keep ease of access in mind, so as to provide a great experience for the user. The user can command E.V.A to perform actions it can and E.V.A will respond immediately to the user.

EVA can also use TTS service to revert back to the query of the user in speech mode thus making it feasible to use the data. E.V.A can also be used to provide computer functionality to physically disabled individuals who find it difficult to operate their computer. Businesses and organizations can use the functionality of E.V.A to automate multiple repeated tasks to save profits and their employees from boring tasks. E.V.A is open source that means various functionalities can be added to / removed from it easily, making it fit for any to all tasks required by the user.

The era of Smart Speakers and the Virtual Assistant

The Siri voice assistant was released as an app for iOS in February 2010. It was the first modern digital virtual assistant installed on a smartphone, introduced later as a feature of the iPhone 4S on 4 October 2011. Others followed including IBM's Watson, Microsoft's Cortana, Amazon's Alexa, Facebook’s “M", Google's G-Assistant, Alibaba's Ali Genie, and Yandex's Alice. And, thus, the era of the smart speaker for the end-consumers began. But also, behind the scenes of the general public were advancements in businesses' use of AI with virtual assistants

1.3 OBJECTIVES

Main objective of building personal assistant software (a virtual assistant) is using semantic data sources available on the web, user generated content and providing knowledge from knowledge databases. The main purpose of an intelligent virtual assistant is to answer questions that users may have. This may be done in a business environment, for example, on the business website, with a chat interface. On the mobile platform, the intelligent virtual assistant is available as a call-button operated service where a voice asks the user “What can I do for you?” and then responds to verbal input. Virtual assistants can tremendously save you time. We spend hours in online research and then making the report in our terms of understanding. JIA can do that for you. Provide a topic for research and continue with your tasks while JIA does the research. Another difficult task is to remember test dates, birthdates or anniversaries. It comes with a surprise when you enter the class and realize it is class test today. Just tell JIA in advance about your tests and she reminds you well in advance so you can prepare for the test. One of the main advantages of voice searches is their rapidity. In fact, voice is reputed to be four times faster than a written search: whereas we can write about 40 words per minute, we are capable of speaking around 150 during the same period of time15. In this respect, the ability of personal assistants to accurately recognize spoken words is a prerequisite for them to be adopted by consumers.

Chapter -2

Literature Survey

Python

Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages.

Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science etc. Python has a lot of libraries for every need of this project. For JIA, libraries used are speech recognition to recognize voice, gTTS for text to speech, selenium for web automation etc.

Python is reasonably efficient. Efficiency is usually not a problem for small examples. If your Python code is not efficient enough, a general procedure to improve it is to find out what is taking most the time, and implement just that part more efficiently in some lower-level language. This will result in much less programming and more efficient code (because you will have more time to optimize) than writing everything in a low-level language.

Speech Recognition

This is a library for performing speech recognition, with support for several engines and APIs, online and offline. It supports APIs like Google Cloud Speech API, IBM Speech to Text, Microsoft Bing Voice Recognition etc.

gTTS

gTTS is a Python library and CLI tool to interface with Google Translate's text-to-speech API. Write spoken mp3 data to a file, a file-like object (byte string) for further audio manipulation, or stdout. Or simply pre-generate Google Translate TTS request URLs to feed to an external program.

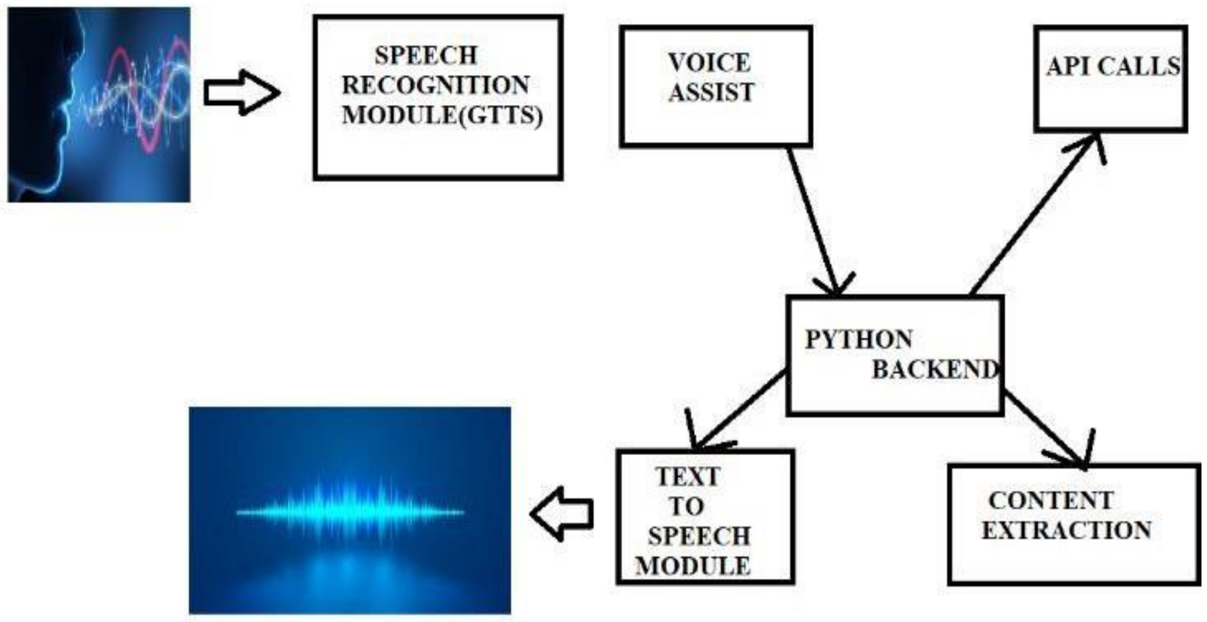
Quepy

Quepy is a python framework to transform natural language questions to queries in a database query language. It can be easily customized to different kinds of questions in natural language and database queries. So, with little coding you can build your own system for natural language access to your database.

Chapter -3

Functionality/Working of Project

Even though it has a complex mechanism, we have tried to present the working of E.V.A in a simply as it can be. Keeping in mind that every functionality of the task added to E.V.A has its own set of methodologies, it is impossible to describe each in detail. However, its core functionality has been described below, in effort to make the visualization better we have also added a uml diagram



3.1 Core Functionality

3.1.1 Speech Recognition:

E.V. A’s core input detection lies in its speech detection system. This is achieved using a python library known as *Speech Recognition.* This library is considered to be the best at performing speech from the user’s microphone and convert it into text, technically it is what’s called a Speech to Text (STT) system. The library functions over different recognition engines and API support. We will discuss them in detail later in its defined sections.

By default, the system uses Google’s online speech recognition system for converting speech input to text. The speech input Users can obtain texts from the special corpora organized on the computer network server at the information center from the microphone is temporarily stored in the system which is then sent to Google cloud for speech recognition. The equivalent text is then received and fed to the central processor.

But how exactly is a speech perceived and detected by the system? We have answer it below in detail.

Speech Structure:

Speech is a continuous audio stream where rather stable states mix with dynamically changed states. In this sequence of states, one can define more or less similar classes of sounds, or phones. Words are understood to be built of phones, but this is certainly not true. The acoustic properties of a waveform corresponding to a phone can vary greatly depending on many factors - phone context, speaker, style of speech and so on. The so-called coarticulation makes phones sound very different from their "canonical" representation. Next, since transitions between words are more informative than stable regions, developers often talk about diphones - parts of phones between two consecutive phones. Sometimes developers talk about sub phonetic units - different substates of a phone. Often three or more regions of a different nature can be found.

The number three can easily be explained: The first part of the phone depends on its preceding phone; the middle part is stable and the next part depends on the subsequent phone. That's why there are often three states in a phone selected for speech recognition.

Sometimes phones are considered in context. Such phones in context are called triphones or even quinphones. For example, "u" with left phone "b" and right phone "d" in the word "bad" sounds a bit different than the same phone "u" with left phone "b" and right phone "n" in word "ban". Please note that unlike diphones, they are matched with the same range in waveform as just phones. They just differ by name because they describe slightly different sounds.

For computational purpose it is helpful to detect parts of triphones instead of triphones as a whole, for example if you want to create a detector for the beginning of a triphone and share it across many triphones. The whole variety of sound detectors can be represented by a small amount of distinct short sound detectors. Usually, we use 4000 distinct short sound detectors to compose detectors for triphones. We call those detectors senones. A senone's dependence on context can be more complex than just the left and right context. It can be a rather complex function defined by a decision tree, or in some other ways.

Next, phones build sub word units, like syllables. Sometimes, syllables are defined as "reduction-stable entities". For instance, when speech becomes fast, phones often change, but syllables remain the same. Also, syllables are related to an intonational contour. There are other ways to build sub words - morphologically-based (in morphology-rich languages) or phonetically-based. Subwords are often used in open vocabulary speech recognition.

sub words form words. Words are important in speech recognition because they restrict combinations of phones significantly. If there are 40 phones and an average word has 7 phones, there must be 40^7 words. Luckily, even people with a rich vocabulary rarely use more then 20k words in practice, which makes recognition way more feasible.

Words and other non-linguistic sounds, which we call fillers (breath, um, uh, cough), form utterances. They are separate chunks of audio between pauses. They don't necessary match sentences, which are more semantic concepts.

On the top of this, there are dialog acts like turns, but they go beyond the purpose of this document.

Recognition Process:

The common way to recognize speech is the following: we take a waveform, split it at utterances by silences and then try to recognize what's being said in each utterance. To do that, we want to take all possible combinations of words and try to match them with the audio. We choose the best matching combination.

There are some important concepts in this matching process. First of all, it's the concept of *features*. Since the number of parameters is large, we are trying to optimize it. Numbers that are calculated from speech usually by dividing the speech into frames. Then for each frame, typically of 10 milliseconds length, we extract 39 numbers that represent the speech. That's called a *feature vector*. The way to generate the number of parameters is a subject of active investigation, but in a simple case it's a derivative from the spectrum.

Second, it's the concept of the *model*. A model describes some mathematical object that gathers common attributes of the spoken word. In practice, for an audio model of senone it is the gaussian mixture of its three states - to put it simple, it's the most probable feature vector.

The model of speech is called *Hidden Markov Model* or *HMM*. It's a generic model that describes a black-box communication channel. In this model process is described as a sequence of states which change each other with a certain probability. This model is intended to describe any sequential process like speech. HMMs have been proven to be really practical for speech decoding.

Third, it's a matching process itself. Since it would take longer than universe existed to compare all feature vectors with all models, the search is often optimized by applying many tricks. At any points we maintain the best matching variants and extend them as time goes on, producing the best matching variants for the next frame.

Models:

According to the speech structure, three models are used in speech recognition to do the match:

An **acoustic model** contains acoustic properties for each senone. There are context-independent models that contain properties (the most probable feature vectors for each phone) and context-dependent ones (built from senones with context).

A **phonetic dictionary** contains a mapping from words to phones. This mapping is not very effective. For example, only two to three pronunciation variants are noted in it. However, it's practical enough most of the time. The dictionary is not the only method for mapping words to phones. You could also use some complex function learned with a machine learning algorithm.

