International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Complete implementation of BRIAN: Basic Response Interaction and Assistance Network

Dharini U[a]*, U Prjwal*[b]*, Jayanth N*[c]*, Vishal NV*[d]*, Suman Jayakumar*[e]

[a] *Student, Dept. of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, Karnataka, Pin Code 570028. Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, Pin Code 590018*
[b] *Student, Dept. of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, Karnataka, Pin Code 570028. Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, Pin Code 590018*
[c] *Student, Dept. of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, Karnataka, Pin Code 570028. Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, Pin Code 590018*
[d] *Student, Dept. of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, Karnataka, Pin Code 570028. Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, Pin Code 590018*
[e] *Assistant Professor, Dept. of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, Karnataka, Pin Code 570028. Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, Pin Code 590018*

A B S T R A C T

In the beginning of 1910's, the concept of talking to machines technically known as "virtual assistant" started. Radio Rex was the voice activated toy dog would come out of the house when its name was called released in 1911 that led to major step to the development of Bell Labs' "Audrey", the Automatic Digit Recognition machine in 1952. The project is a software agent that has both natural language and artificial intelligence. The natural language is a technology that involves the combined form of automatic speech recognition while artificial intelligence is the intelligence demonstrated by the machine. As they say, "To move mountains" the virtual assistance has more evolved from a 6 foot tall "Audrey" to a 9.25 inch "Amazon Echo". The core idea behind this initiative is to create an intelligent voice assistant system that can interact with people in their local language so that the system can help them to make their day to day life simpler. The system is designed while taking the rural life in consideration as well as their basic needs. This paper also tends to explore some new possibilities.

1. Introduction

An intelligent virtual assistant (IVA) or intelligentpersonal assistant (IPA) is a software agent thatcan perform tasks or services for an individualbased on commands or questions. In 1990s digitalrecognition technology became a feature of the PCwith IBM and Philips and Lernout & Hauspiefighting for customers. Later, in 1994 the firstsmartphone IBM Simon was launched that laid tothe foundation of smart virtual assistants. Virtualassistants work via Text, Voice and images. Theseassistants use NPL that is Natural ProcessingLanguage to match text or voice input toexecutable commands, AI Artificial Intelligenceand techniques including ML Machine Learningand sometimes Image Processing too. The deviceand the objects that use virtual assistants are smartspeakers like Google Home, Amazon Alexa andAppleHomePod, instant messaging apps, mobileoperating system, appliances, cars etc. They canprovide wide range of services like provinginformation, broadcasting weather reports, playingmusic and videos and also it can be a remainder toyou as well.

2. Literature Survey

- Speech recognition software has come along, long way to where we are today.Echo is slimmer than a cool drink glass,but the first speech recognition

* *Corresponding author.*
E-mail address: dharini.10011@gmail.com

machinesdeveloped during the middle of the 20thCentury nearly took up an entire room.

- In 1773, Russian scientist ChristianKratzenstein, a professor of physiology inCopenhagen, built a peculiar device thatproduced sounds similar to human vowelsusing resonance tubes connected to organpipes.
- A decade later, Wolfgang von Kempelenin Vienna created a similar Acoustic-Mechanical Speech Machine.
- Then in 1881, Alexander Graham Bell, hiscousin Chichester Bell and CharlesSumner Tainter built a rotating cylinderwith a wax coating that used to respond toincoming sound pressure. The inventionpaved the way for the first recordingmachine, the "Dictaphone", patented in1907.
- In 1952 Bell Labs came along “Audrey”the Automatic Digit Recognition machine.
- Audrey was not the only kid on the block,though. In the 1960s, several Japaneseteams worked on speech recognition, withthe most notable ones a vowel recognizerfrom the Radio Research Lab in Tokyo, aphoneme recogniser from Kyoto University, and a spoken-digit recognizer from NEC Laboratories.
- All the way from Acoustic-Mechanical Speech Machine by Wolfgang von Kempelen till today we have come a long way. Today we have many advanced Virtual Assistants like Google Assistant, Amazon’s Alexa, Apple’s Siri, Samsung’s Sam, etc.

3. System Analysis

3.1 Proposed System

- Subroutines may be defined within programs, or separately in libraries that can be used by many programs. BRIAN is a program that will run as a subroutines. This subroutinewill responds to the user’s voice.
- This subroutine uses python’s speech recognition module for recognising the user’s voice.
- Subroutine uses google text-to-speech module for the conversion of text to voice.
- Our project provides hands free usage of the system, it responds to the voice, and performs basic functionality like file search, file opening, give details about date, time and system performance etc.