A **language model** is used to restrict word search. It defines which word could follow previously recognized words (remember that matching is a sequential process) and helps to significantly restrict the matching process by stripping words that are not probable. The most common language models are n-gram language models–these contain statistics of word sequences–and finite state language models–these define speech sequences by finite state automation, sometimes with weights. To reach a good accuracy rate, your language model must be very successful in search space restriction. This means it should be very good at predicting the next word. A language model usually restricts the vocabulary that is considered to the words it contains. That's an issue for name recognition. To deal with this, a language model can contain smaller chunks like sub words or even phones. Please note that the search space restriction in this case is usually worse and the corresponding recognition accuracies are lower than with a word-based language model.

Those three entities are combined together in an engine to recognize speech. If you are going to apply your engine for some other language, you need to get such structures in place. For many languages there are acoustic models, phonetic dictionaries and even large vocabulary language models available for download.

Other used concepts:

1. A *Lattice* is a directed graph that represents variants of the recognition. Often, getting the best match is not practical. In that case, lattices are good intermediate formats to represent the recognition result.

2. *N-best lists* of variants are like lattices, though their representations are not as dense as the lattice ones.

3. *Word confusion networks* (sausages) are lattices where the strict order of nodes is taken from lattice edges.

4. *Speech database* - a set of typical recordings from the task database. If we develop dialog system it might be dialogs recorded from users. For dictation system it might be reading recordings. Speech databases are used to train, tune and test the decoding systems.

5. *Text databases* - sample texts collected for e.g., language model training. Usually, databases of texts are collected in sample text form. The issue with such a collection is to put present documents (like PDFs, web pages, scans) into a spoken text form. That is, you need to remove tags and headings, to expand numbers to their spoken form and to expand abbreviations.

Optimization:

Usually, the system is tested on a test database that is meant to represent the target task correctly. The following characteristics are used:

1. ***Word error rate.*** Let's assume we have an original text and a recognition text with a length of N words. I is the number of inserted words, D is the number of deleted words and S represent the number of substituted words. With this, the word error rate can be calculated as

   $$WER = (I + D + S) / N$$

   The WER is usually measured in percent.

2. ***Accuracy.*** It is almost the same as the word error rate, but it doesn't take insertions into account.

   $$Accuracy = (N - D - S) / N$$

   For most tasks, the accuracy is a worse measure than the WER, since insertions are also important in the final results. However, for some tasks, the accuracy is a reasonable measure of the decoder performance.

3. ***Speed.*** Suppose an audio file has a recording time (RT) of 2 hours and the decoding took 6 hours. Then the speed is counted as 3xRT.

4. ***ROC curves.*** When we talk about detection tasks, there are false alarms and hits/misses. To illustrate these, ROC curves are used. Such a curve is a diagram that describes the number of false alarms versus the number of hits. It tries to find the optimal point where the number of false alarms is small and the number of hits matches 100%.

3.1.2 Python Backend:

E.V.A is coded on python language as to date, it is the most commonly used programming language in the field of machine learning and artificial intelligence. The python backend gets the output from the speech recognition module and then identifies whether the command or the speech output is an API Call and Context Extraction. The output is then sent back to the python backend to give the required output to the user.

Python is an outstanding language majorly because it doesn't need compiling into machine language instruction to be executed. A developer can directly run a program written in Python. Other than this python also provides the following advantages mentioned below:

- **A huge library ecosystem:** Python offers a vast choice of libraries for AI development, which contain base-level items that save coding time. These libraries also make it easy to access, handle, and transform data. Some of the most popular libraries are sklearn, Pandas, NumPy, Keras and caffe.

- **High Readability:** Python is famous for its compact, readable code, and is practically unmatched with regards to usability, especially for new developers. This has made it a preferred language for AI and deep learning. AI depends on incredibly complex calculations and multi-stage work processes, so the less a developer needs to stress over the complexities of coding, the more they can concentrate on discovering answers for issues, and accomplishing the objectives of the venture. Python reads like our everyday English language, thus making AI development easier and less complex. Python's concise syntax implies that it requires less coding time that most other programming languages, and permits the developer to rapidly test algorithms without executing them. Moreover, easily comprehensible code is priceless for collective coding, or when AI ventures are shared between different development groups. This is especially valid if a venture contains a lot of custom business logic or outsider parts.

- **Flexibility:** Python for AI is an extraordinary language, as it is truly flexible: It offers a choice to pick from using Object Oriented Programming (OOPS) or scripting. There's no compelling reason to recompile the source code; developers can actualize any changes and observe the outcomes. Software developers can join Python and other languages to achieve their goals. Besides, flexibility permits developers to pick the programming styles which they are completely comfortable with or even join these styles to tackle various kinds of issues in the most productive manner. The imperative style comprises commands that portray how a PC ought to play out the given commands. With this style, you characterize the sequence of calculations. The functional style is additionally called so in light of the fact that it declares what tasks ought to be performed. It doesn't consider the program state. Unlike in the imperative style, it declares proclamations as mathematical equations. The object-oriented situated style depends on two ideas: class and object, where similar objects create classes. This style isn't completely supported by Python, as it can't completely perform exemplification, yet developers can utilize this style to a limited degree. The procedural style is the most widely recognized among first-time developers, as it continues errands in a bit-by-bit position. It's frequently utilized for sequencing, iteration, modularization, and selection. The flexibility of Python diminishes the plausibility of blunders, as software developers get an opportunity to take control of the situation and work in a comfortable environment.

- **Community support:** Python is an open-source programming language and is supported by a ton of assets and top-notch documentation. It additionally flaunts a huge and dynamic network of developers ready to give guidance and help through all phases of the development procedure. A strong developers' community can be of great help while using python for AI development. A ton of Python documentation is accessible online just as in Python communities and forums, where software engineers and AI designers talk about mistakes, take care of issues, and help each other out. Python programming language is totally free just like its assortment of valuable libraries and tools.

- **Visualization options:** As mentioned before, Python has an extensive set of libraries and some of them offer amazing visualization tools. This is of great use in AI as it involves the representation of data in a human-readable format. Matplotlib is a library for data scientists that allows them to make charts, histograms, and graphs to present data in a more comprehensible and visualized manner. In fact, making clear reports is also easy with various application programming interfaces that come with Python.

3.1.3 API calls:

API stands for Application Programming Interface. An API is a software intermediary that allows two applications to talk to each other. In other words, an API is a messenger that delivers your request to the provider that you're requesting it from and then delivers the response back to you. E.V.A uses several APIs to provide several functionalities the user demands. APIs can retrieve real time data from online servers or websites like Reddit, twitter, google or Facebook. To use an API, you make a request to a remote web server, and retrieve the data you need.
But why use an API instead of a static CSV dataset you can download from the web? APIs are useful in the following cases:

- **The data is changing quickly**. An example of this is weather data. It doesn't really make sense to regenerate a dataset and download it every minute — this will take a lot of bandwidth, and be pretty slow.
- **You want a small piece of a much larger set of data**. Reddit comments are one example. What if you want to just pull your own comments on Reddit? It doesn't make much sense to download the entire Reddit database, then filter just your own comments.
- **There is repeated computation involved**. Spotify has an API that can tell you the genre of a piece of music. You could theoretically create your own classifier, and use it to compute music categories, but you'll never have as much data as Spotify does.

3.1.4 Text-to-speech module:

Text-to-Speech (TTS) refers to the ability of computers to read text aloud. E.V.A uses a TTS Engine to provide spoken results to the user's query and interact in a more human way. A TTS converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound.

TTS technology also referred to as Speech Synthesis. Speech synthesis is simply a form of output where a computer or other machine reads words to you out loud in a real or simulated voice played through a loudspeaker. Talking machines are nothing new—somewhat surprisingly, they date back to the 18th century—but computers that routinely speak to their operators are still extremely uncommon. In 1769, Austro-Hungarian inventor *Wolfgang von Kempelen* develops one of the world's first mechanical speaking machines, which uses bellows and bagpipe components to produce crude noises similar to a human voice. It's an early example of articulatory speech synthesis.

How does Speech Synthesis work?

There are essentially three stages involved, which I'll refer to as text to words, words to phonemes, and phonemes to sound.

1. **Text to Word:**

   The initial stage in speech synthesis, which is generally called pre-processing or normalization, is all about reducing ambiguity: it's about narrowing down the many different ways you could read a piece of text into the one that's the most appropriate.

   Preprocessing involves going through the text and cleaning it up so the computer makes fewer mistakes when it actually reads the words aloud. Things like numbers, dates, times, abbreviations, acronyms, and special characters (currency symbols and so on) need to be turned into words—and that's harder than it sounds. The number 1843 might refer to a quantity of items ("one thousand eight hundred and forty-three"), a year or a time ("eighteen forty-three"), or a padlock combination ("one eight four three"), each of which is read out slightly differently. While humans follow the sense of what's written and figure out the pronunciation that way, computers generally don't have the power to do that, so they have to use statistical probability techniques (typically Hidden Markov Models) or neural networks (computer programs structured like arrays of brain cells that learn to recognize patterns) to arrive at the most likely pronunciation instead. So, if the word "year" occurs in the same sentence as "1843," it might be reasonable to guess this is a date and pronounce it "eighteen forty-three." If there were a decimal point before the numbers (".843"), they would need to be read differently as "eight four three." Preprocessing also has to tackle homographs, words pronounced in different ways according to what they mean. The word "read" can be pronounced either "red" or "reed," so a sentence such as "I read the book" is immediately problematic for a speech synthesizer. But if it can figure out that the preceding text is entirely in the past tense, by recognizing past-tense verbs ("I got up... I took a shower... I had breakfast... I read a book..."), it can make a reasonable guess that "I read [red] a book" is probably correct. Likewise, if the preceding text is "I get up... I take a shower... I have breakfast..." the smart money should be on "I read [reed] a book."

2. **Words to phonemes:**

   Having figured out the words that need to be said, the speech synthesizer now has to generate the speech sounds that make up those words. In theory, this is a simple problem: all the computer needs is a huge alphabetical list of words and details of how to pronounce each one (much as you'd find in a typical dictionary, where the pronunciation is listed before or after the definition). For each word, we'd need a list of the phonemes that make up its sound.

What are Phonemes?

Crudely speaking, phonemes are to spoken language what letters are to written language: they're the atoms of spoken sound—the sound components from which you can make any spoken word you like. The word cat consists of three phonemes making the sounds /k/ (as in can), /a/ (as in pad), and /t/ (as in tusk). Rearrange the order of the phonemes and you could make the words "act" or "tack."

There are only 26 letters in the English alphabet, but over 40 phonemes. That's because some letters and letter groups can be read in multiple ways (a, for example, can be read differently, as in 'pad' or 'paid'), so instead of one phoneme per letter, there are phonemes for all the different letter sounds. Some languages need more or fewer phonemes than others (typically 20-60).

In theory, if a computer has a dictionary of words and phonemes, all it needs to do to read a word is look it up in the list and then read out the corresponding phonemes, right? In practice, it's harder than it sounds. As any good actor can demonstrate, a single sentence can be read out in many different ways according to the meaning of the text, the person speaking, and the emotions they want to convey (in linguistics, this idea is known as prosody and it's one of the hardest problems for speech synthesizers to address). Within a sentence, even a single word (like "read") can be read in multiple ways (as "red"/"reed") because it has multiple meanings. And even within a word, a given phoneme will sound different according to the phonemes that come before and after it.

An alternative approach involves breaking written words into their graphemes (written components units, typically made from the individual letters or syllables that make up a word) and then generating phonemes that correspond to them using a set of simple rules. This is a bit like a child attempting to read words he or she has never previously encountered (the reading method called phonics is similar). The advantage of doing that is that the computer can make a reasonable attempt at reading any word, whether or not it's a real word stored in the dictionary, a foreign word, or an unusual name or technical term. The disadvantage is that languages such as English have large numbers of irregular words that are pronounced in a very different way from how they're written (such as "colonel," which we say as kernel and not "coll-o-nell"; and "yacht," which is pronounced "yot" and not "yach-t") —exactly the sorts of words that cause problems for children learning to read and people with what's known as surface dyslexia (also called orthographic or visual dyslexia).

3. Phonemes to sound

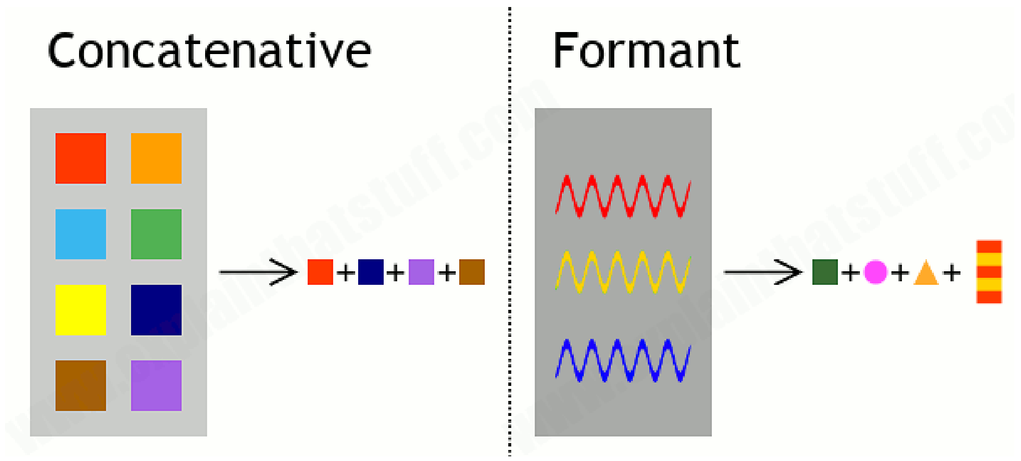
Okay, so now we've converted our text (our sequence of written words) into a list of phonemes (a sequence of sounds that need speaking). But where do we get the basic phonemes that the computer reads out loud when it's turning text into speech? There are three different approaches. One is to use recordings of humans saying the phonemes, another is for the computer to generate the phonemes itself by generating basic sound frequencies (a bit like a music synthesizer), and a third approach is to mimic the mechanism of the human voice.

Concatenative

Speech synthesizers that use recorded human voices have to be preloaded with little snippets of human sound they can rearrange. In other words, a programmer has to record lots of examples of a person saying different things, break the spoken sentences into words and the words into phonemes. If there are enough speech samples, the computer can rearrange the bits in any number of different ways to create entirely new words and sentences. This type of speech synthesis is called concatenative (from Latin words that simply mean to link bits together in a series or chain). Since it's based on human recordings, concatenation is the most natural-sounding type of speech synthesis and it's widely used by machines that have only limited things to say (for example, corporate telephone switchboards). It's main drawback is that it's limited to a single voice (a single speaker of a single sex) and (generally) a single language.

Formant

If you consider that speech is just a pattern of sound that varies in pitch (frequency) and volume (amplitude)—like the noise coming out of a musical instrument—it ought to be possible to make an electronic device that can generate whatever speech sounds it needs from scratch, like a music synthesizer. This type of speech synthesis is known as formant, because formants are the 3–5 key (resonant) frequencies of sound that the human vocal apparatus generates and combines to make the sound of speech or singing. Unlike speech synthesizers that use concatenation, which are limited to rearranging prerecorded sounds, formant speech synthesizers can say absolutely anything—even words that don't exist or foreign words they've never encountered. That makes formant synthesizers a good choice for GPS satellite (navigation) computers, which need to be able to read out many thousands of different (and often unusual) place names that would be hard to memorize. In theory, formant synthesizers can easily switch from a male to a female voice (by roughly doubling the frequency) or to a child's voice (by trebling it), and they can speak in any language. In practice, concatenation synthesizers now use huge libraries of sounds so they can say pretty much anything too. A more obvious difference is that concatenation synthesizers sound much more natural than formant ones, which still tend to sound relatively artificial and robotic.



Concatenative versus formant speech synthesis. Left: A concatenative synthesizer builds up speech from pre-stored fragments; the words it speaks are limited rearrangements of those sounds. Right: Like a music synthesizer, a formant synthesizer uses frequency generators to generate any kind of sound.

Articulatory

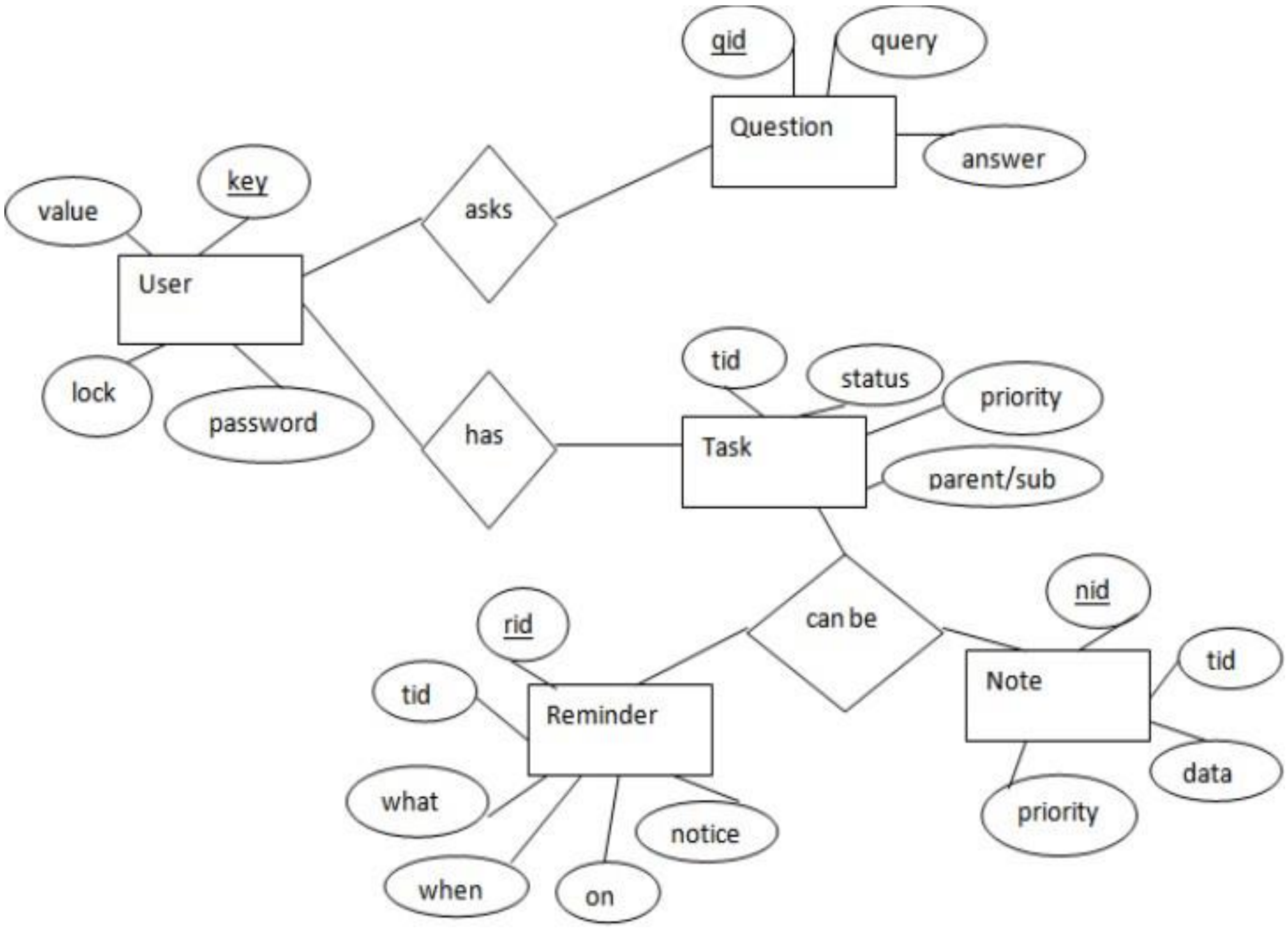
The most complex approach to generating sounds is called articulatory synthesis, and it means making computers speak by modeling the amazingly intricate human vocal apparatus. In theory, that should give the most realistic and humanlike voice of all three methods. Although numerous researchers have experimented with mimicking the human voicebox, articulatory synthesis is still by far the least explored method, largely because of its complexity. The most elaborate form of articulatory synthesis would be to engineer a "talking head" robot with a moving mouth that produces sound in a similar way to a person by combining mechanical, electrical, and electronic components, as necessary.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

The software is designed to be light -weighted so that it doesn't be a burden on the machine running it. This system is being build keeping in mind the generally available hardware and software compatibility. Here are the minimum hardware and software requirement for virtual assistant.

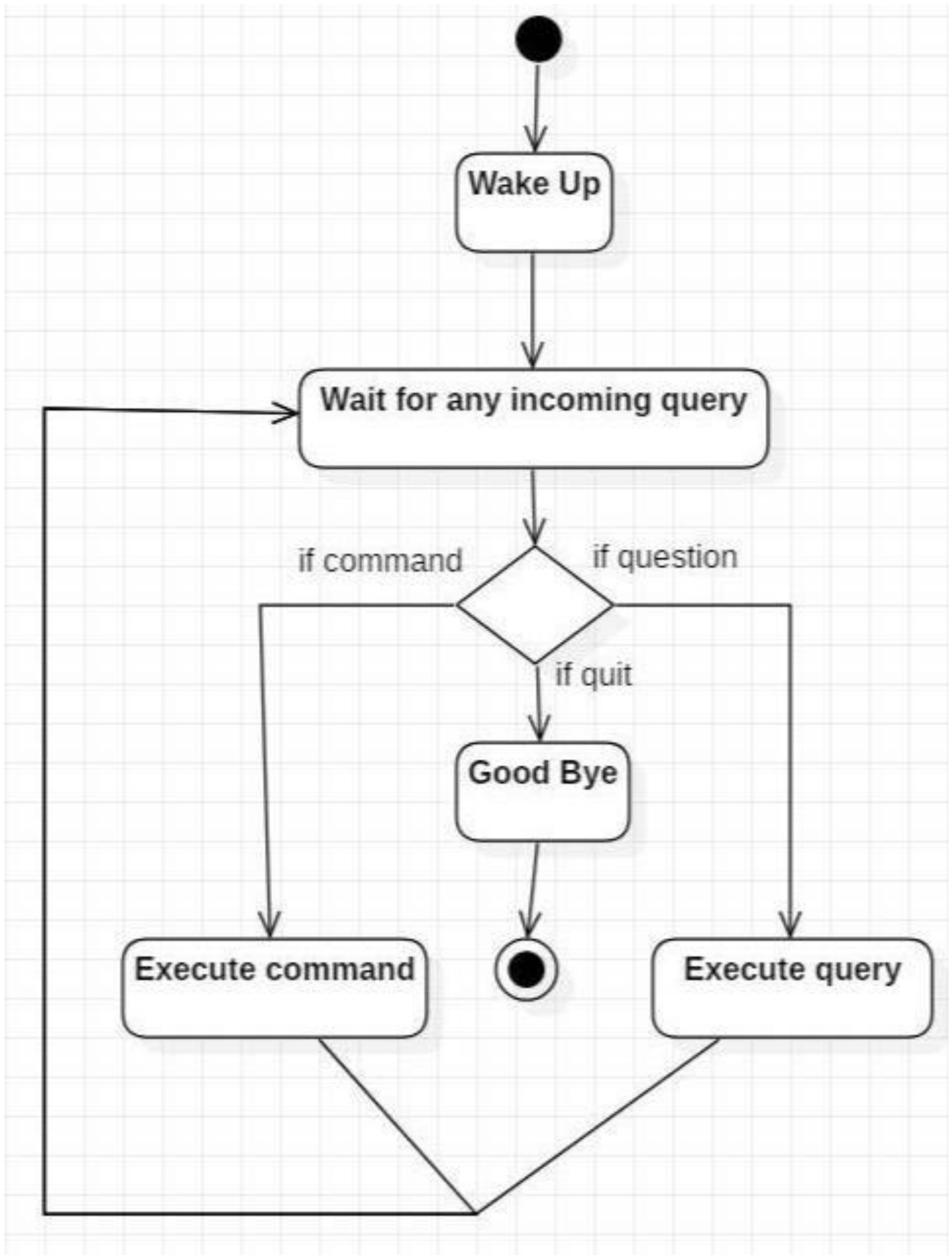
Hardware:
• Pentium-pro processor or later.
• RAM 512MB or more.
Software:
• Windows 7(32-bit) or above.
• Python 2.7 or later
• Chrome Driver
• Selenium Web Automation