3.2 Methodology

The B.R.I.A.N. assistant performs the task given to it via speech. To make the assistant do the task the system must be awakened first. Once awake the systemwill wait for the command which the userwishes to run.

- To awake the system, user has to speak awake-word like “BRIAN” in this case.User needs to speak hiscommand/request/question just afterhe/she has said the wake-word.
- After waking the assistant the system willwait for a certain time before going tosleep again.During that waiting time the user cancommand the assistant without using the awakening word. After the specified time the system goes to sleep and wait for user to awake it.
- The wake-word initiates the voice recognition tool and all the words spoken after the wake-word gets accepted by the voice recognition tool. This tool then analyses the input and bifurcate it as a request or question or command.
- After analyzing and understanding of the input, the software gathers the related information about the query within the system, and the system can also search the internet if the query specifies it. As soon as the tool gathers the necessary and relevant information, the system presents the desired information to the user via speech.

4. System Design

Block Diagram:

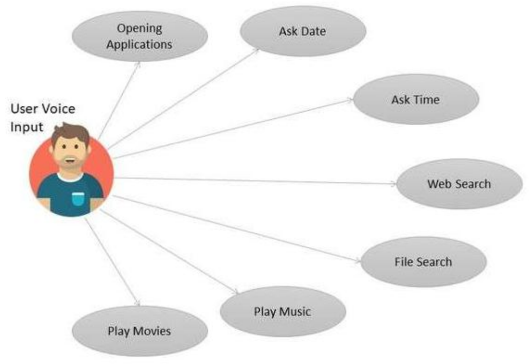

Fig 1: Block diagram of BRIAN

Fig 1 represents the functionalities of BRIAN. Based on the user inputs, BRIAN can do the following tasks:

- Open applications
- File search
- Open files
- Play music and movies
- Wikipedia search

5. System Implementation

5.1 What are BRIAN's features?

In this system, skills such as greet, date time, CPU utilization statistics, local file search, web search and opening local files. All these commands are processed in English language only.

The assistant utilizes the text-to-speech module to interact with the user and speech recognition module to listen the user's commands.

- Greet:
  The command "Hello" or even the wake word "Brian" would activate this function. As soon the wake word is used the system greets the user according to the time of the day.
- Date time:
  The "Date" and "Time" commands respectively can be used to get the date and time of the running system. The system informs the user on the date and time of the system while the system interacts with the user.
- CPU utilization statistics:
  "CPU stats" is the command used to get CPU utilization statistics of the user's system. This skill comes in handy to the user by just commanding the system to get CPU utilization statistics instead of manually checking for it.
- Local file search:
  The assistant would search for the file through the command "Search" followed by the filename. As the assistant focusses on local lightweight usage, a local file search through voice commands would prove more time efficient than manually searching for it.
- Wikipedia searching
  The assistant would search for the file through the command "Wikisearch" followed by the search name. The assistant is capable of searching the user requirements through the internet too instead of just looking for them in local system only. Through this the assistant would return a brief summary of the search carried on by the user.
- Opening files:
  Any local files stored on the system could be opened/run through the command "Open" followed by the filename. The assistant can open/run any sort of file irrespective of the file type though vocal command of the user.

5.2 Python Module

- speech_recognition module:
  Speech recognition helps us to save time by speaking instead of typing and this module gives power to communicate with our devices. In python there are 4 speech recognitions libraries
  i. Emu Sphinx
  ii. Speech recognition
  iii. Kaldi
  iv. Wav2 letter ++

  In this project we have used speech Recognition library. It is a library for preparing speech recognition, with support for several engines and APIs, online and offline. The library has many classes.
- Wikipedia module:
  When we retrieve the information form various sources it's called Data Scrapping. Wikipedia provides information. Python's Wikipedia module (or API) to scrap the data from the Wikipedia pages. This module allows us to get and parse the information from Wikipedia. The function used id wikkipedia.summary.
- Webbrowser module:
  This provides a high level interface which allows displaying web-based documents to user. The webbrowser module can be used to launch a browser in a platform independent manners.
- Import webbrowser

  Webbrowser.open (http://www.google.com)

  This opens the request the requested page using the default browser

  To have a bit more control over how the page get opened, use of the following functions like
  i. Open new browser
  ii. Open new tab

  To open specific browser we use webbrowser.get().
- Os module:
  This module in python provides function for interacting with the operating system. Os comes under python's standardutility modules. Third module provides a portable way of using operating system-dependent functionality the *os* and *os.path* modules include many function to interact with file system.
  i. Os.walk- finds the directory
  ii. Os.path-finds the path
  iii. Os.path.join- will join two paths.