3.3 ER Diagram:



The above diagram shows entities and their relationship for a virtual assistant system. We have a user of a system who can have their keys and values. It can be used to store any information about the user. Say, for key "name" value can be "Jim". For some keys user might like to keep secure. There he can enable lock and set a password (voice clip). Single user can ask multiple questions. Each question will be given ID to get recognized along with the query and its corresponding answer. User can also be having n number of tasks. These should have their own unique id and status i.e., their current state. A task should also have a priority value and its category whether it is a parent task or child task of an older task

3.4 Activity Diagram

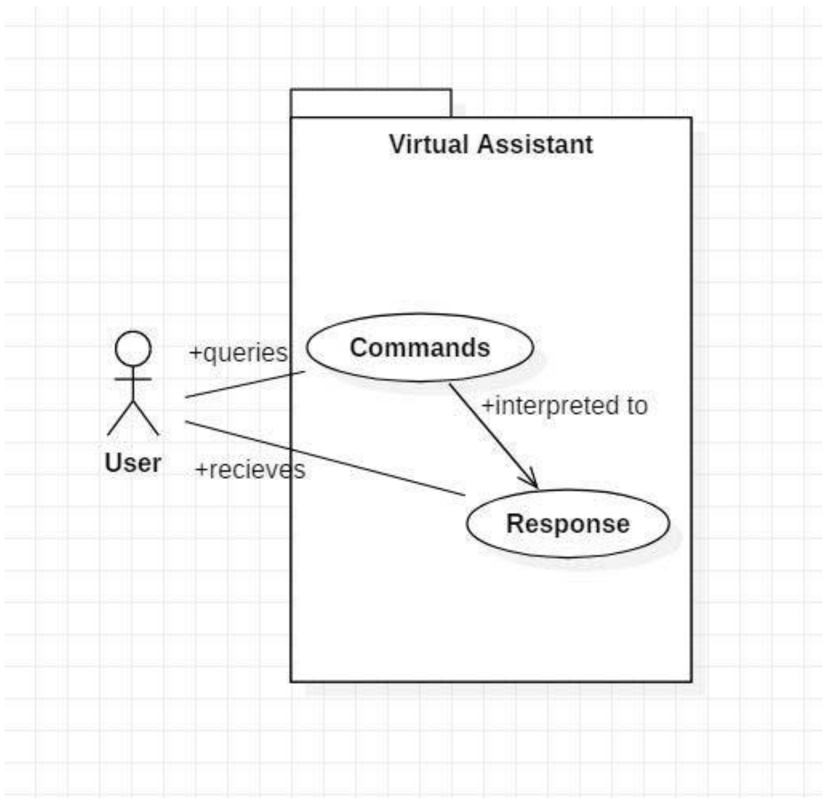


Initially, the system is in idle mode. As it receives any wake-up call it begins execution. The received command is identified whether it is a questionnaire or a task to be performed. Specific action is taken accordingly. After the Question is being answered or the task is being performed, the system waits for another command. This loop continues unless it receives quit command. At that moment, it goes back to sleep.

3.5 CLASS DIAGRAM

The class user has 2 attributes command that it sends in audio and the response it receives which is also audio. It performs function to listen the user command. Interpret it and then reply or sends back response accordingly. Question class has the command in string form as it is interpreted by interpret class. It sends it to general or about or search function based on its identification.

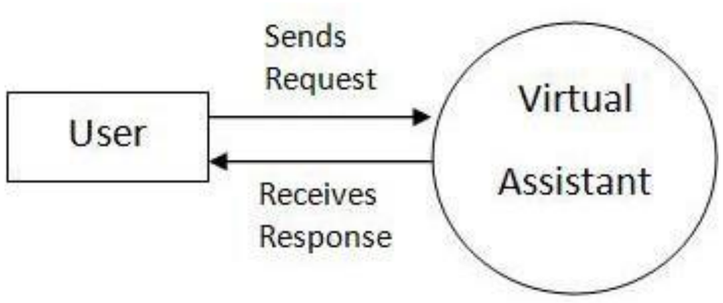
3.6 USE CASE DIAGRAM



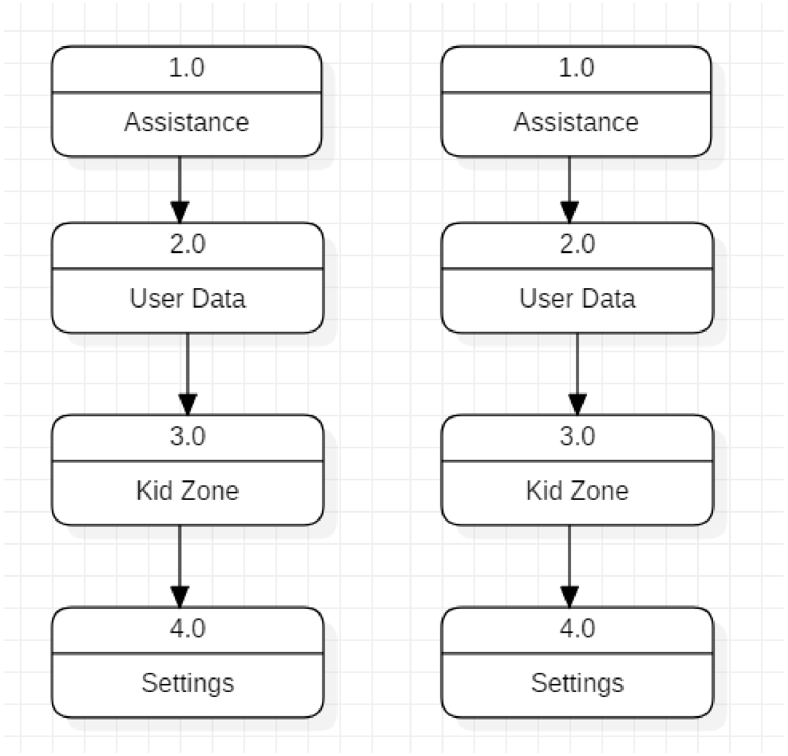
The user queries command to the system. System then interprets it and fetches answer. The response is sent back to the user.

3.7 DATAFLOW DIAGRAM

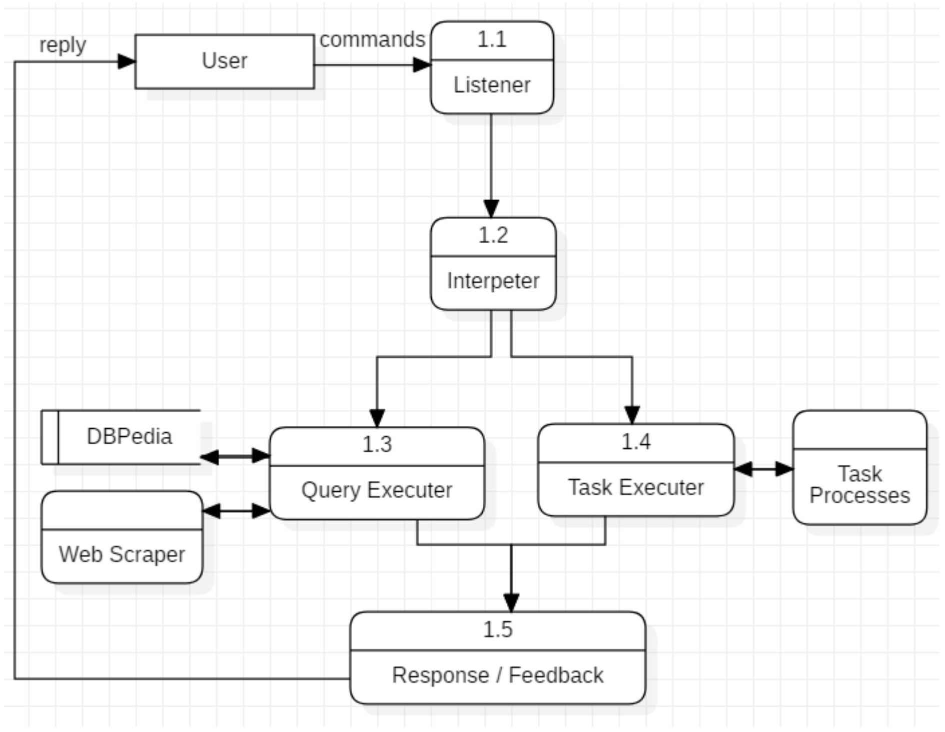
3.7.1 DFD Level 0 (Context Level Diagram)