- datetime module:
  Datetime is kind of a combination of date and time. It can represent all the attributes from date and time objects. Let's hop onto the examples to make it clearer. The datetime object provides the flexibility of using date only, or date and time combined. There is a hierarchy among the parameters starting from year and going down to microsecond level.
- re module:
  Regex functionality in Python resides in a module named re. The re module contains many useful functions and methods, most of which you'll learn about in the next tutorial in this series. For now, you'll focus predominantly on one function,
  i. re.search().re.search(<regex>, <string>): Scans a string for a regex match.re.search(<regex>, <string>) scans <string> looking for the first location where the pattern <regex> matches. If a match is found, then re.search() returns a match object. Otherwise, it returns none.
  ii. re.search() takes an optional third <flags> argument that you'll learn about at the end of this tutorial.

- pyttsx3 module:
  pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the pyttsx3.init() factory function to get a reference to a pyttsx3. Engine instance. it is a very easy to use tool which converts the entered text into speech.
  The pyttsx3 module supports two voices first is female and the second is male which is provided by "sapi5" for windows. It supports three TTS engines:
  i. sapi5 – SAPI5 on Windows
  ii. nsss – NSSpeechSynthesizer on Mac OS X
  iii. espeak – eSpeak on every other platform

- getpass module:
  getpass() prompts the user for a password without echoing. The getpass module provides a secure way to handle the password prompts where programs interact with the users via the terminal.
  This module provides two functions:
  i. getpass():
  getpass.getpass(prompt='Password:', stream=None)
  The getpass() function is used to prompt to users using the string prompt and reads the input from the user as Password. The input read defaults to "Password:" is returned to the caller as a string.
  ii. getuser()
  getpass.getuser()
  The getuser() function displays the login name of the user. This function checks the environment variables LOGNAME, USER, LNAME and USERNAME, in order, and returns the value of the first non-empty string.

6. Flowchart

The entire working of BRIAN is explained using the 2 flowcharts shown bellow. First flowchart (Fig.2) explains the major working principle of BRIAN. Second flowchart (Fig.3) explains the working principle of Launch Action Manager.

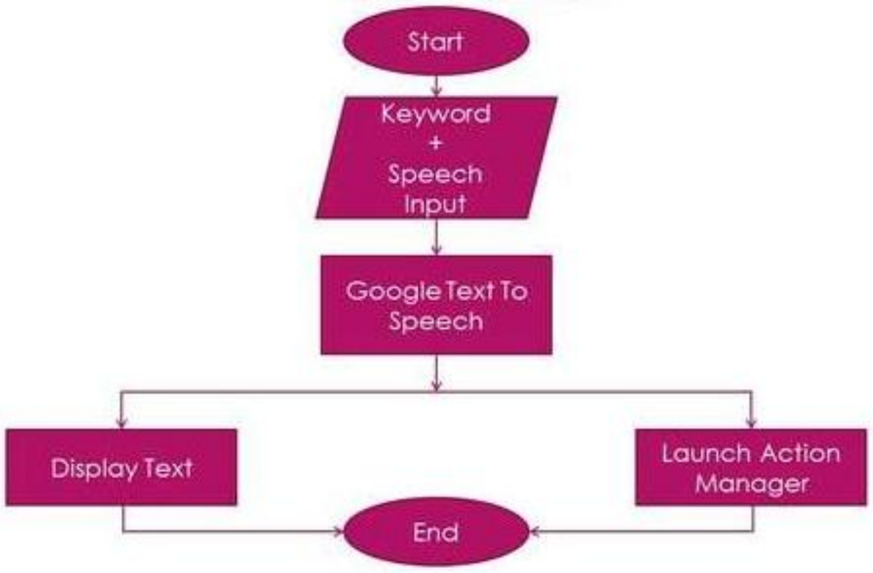

Fig 2: Flowchart of BRIAN

The above flowchart explains the basic working principle of BRIAN. First when the program is executed, it runs as a subroutine. Once the subroutine starts, the program is in sleep state. BRIAN has to be called to wake state using the word "Brian". Once the BRIAN is awake, we can give input to the BRIAN in the form of voice command. Once the voice command is given, that command is sent to Python Text-to-Speech module for converting the voice input to text. Now, the converted text is displayed on the screen and the converted text is sent to "Launch Action Manager". The "Launch Action Manager" processes the given input and performs the given task (given through input). Once the tasks given by users to BRIAN are completed, the BRIAN goes to sleep state. We can close the subroutine of BRIAN by giving the close command.