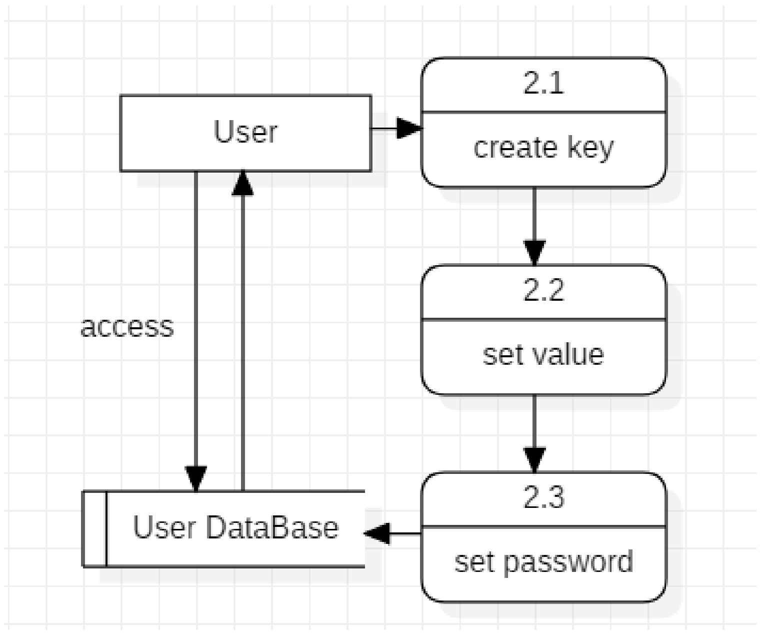


3.7.2 DFD Level 1

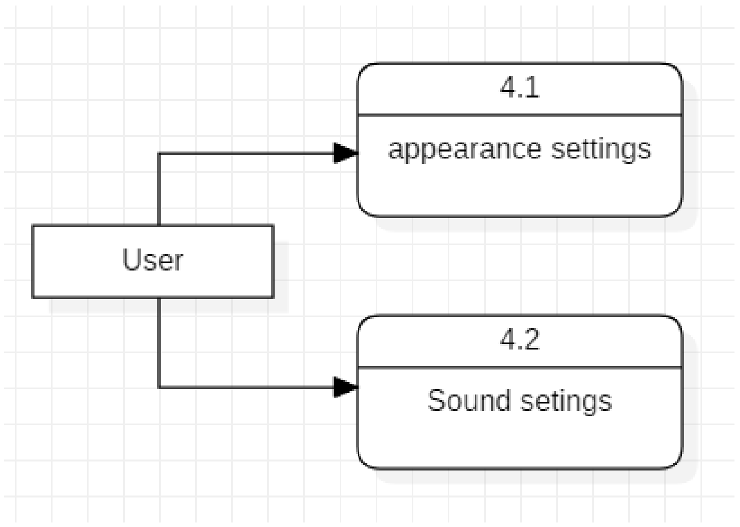


3.7.3 DFD Level 2



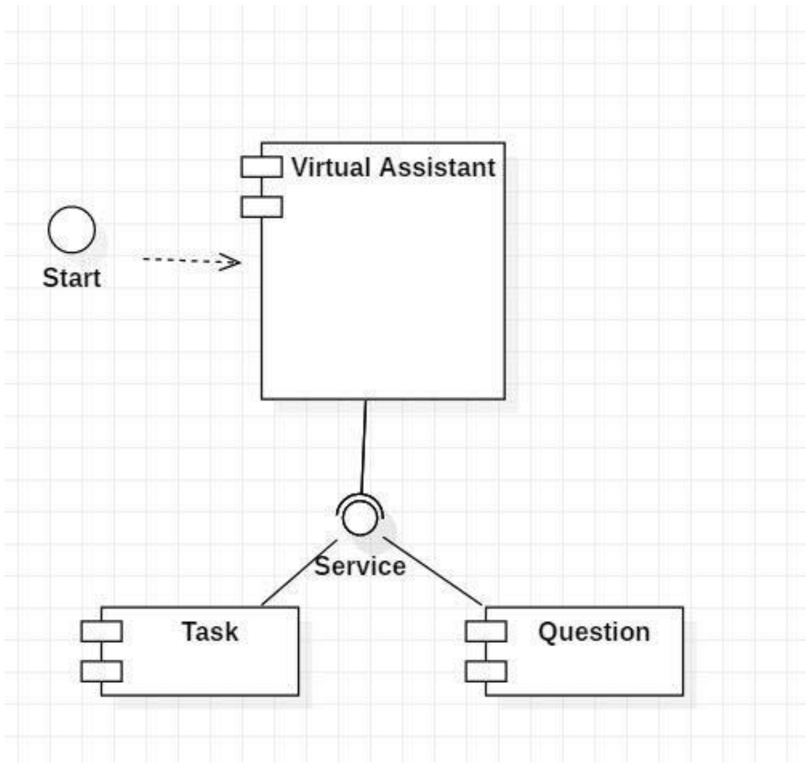


Managing User Data



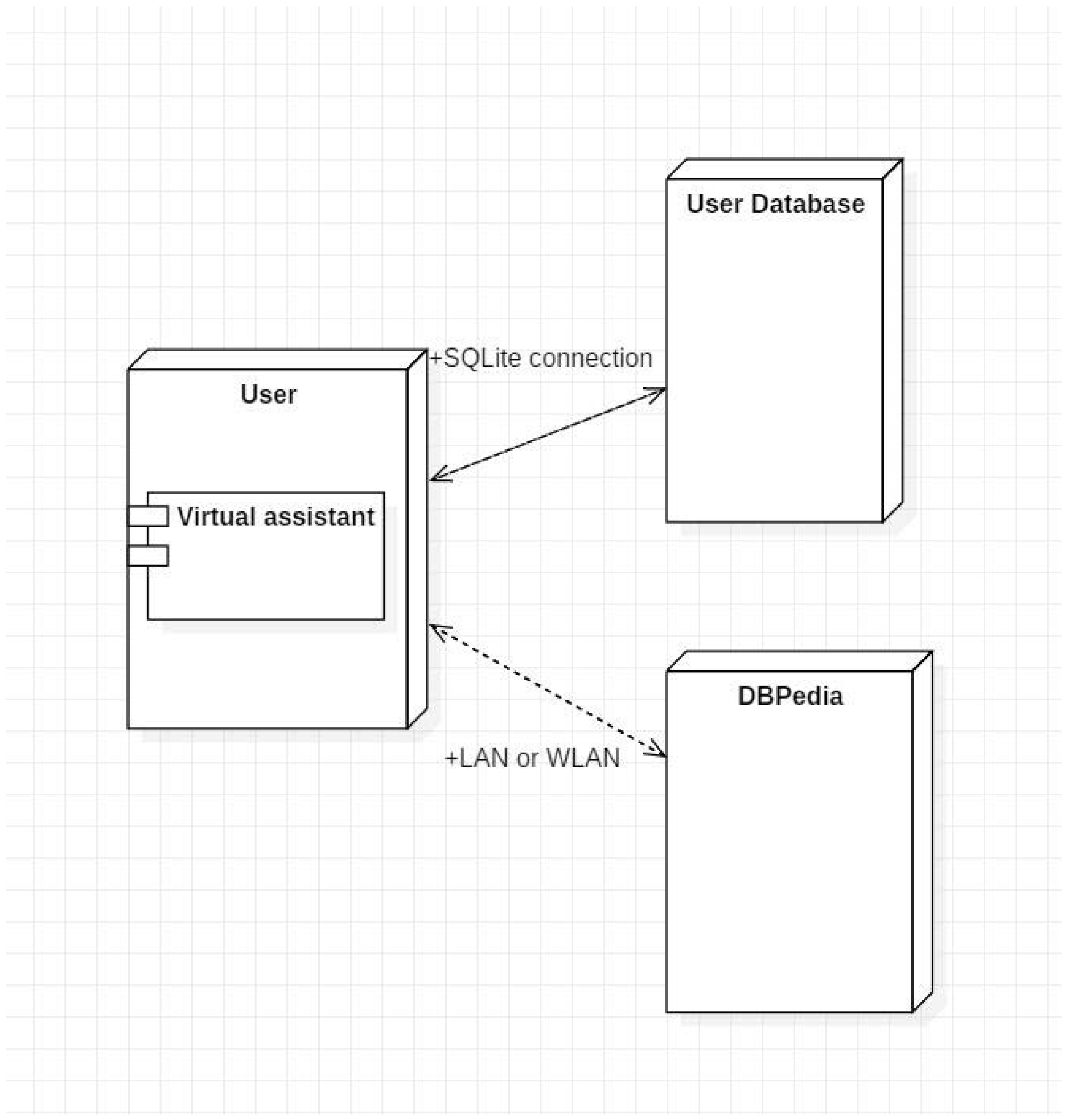
Settings of Virtual Assistant

3.8 COMPONENT DIAGRAM



The main component here is the Virtual Assistant. It provides two specific service, executing Task or Answering your question.

3.9 DEPLOYMENT DIAGRAM



The user interacts with SQLite database using SQLite connection in Python code.

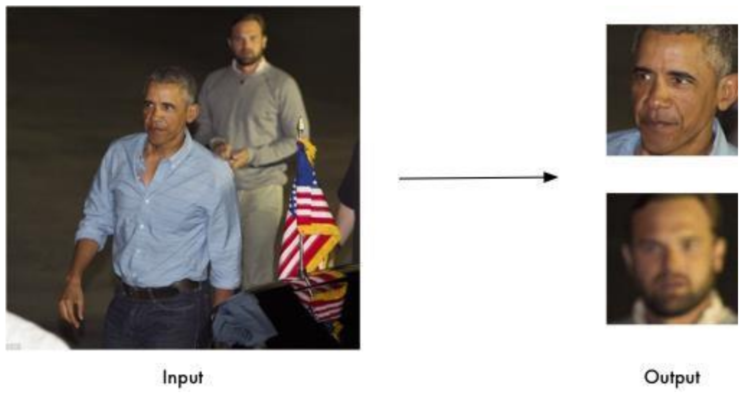
The knowledge database DBPedia must be accessed via internet connection. This requires LAN or WLAN / Ethernet network.

Chapter -4
MODULE DESCRIPTION

4.1 FACE RECOGNITION:

The virtual assistant is further improved by adding Facial Recognition framework. The facial recognition mechanism for assistant using AI strategies to detect and recognize faces.
They are two kinds of users like approved or unapproved users.
The person stands before the camera which takes different pictures of that person. The captured pictures will then undergo the face detection process.
In this procedure it basically identifies faces in the pictures and enables access to the virtual assistant for the approved users.

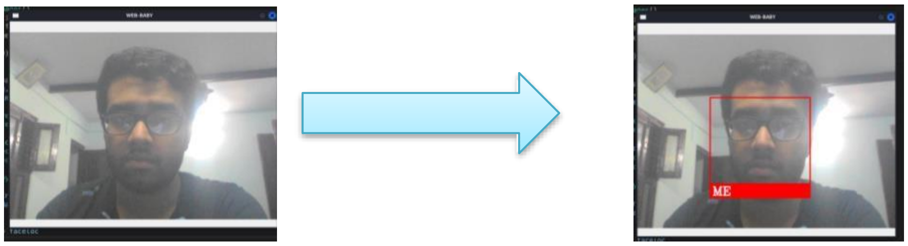
This module is used to recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face recognition command line tool that lets you do face recognition on a folder of images from the command line!



Usage

Command-Line Interface:
When you install face_recognition, you get a simple command-line program called face_ recognition that you can use to recognize faces in a photograph or folder full for photographs. First, you need to provide a folder with one picture of each person you already know. There should be one image file for each person with the files named according to who is in the picture.

Current Output



4.2 HAND GESTURE MECHANISM:

MediaPipe offers ready-to-use yet customizable Python solutions as a prebuilt Python package. MediaPipe Python package is available on PyPI for Linux, macOS and Windows.

MediaPipe Python Framework: The ready-to-use solutions are built upon the MediaPipe Python framework, which can be used by advanced users to run their own MediaPipe graphs in Python. Please see here for more info.

The MediaPipe Python framework grants direct access to the core components of the MediaPipe C++ framework such as Timestamp, Packet, and Calculator Graph, whereas the ready-to-use Python solutions hide the technical details of the framework and simply return the readable model inference results back to the callers.

MediaPipe framework sits on top of the pybind11 library. The C++ core framework is exposed in Python via a C++/Python language binding. The content below assumes that the reader already has a basic understanding of the MediaPipe C++ framework. Otherwise, you can find useful information in Framework Concepts.

Packet

The packet is the basic data flow unit in MediaPipe. A packet consists of a numeric timestamp and a shared pointer to an immutable payload. In Python, a MediaPipe packet can be created by calling one of the packet creator methods in the mp.packet_creator module. Correspondingly, the packet payload can be retrieved by using one of the packet getter methods in the mp.packet_getter module. Note that the packet payload becomes immutable after packet creation. Thus, the modification of the retrieved packet content doesn't affect the actual payload in the packet. MediaPipe framework Python API supports the most commonly used data types of MediaPipe (e.g., ImageFrame, Matrix, Protocol Buffers, and the primitive data types) in the core binding. The comprehensive table below shows the type mappings between the Python and the C++ data type along with the packet creator and the content getter method for each data type supported by the MediaPipe Python framework API.

Timestamp

Each packet contains a timestamp that is in units of microseconds. In Python, the Packet API provides a convenience method *packet.at()* to define the numeric timestamp of a packet. More generally, packet.timestamp is the packet class property for accessing the underlying timestamp. To convert a Unix epoch to a MediaPipe timestamp, the Timestamp API offers a method *mp.Timestamp.from_seconds()* for this purpose.

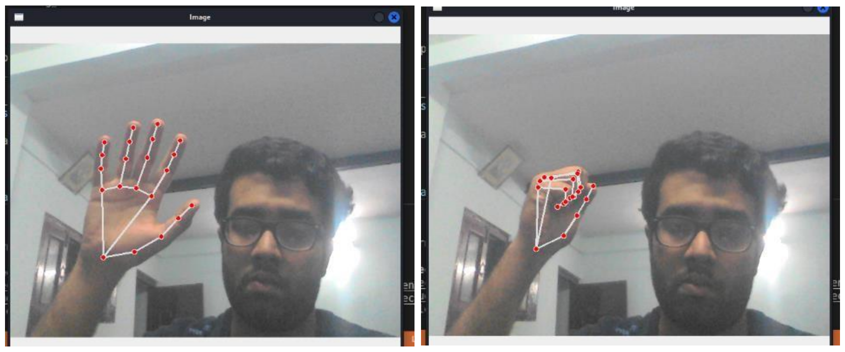
ImageFrame

ImageFrame is the container for storing an image or a video frame. Formats supported by ImageFrame are listed in the ImageFormat enum. Pixels are encoded row-major with interleaved color components, and ImageFrame supports uint8, uint16, and float as its data types. MediaPipe provides an ImageFrame Python API to access the ImageFrame C++ class. In Python, the easiest way to retrieve the pixel data is to call *image_frame.numpy_view()* to get a NumPy ndarray. Note that the returned NumPy ndarray, a reference to the internal pixel data, is unwritable. If the callers need to modify the NumPy ndarray, it's required to explicitly call a copy operation to obtain a copy. When MediaPipe takes a NumPy ndarray to make an ImageFrame, it assumes that the data is stored contiguously. Correspondingly, the pixel data of an ImageFrame will be realigned to be contiguous when it's returned to the Python side.

Graph

In MediaPipe, all processing takes places within the context of a CalculatorGraph. The CalculatorGraph Python API is a direct binding to the C++ CalculatorGraph class. The major difference is the CalculatorGraph Python API raises a Python error instead of returning a non-OK Status when an error occurs. Therefore, as a Python user, you can handle the exceptions as you normally do. The life cycle of a CalculatorGraph contains three stages: initialization and setup, graph run, and graph shutdown.

Current Output:



4.3 QT (GUI):

Qt for Python is the project that provides the official set of Python bindings (PySide6) that will supercharge your Python applications. While the Qt APIs are world renowned, there are more reasons why you should consider Qt for Python.

Qt for Python offers the official Python bindings for Qt, and has two main components:

- PySide6, so that you can use Qt6 APIs in your Python applications, and

- Shiboken6, a binding generator tool, which can be used to expose C++ projects to Python, and a Python module with some utility functions.

QT has 2 main libraries that work with python. They are:

1. **PyQT:**

   PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those applications. PyQt5 comprises a number of different components. First of all, there are a number of Python extension modules. These are all installed in the PyQt5 Python package. PyQt5 contains plugins that enable Qt Designer and qmlscene to be extended using Python code. PyQt5 also contains a number of utility programs.

   a. **pyuic5** corresponds to the Qt uic utility. It converts QtWidgets based GUIs created using Qt Designer to Python code.

   b. **pyrcc5** corresponds to the Qt rcc utility. It embeds arbitrary resources (e.g., icons, images, translation files) described by a resource collection file in a Python module.

   c. pylupdate5 corresponds to the Qt lupdate utility. It extracts all of the translatable strings from Python code and creates or updates .ts translation files. These are then used by Qt Linguist to manage the translation of those strings.

   The DBus support module is installed as dbus.mainloop.pyqt5. This module provides support for the Qt event loop in the same way that the dbus.mainloop.glib included with the standard dbus-python bindings package provides support for the GLib event loop. The API is described in DBus Support. It is only available if the dbus-python v0.80 (or later) bindings package is installed. The QtDBus module provides a more Qt-like interface to DBus. PyQt5 includes a large number of examples. These are ports to Python of many of the C++ examples provided with Qt. They can be found in the example's directory of the sdist. Finally, PyQt5 contains the specification files that allow bindings for other Qt based class libraries that further extend PyQt5 to be developed and installed.

2. **Pyside 2/6:**

   PySide2 is the official Python module from the Qt for Python project, which provides access to the complete Qt 5.12+ framework. PySide6 is the official Python module from the Qt for Python project, which provides access to the complete Qt 6.0+ framework.

Qt is currently being developed by The Qt Company, a publicly listed company, and the Qt Project under open-source governance, involving individual developers and organizations working to advance Qt. Qt is available under both commercial licenses and open-source GPL 2.0, GPL 3.0, and LGPL 3.0 licenses.

Qt is used for developing graphical user interfaces (GUIs) and multi-platform applications that run on all major desktop platforms and most mobile or embedded platforms. Most GUI programs created with Qt have a native-looking interface, in which case Qt is classified as a widget toolkit. Non-GUI programs can also be developed, such as command-line tools and consoles for servers. An example of such a non-GUI program using Qt is the Cutelyst web framework.

Qt supports various compilers, including the GCC C++ compiler, the Visual Studio suite, PHP via an extension for PHP5, and has extensive internationalization support. Qt also provides Qt Quick, that includes a declarative scripting language called QML that allows using JavaScript to provide the logic. With Qt Quick, rapid application development for mobile devices became possible, while logic can still be written with native code as well to achieve the best possible performance.

Other features include SQL database access, XML parsing, JSON parsing, thread management and network support.

Qt is built on these key concepts:

Complete abstraction of the GUI:

When first released, Qt used its own paint engine and controls, emulating the look of the different platforms it runs on when it drew its widgets. This made the porting work easier because very few classes in Qt really depended on the target platform; however, this occasionally led to slight discrepancies where that emulation was imperfect. Recent versions of Qt use the native style APIs of the different platforms, on platforms that have a native widget set, to query metrics and draw most controls, and do not suffer from such issues as often. On some platforms (such as MeeGo and KDE) Qt is the native API. Some other portable graphical toolkits have made different design decisions; for example, wxWidgets uses the toolkits of the target platform for its implementations.

Signals and slots

A language construct introduced in Qt for communication between objects which makes it easy to implement the observer pattern while avoiding boilerplate code. The concept is that GUI widgets can send signals containing event information which can be received by other controls using special functions known as slots.

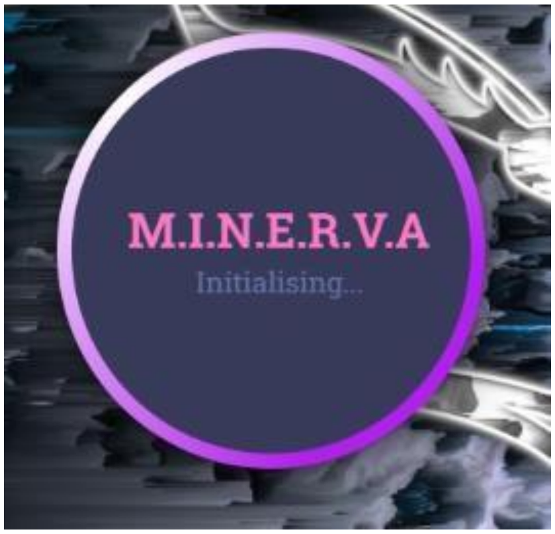
Metaobject compiler

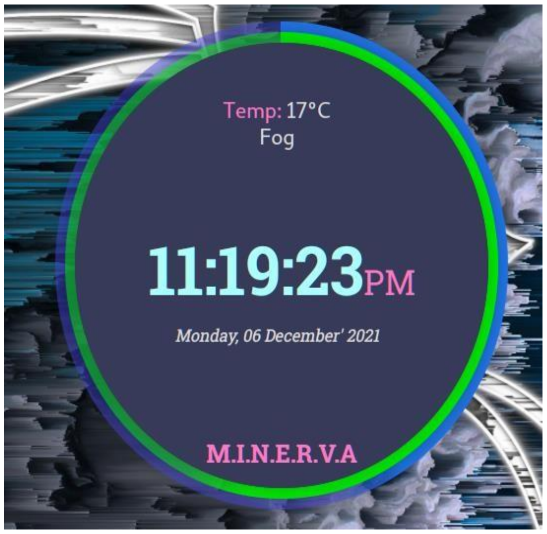
The metaobject compiler, termed moc, is a tool that is run on the sources of a Qt program. It interprets certain macros from the C++ code as annotations, and uses them to generate added C++ code with meta information about the classes used in the program. This meta information is used by Qt to provide programming features not available natively in C++: signals and slots, introspection and asynchronous function calls.

Language bindings

Qt can be used in several programming languages other than C++, such as Python, JavaScript, C# and Rust via language bindings; many languages have bindings for Qt 5 and bindings for Qt 4. The Ring programming language includes Qt in the standard library.

Current Output:





4.4 Speech Recognition:

It is a Library for Performing Library for performing speech recognition, with support for several engines and APIs, online and offline.

Speech recognition engine/API support:

- CMU Sphinx (works offline)
- Google Speech Recognition
- Google Cloud Speech API
- Wit.ai
- Microsoft Bing Voice Recognition
- Houndify API
- IBM Speech to Text

CMUSphinx:

The CMUSphinx toolkit is a leading speech recognition toolkit with various tools used to build speech applications. CMUSphinx contains a number of packages for different tasks and applications. Sometimes, it's confusing what to choose. To shed some light on the parts of the toolkit, here is a list:

- Pocketsphinx — lightweight recognizer library written in C.
- Sphinxbase — support library required by Pocketsphinx
- Sphinx4 — adjustable, modifiable recognizer written in Java
- Sphinxtrain — acoustic model training tools

Of course, many things are missing. Things like building a phonetic model capable of handling an infinite vocabulary, postprocessing of the decoding result, sense extraction and other semantic tools should be added one day. Probably you should take it on.