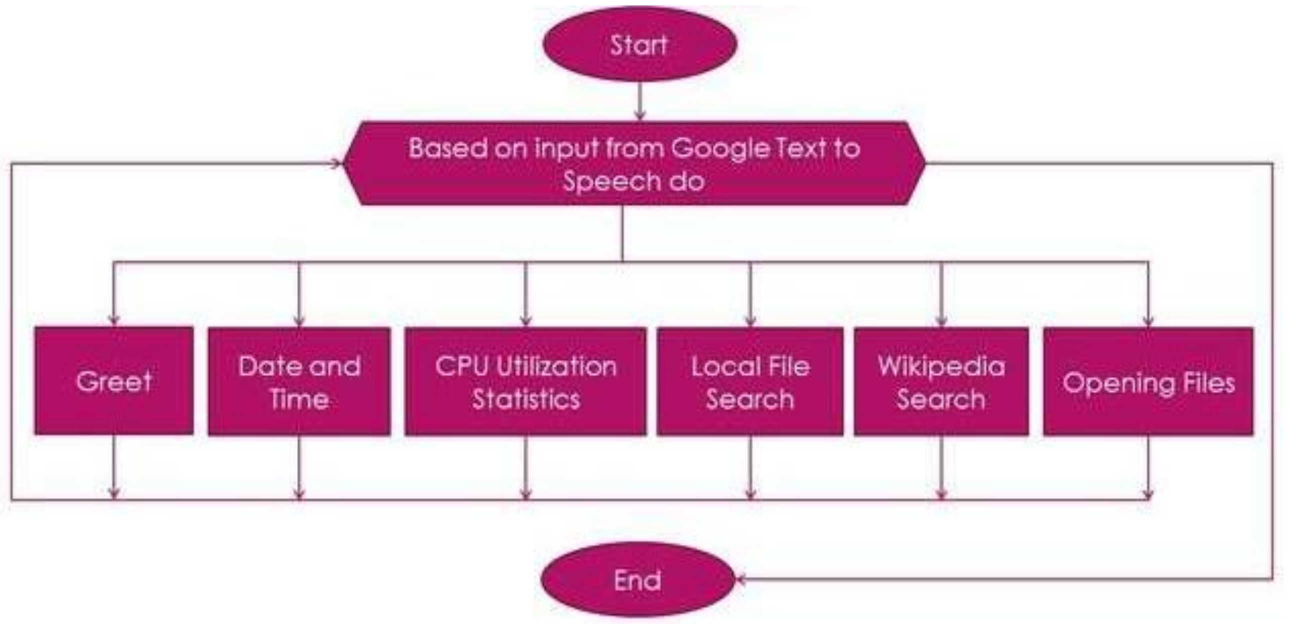

Fig 3: Flowchart of Launch Action Manager

The above flowchart explains the basic working principle of the "Launch Action Manager". The given voice command is converted to text using Python Text-to-Speech module. The converted text is given as input to Launch Action Manager. Based on the input, operations like file search, Wikipedia search, giving information about CPU Utilization Statistics, etc., are performed. Launch Action Manager Works like a loop. Until it gets a sleep or exit command, it executes continuously. Once it gets such command, it stops its execution which leads to the termination of the entire subroutine of BRIAN.

7. Algorithm

In this system, skills such as greet, date time, CPU utilization statistics, local file search, web search and opening local files. All these commands are processed in English language only.

Step 1: Authenticate and GREET

The command "Hello" or even the wake word "Brian" would activate this function. As soon the wake word is used the system will turn on the camera and it will authenticate the user. If authentication is successful, user is greeted else the program terminates.

Step 3: Voice Input

After the system greets, the system takes the voice input given by the user.

Step 4: Conversion

Once the input is received in the form of voice command, the system uses the Python Text-to-Speech to convert the voice input to text.