Google Speech Recognition/ Cloud Speech API:

google is one of the most advance companies that excel in the art of developing new AI and ML based technologies. Therefore, it is no surprise that they have worked on the best speech recognition system considered in the market till date. Some key features of google speech recognition system are:

- **Speech adaptation**

  Customize speech recognition to transcribe domain-specific terms and rare words by providing hints and boost your transcription accuracy of specific words or phrases. Automatically convert spoken numbers into addresses, years, currencies, and more using classes.

- **Domain-specific models**

  Choose from a selection of trained models for voice control and phone call and video transcription optimized for domain-specific quality requirements. For example, our enhanced phone call model is tuned for audio originated from telephony, such as phone calls recorded at an 8khz sampling rate.

- **Easily compare quality**

  Experiment on your speech audio with our easy-to-use user interface. Try different configurations to optimize quality and accuracy.

- **Speech-to-Text On-Prem**

  Have full control over your infrastructure and protected speech data while leveraging Google's speech recognition technology on-premises, right in your own private data centers.

- **Global vocabulary**

  Support your global user base with Speech-to-Text's extensive language support in over 125 languages and variants.

- **Streaming speech recognition**

  Receive real-time speech recognition results as the API processes the audio input streamed from your application's microphone or sent from a prerecorded audio file (inline or through Cloud Storage).

Wit.ai:

Wit enables you to turn what users say into actions. You can start by entering samples of your users might say (utterances). Behind the scene, Wit combines various state-of-the-art Natural Language Processing techniques and several speech recognition engines in order to achieve low latency and high robustness to both surrounding noise and paraphrastic variations (there are millions of ways to say the same thing). Fortunately, you don't need to care about all this machinery. We focus all our energy into creating the simplest developer experience possible. You can be up and running in a few minutes using our website. Wit will adapt to your domain over time, from ice-cream distribution to space missions. Wit makes no assumptions and remains 100% configurable. It will take you just 5 minutes to build your own Wit configuration.

Microsoft Bing Voice Recognition:

Microsoft Bing Speech API Voice Recognition software is a cloud-based API that allows users to add speech-driven actions like real-time interactions to their applications. Users can use this software to perform speech translations, convert audio to text, and text to speech. This cloud-based software allows users to add real-time speed-to-text functionalities to their applications for use cases like call center log analysis, conversation transcription, and voice commands. Plus, users can tailor their speech recognition models to adapt to different speaking styles, unique vocabularies, and expressions, and to accommodate accents, noise, and voice patterns. Developers can use this software to build applications that speak to users using natural language. Microsoft Bing Speech API Voice Recognition software allows users to convert text to an audio in near real-time, and tailor the speed, volume, and pitch of the speech. Users can customize their applications using custom voice models to fit their brand voice; all that is needed is to record and upload the training data, then the software will create a distinct voice font.

Microsoft Bing Speech API Voice Recognition software enables users to add real-time speech translation functionalities to their app in over 30 languages to receive either a speech or text translation back. This software bases its speech translation models on speech recognition and neural machine translation technologies. The Microsoft Bing Speech API Voice Recognition software understands the way people speak in real life because of its optimization.

Features:

Microsoft Bing Speech API Voice Recognition software allows businesses to build apps that speak naturally. They are over 100 voices in more than 40 languages available on this software for users to choose from for their apps and services. This software enables users to customize their brand voice, to fit their identity and access different speaking styles and tones to meet their business use case. Businesses can tune voice output different scenarios by adjusting pitch, rate, pauses, and pronunciation. Plus, the Microsoft Bing Speech API Voice Recognition software supports flexible deployment from the cloud to the edge and different speaking styles, like chat, customer service, emotions, and newscasts.

Microsoft Bing Speech API Voice Recognition software offers users full privacy and security. With this software, users have complete control of their data, and it does not store their text data during audio generation or data processing. Backed by Azure infrastructure, this software offers users enterprise-grade security, compliance, manageability, and availability. Besides, the Microsoft Bing Speech API Voice Recognition software encrypts all stored data so users and delete and view their custom voice data and models whenever.

Microsoft Bing Speech API Voice Recognition software allows multilingual communication to take place. Users can translate the audio from over 30 languages and customize the translations to fit the specific terms of their organization. This software offers reliable and fast translations powered by neural machine translation technology. Businesses can tailor models to recognize unique speaking styles and domain-specific terms.

With the engine trained to normalize speech output, users can efficiently deliver readable translations. This software does not log speech input during processing. Users can add high-quality speech-to-speech and speech-to-text translations to their apps. Additionally, the Microsoft Bing Speech API Voice Recognition software allows users to deploy and train a custom translation system without needing machine learning expertise.

Microsoft Bing Speech API Voice Recognition software helps users convert spoken audio to text accurately in different languages. This software allows businesses to customize models to improve accuracy for domain-specific terminology. Users can enable analytics or search on transcribed documents to get more value from the audio. The Microsoft Bing Speech API Voice Recognition software allows users to convert audio from different sources like audio files, blob storage, and microphones.

Users can use speaker diarization to determine the speakers, and they can use automatic formatting and punctuation to improve the readability of the transcripts. This software helps users overcome barriers like accents, unique vocabulary, or background noise. Also, users can generate custom models with Office 365 data automatically to optimize speech recognition accuracy for their organization.

Microsoft Bing Speech API Voice Recognition software offers users flexible pricing to enable them to pay for only what they use. Businesses can use this software's SDK to develop ambient devices that have a customized keyboard. The Microsoft Bing Speech API Voice Recognition software SDK allows different scenarios, including a smart speaker, voice assistants, and conversation transcription. Plus, this software supports Android, Linux, and Windows devices.

Microsoft Bing Speech API Voice Recognition software enables users to create audio content with human-sounding voices. Content creators can use this software to visually inspect speech attributes like volume, pitch, rate, voice style, and pronunciation in real-time. This inspection allows users to tailor speech patterns and develop customized audio output with ease.

Media organizations can use the Microsoft Bing Speech API Voice Recognition software to ensure reporting precision. This software allows users to fine-tune domain-specific expressions to fit their needs. Besides, businesses can use this platform to create human-like audio for training videos and product demos.

Microsoft Bing Speech API Voice Recognition software allows users to configure commands easily. With custom commands, users can use voice to complete their tasks. This software enables users to customize the controls to fit the unique needs of their brand or industry. Additionally, the Microsoft Bing Speech API Voice Recognition software supports natural input and output, and multiple commands from users.

Houndify API:

Houndify has features as specified below:

- **Wake words:**

  Control your product voice experience and deepen user engagement through an independent, custom voice assistant that's activated by a branded wake word (or wake phrase). Houndify Wake Words also support local, embedded hardware implementations that include certified multi-phrase technology, enabling multiple assistants to live side-by-side in a single device.

- **Automatic Speech recognition:**

  Advanced artificial intelligence is at the root of Houndify's Automatic Speech Recognition. Beginning with the first word spoken by the user, our innovative ASR actively listens and processes complex language patterns, accurately capturing user input in real time—even in the noisiest of environments. Available in multiple languages and growing.

- **Natural Language Understanding:**

  Houndify's Natural Language Understanding (NLU) is built upon our Deep Meaning Understanding technology, allowing voice assistants to interpret complex conversations containing multiple criteria, exclusions, and cross-domain compound queries. Greater understanding and the ability to respond conversationally enables humans to interact with products and services the way we interact with each other, by speaking naturally.

- **Custom Domain:**

  Designed to provide flexibility and growth over time, the Houndify Voice AI platform includes access to hundreds of content domains connected via a dynamic knowledge graph. Houndify Custom Domains allow content providers to voice-enable their data and services through products powered by Houndify.

IBM Speech to Text:

The Speech to Text service provides an API to add speech transcription capabilities to applications. It combines information about language structure with the composition of the audio signal.

The IBM Watson Speech to Text service offers many advanced features to help you get the most from your audio transcription. The service offers multiple speech recognition interfaces, and these interfaces support many features that you can use to manage how you pass your audio to the service and the results that the service returns. You can also customize the service to enhance its vocabulary and to accommodate the acoustic characteristics of your audio. And as with all Watson services, SDKs are available to simplify application development in many programming languages.

- **Using languages and models**

  The service supports speech recognition for the many languages listed in Language support. The service provides different models for the languages that it supports. Most language models are generally available (GA) for production use; a few are beta and subject to change.

  - For most languages, the service offers previous-generation Broadband and Narrowband models. Most previous-generation models are GA.

  - For a growing set of languages, the service offers next-generation Multimedia and Telephony models that improve upon the speech recognition capabilities of the previous-generation models. All next-generation models are GA. Next-generation models return results with greater throughput and higher accuracy than previous-generation models.

  For most languages, you can transcribe audio at one of two sampling rates:

  - Use Broadband or Multimedia models for audio that is sampled at a minimum sampling rate of 16 kHz.

  - Use Narrowband or Telephony models for audio that is sampled at a minimum sampling rate of 8 kHz.

- **Using audio formats**

  The service supports speech recognition for the many audio formats listed in Audio support. Different formats support different sampling rates and other characteristics. By using a format that supports compression, you can maximize the amount of audio data that you can send with a request.

Recognizing speech with the service

The Speech to Text service offers a WebSocket interface and synchronous and asynchronous HTTP Representational State Transfer (REST) interfaces.

- The WebSocket interface offers an efficient, low-latency, and high-throughput implementation over a full-duplex connection.
- The synchronous HTTP interface provides a basic interface to transcribe audio with blocking requests.
- The asynchronous HTTP interface provides a non-blocking interface that lets you register a callback URL to receive notifications or poll the service for job status and results.

All interfaces provide the same basic speech recognition capabilities, but you might specify the same parameter as a request header, a query parameter, or a parameter of a JSON object depending on the interface that you use. The service can also return different results depending on the interface and parameters that you use with a request.

4.5 gTTS:

gTTS (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Write spoken mp3 data to a file, a file-like object (byte string) for further audio manipulation, or stdout. Or simply pre-generate Google Translate TTS request URLs to feed to an external program.

gTTS is a very easy to use tool which converts the text entered, into audio which can be saved as a mp3 file. The gTTS API supports several languages including English, Hindi, Tamil, French, German and many more. This is extremely useful when there is a communication barrier and the user is unable to convey his messages to people. Text-to-speech is a great help to the visually impaired people or people with other disabilities as it can help them by assisting in the text to speech translation. There are also many ideas possible with the gTTS module and it can be used for other languages as well. The speech can be delivered in any one of the two available audio speeds, fast or slow. However, as of the latest update, it is not possible to change the voice of the generated audio.

Definitions:-

Pre-processor: Function that takes text and returns text. Its goal is to modify text (for example correcting pronunciation), and/or to prepare text for proper tokenization (for example enduring spacing after certain characters).

Tokenizer: Function that takes text and returns it split into a list of tokens (strings). In the gTTS context, its goal is to cut the text into smaller segments that do not exceed the maximum character size allowed for each TTS API request, while making the speech sound natural and continuous. It does so by splitting text where speech would naturally pause (for example on ".") while handling where it should not (for example on "10.5" or "U.S.A."). Such rules are called tokenizer cases, which it takes a list of.

Tokenizer case: Function that defines one of the specific cases used by gtts.tokenizer.core.Tokenizer. More specifically, it returns a regex object that describes what to look for for a particular case. gtts.tokenizer.core.Tokenizer then creates its main regex pattern by joining all tokenizer cases with "|".

Pre-processing and tokenizing

The gtts.tokenizer module powers the default pre-processing and tokenizing features of gTTS and provides tools to easily expand them. gtts.tts.gTTS takes two arguments pre_processor_funcs (list of functions) and tokenizer_func (function).

Abbreviations

The default abbreviations are defined by the gtts.tokenizer.symbols.ABBREVIATIONS list. Add a custom one to it to add a new abbreviation to remove the period from. Note: the default list already includes an extensive list of English abbreviations that Google Translate will read even without the period.

Minimizing

The Google Translate text-to-speech API accepts a maximum of 100 characters.
If after tokenization any of the tokens is larger than 100 characters, it will be split in two:
• On the last space character that is closest to, but before the 100th character;
• Between the 100th and 101st characters if there’s no space

Chapter -5

FUTURE SCOPE

We are currently trying to embed all the features in Virtual Assistant which will be implemented in a highly trained Chatbot System. If successful then this can further revolutionize the way we interact with machines. people would be able to control machines by just using gestures and through voice commands. This system might get implemented into every major operating system there exists and thus people would prefer this over other modes of interactive environments and it can become the future computer main basis of interaction.

There have recently been several advancements in the field of Virtual Assistants. With companies like Google, Amazon, Apple and IBM investing in this technology it is for sure considered a subject of near future. The Siri voice assistant was released as an app for iOS in February 2010. It was the first modern digital virtual assistant installed on a smartphone, introduced later as a feature of the iPhone 4S on 4 October 2011. Others followed including IBM's Watson, Microsoft's Cortana, Amazon's Alexa, Facebook’s “M", Google's G-Assistant, Alibaba's Ali Genie, and Yandex's Alice. And, thus, the era of the smart speaker for the end-consumers began. But also, behind the scenes of the general public were advancements in businesses' use of AI with virtual assistants.

There is not just one field where the Virtual Assistants can be implemented, currently they are being employed as personal assistants in the modern-day mobile phone (like Siri, Google Assistant, and Alexa), in home automation systems (like amazon Echo), in self driven cars of tesla, for answering queries and helping with services monitoring for IBM (IBM Watson) and several other businesses to either help the optimize their workflow or to predict their sales outcome for the year. It is no doubt that virtual assistants have made their place in the world already.

Future of Virtual Assistants for the Enterprise

With businesses across all verticals, 2020 kicked off a decade of innovation with Conversational AI and the use of cloud computing with advanced platforms being offered by Amazon Web Services, Google Cloud, and Microsoft Azure. But now, it's less about science projects with a voice-activated typewriter, and more about applied AI solving real problems. True innovation is in the application rather than the AI itself.

Across every vertical, over 50% of companies surveyed by McKinsey's "The state of AI in 2020," have already deployed AI within at least one business function. And with an acceleration of digital transformation last year due to COVID, the use of virtual assistant technologies can be coupled with the benefits of expanded omnichannel and digital in general.

In this first of many in a series, we are going to double-click across the use cases and impact across various industries such as Retail, Automotive, Financial Services, Healthcare, and Life Sciences, and more. Up first, let's take a look at two particular instances of how these impacts: The retail industry (B2C) and the technology industry (B2B).

Retail and Virtual Assistants

Like in many cases, the retail industry tends to experience consumer-level disruption earlier than other industries, and we're seeing a significant impact on retail e-commerce during the pandemic. Over the last 12 months to 18 months, investments in engagement technologies such as live video, virtual stores, and chats have increased steadily. Per Forrester, over 90% of customers say their behaviors are different due to COVID as they avoid physical stores, put discretionary shopping on hold, and buy exclusively online or as much as possible online.

It's no surprise that e-commerce participants are increasingly turning towards chatbots and virtual assistants to provide 24×7x365 support to their online shoppers. Chatbots on websites have become a standard approach to connecting the growing number of online digital prospects with the human call center. Salesforce customer service research forecasts a 136% increase adoption of AI-powered chatbots. AI is transforming workers' roles, and customer service agents are no exception. Overall, 70% of agents believe that automating routine tasks would allow them to focus on more complex work. A whopping 84% of service organizations with AI report improved prioritization of agents' work -- the top benefit -- and a similar number (82%) have increased first contact resolution (FCR) rates for customers. For nearly four-fifths (79%) of these teams, AI has led to increased customer satisfaction or Net Promoter Scores, and three-quarters even report happier agents. Sixty-nine percent of teams with AI have even seen increased case deflection as a result of their adoption, a boon to agents and customers alike. Seventy-four percent of AI users report reduced agent email and calls, and three-quarters even credit AI with increased agent morale. Salesforce marketing research also indicates explosive trends toward the adoption of AI and voice-enabled technologies. A whopping 84% of marketers reporting use of AI applications in 2020, up from 29% in 2018." Marketers with AI have an average of seven use cases, up from six in 2018. 70% of high-performing marketing organizations have a fully defined AI strategy.

However, with advances in technology, the simple selection of questions that lead to connecting with a live person doesn't scale and is now being replaced with a more intelligent agent that is more human-like and can actually hold a personalized, contextual conversation. This means servicing a high volume of digital touch without involving the call center or a sales representative. It also means allowing for a much higher level of scale with an expected timely response. But even more importantly it provides for the personalized experience the digital natives expect at scale

Virtual assistants can have an even greater impact. The future involves a whole new concept of guided selling for real-time ecommerce transactions. Ecommerce transactions are already seeing massive transformation as shopping moves from traditional website storefronts to "shopping at the edge" -- live video, virtual client ling, social commerce, and more as brands engage the customers on the channel of their choice rather than being forced to go to a particular site to complete their purchase. As the purchase experience is brought to the consumer versus bringing the consumer to the purchase experience, they will benefit from their personal concierge, an in-app agent. The consumer's experience becomes ideal, and the enterprise benefits from reduced cart abandonment and increased market basket.

So, imagine browsing and shopping through a virtual store and having E.V.A, your virtual assistant, helping you as you browse -- recommending products that are popular; options to add accessories; answering questions about size, availability, delivery / pick up options; warranties, and more. This would especially be relevant for buying experiences for more expensive or complex products that may require a dialogue with a salesperson. Products such as luxury goods (e.g. plans, boats, automobiles) or those requiring significant fit requirements (e.g., shoes) would fit really well here. E.V.A would be there throughout your shopping and browsing experience ensuring that your purchase experience is positive.

Not only will E.V.A be there to help you purchase, but she will also follow up on your order by email or SMS or messaging app to ensure you are happy. Effectively, E.V.A will be there to support the shopping, buying, and owning experiences throughout the consumer's journey. So any issues or questions, talk to E.V.A! And E.V.A never sleeps, is always there on-demand, is extremely polite, and always remembers your previous conversations!

One retailer explained a challenge that their individual stores were having with after-hour inquiries, which increased during COVID-19. In response, they "hired" Marisa, their virtual assistant, who now follows up during any time of the day, and works with customers after hours to engage with a person the next business day. She also assists with retail store shows and events to not only make sure attendance expectations are exceeded, but that post-show follow-up is handled.

The impact of virtual assistants is not limited to just B2C, but they can also have a huge impact on B2B Commerce. Hanover Research believes that by 2025, the B2B e-commerce sector will grow to be as significant as its B2C counterpart. B2B e-commerce is expected to draw 12% of all B2B sales by harnessing B2C e-commerce strategies to acquire new customers, upsell/cross-sell with current buyers, and empower buyers to more easily serve themselves.

Technology Companies and Virtual Assistants

If you are a technology provider serving other businesses, then embracing Conversational AI can assist you in providing personalized and differentiated experiences that build relationships with B2B customers. The one-enterprise-to-many-SMB-customers is a known challenge for many technology companies. By using automation where each interaction can be context-aware and informed by past interactions, B2B enterprises will benefit from improved customer acquisition, reduced churn, increased revenue per customer, and reduced cost to serve.

As one can imagine these businesses are, by definition, savvy about the latest in the cloud and cloud-scale innovations such as Conversational AI, and it comes as no surprise that they would embrace these technologies faster than other industries. In fact, many larger B2B enterprises either build their own solutions or use cloud frameworks to build their solutions.

Technology companies find that 80% of their volume is not yet ready to talk to human sales representatives. Virtual assistants can work to offload marketing and sales teams, focusing their energy on the 20% that is ready to engage a live person. Intelligent virtual assistants will drive pre-event registration; re-engage retired leads; proactively prospect with account-based programs and outbound sales plays; follow-up on software and content downloads; progress pipeline with opportunities that have no recent sales activity; and drive usage for no or low usage customers who are at risk of churning with incentives like service credits.

A typical technology customer at Conversica might have virtual assistants in different functional organizations like marketing and sales; they might be aligned with different product groups; and finally assigned to operate in different regions of the world speaking different languages and in different time zones (see graphic).

A World with AI-Based Virtual Assistants

Virtual Assistants, in general, provide a unique opportunity to create a new digital workforce that augments the human workforce, and in a way that ultimately benefits the B2C end-user consumer and/or B2B end-customer. With Conversational AI and cloud-enabled services, virtual assistant technology can be applied to external customer-facing use-cases effectively, helping companies address the needs of the digital natives, and scaling omnichannel digital touch with omnichannel digital virtual assistants. And most importantly, "out-of-the-box" virtual assistant applications have been developed for augmenting specific enterprise teams - like marketing, sales, and customer success teams - which can turn a once complex build-it-yourself project into a simple SaaS-deployment project.

Chapter-6

Result and Conclusion:

The project is able to be implemented into any distro of Linux and Windows. The main purpose of the implementation of Chatbot and providing the functionalities of Robotic Process Automation has been fulfilled. E.V.A is now able to perceive several input using speech recognition techniques and respond to the queries either with a resulting task or a resulting speech, thanks to gTTS. E.V.A is able to scrape the web for information, play music on the system, authenticate a user by face recognition system, create and save text files, execute several applications all by itself without the user having to touch the system. The user just has to give a voice command or ask a query through voice enabling E.V.A to response in an expected manner. E.V.A is also equipped with a hand gesture recognition system in case the user has to increase the volume, change volume, or control a mouse pointer without having to touch the hardware components of the systems.

E.V.A is just a starting phase of a bigger project named Machine Integrated Neural network-based Environment Restrictive Virtual Assistant (M.I.N.E.R.V.A), which is expected be completed with its own desktop environment and be accessible to linux users. MINERVA will be able to take complete control over that desktop environment and provide with assistance by just more than a speaking software as the most of virtual assistant today are. Minerva will be able to scrape web and provide accurate result to the user's query and also automatically play soothing music when it detects that the user is distressed. Minerva could perform several calculations based on users need and provide with an accurate result before the users might ask for. Also as a security factor, Minerva could protect the system from various breaches to privacy or from different viruses and trojans.

REFERENCES

1. https://www.qt.io/qt-for-python
2. https://google.github.io/mediapipe/
3. https://face-recognition.readthedocs.io/en/latest/readme.html
4. https://pypi.org/project/gTTS/
5. https://pypi.org/project/SpeechRecognition/
6. https://cmusphinx.github.io/wiki/tutorialconcepts/