Step 5: Launch Action Manager

Once the voice command is converted into text, the resultant text is used as input for the Launch Action Manager. Based on the input given by the user, system performs the following tasks:

- **Command "Date" and "Time":** The "Date" and "Time" commands respectively can be used to get the date and time of the running system. The system informs the user on the date and time of the system while the system interacts with the user.
- **Command "CPU stats":** "CPU stats" is the command used to get CPU utilization statistics of the user's system. This skill comes in handy to the user by just commanding the system to get CPU utilization statistics instead of manually checking for it.
- **Command "Search":** The assistant would search for the file through the command "Search" followed by the file name. As the assistant focusses on local lightweight usage, a local file search through voice commands would prove more time efficient than manually searching for it.

- **Command "Wikisearch":** The assistant would search for the file through the command "Wikisearch" followed by the search name. The assistant is capable of searching the user requirements through the internet too instead of just looking for them in local system only. Through this the assistant would return a brief summary of the search carried on by the user.
- **Command "Open":** Any local files stored on the system could be opened/run through the command "Open" followed by the filename. The assistant can open/run any sort of file irrespective of the file type though vocal command of the user.
- **Command "Sleep":** The command "Sleep" is used to close the program temporarily. BRIAN goes to sleep mode if the BRIAN doesn't receives any command from the user for 1 minute (or 60 seconds). The BRIAN program becomes active if it is called by using the command "Hello".
- **Command "Exit":** The command "Exit" is used to bring the program completely out of execution. This command terminates the execution completely. To restart this program, we have to start the BRIAN program manually.

Step 6: Repeat step 4 until the user gives the "Sleep" command or "Exit" command.

Step 7: Exit Program

Once the user gives the command as "Exit", the program closes permanently or by turning off the computer, the program closes by stopping its execution.

8. Conclusion

Today, there are many virtual assistants in the market. Few of them are Amazon's Alexa, Google's Assistant, Apple's Siri, Samsung's Sam, etc. All these virtual assistants are developed in foreign countries. BRIAN (Basic Response Interaction and Assistance Network) supports the "Make in India" Initiative as it is developed by us, Indian students.

Data is the new gold. The major problem in today's world is the data privacy. We can see many companies that produce these virtual assistants use user's data to provide the service. Some collect minimum data to provide the service whereas some companies collect user's data which are not needed for providing the service. These data are used for many purposes like personalized ads, improving the service, influencing local elections, etc. Companies earn billions of dollars out of these data. This threat doesn't stops here. In today's world we are seeing many cyber threats. Many black hat hackers hack into these company's data centers, servers, etc., and ask ransom from the companies or they will leak those data in the dark web for money. This is a major threat to the users and the company. When it comes to BRIAN, it doesn't collects user's data at all. Majority of the process takes place within the local machine itself. So when there's no data collection from the users, there's no problem of data privacy. Hence the problem of data privacy doesn't arises here.

BRIAN (Basic Response Interaction and Assistance Network) supports Windows laptops which can be integrated in other operating systems like Linux based operating systems, Mac OS, etc., in future. It can be incorporated in hand held devices like smartphones with different operating systems. In future, it can also be integrated in voice assistant devices which can be used for controlling IoT devices which has a promising future in the market. For more convenient use, BRIAN can be implemented with regional language feature as future enhancements.

Acknowledgements

Any accomplishment achieved does not rest entirely on individual endeavors but on the direction, inspiration, and support of intellectuals, guides, and teachers. Numerous personalities, in their own aptitudes, have helped us in carrying out this project work. We intend to take an opportunity to thank them all.

At the very outset, we owe our gratitude to our beloved Chairman, Sri Vasu V and Secretary, Sri Kaveesh Gowda V of Vidya Vikas Institute of Engineering and Technology for their encouragement and support in all our Endeavors.

We would like to express our colossal gratitude to Dr.Ravishankar M, Principal, Vidya Vikas Institution of Engineering and Technology, Mysuru, for his help and inspiration during the tenure of the course.

We also thank Dr.Madhu B.K, Professor and Head of Department of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, for his support.

We would like to express our heartfelt gratitude to our guide Prof.Drakshayini K.B, Asst. Professor, Department of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, for her timely advice on the project and regular guidance throughout the project-work.

We thank our project coordinator Prof. Suman Jayakumar, Asst. Professor, Department of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, for her encouragement.

We also extend our sincere thanks to all the faculty members of Information Science and Engineering, Vidya Vikas Institute of Engineering and Technology, Mysore, for their constant encouragement and backing.

References

- International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 10, Number 1 (2017) © International Research Publication House http://www.irphouse.com.

Title: "Personal Assistant with Voice Recognition Intelligence"

- Real World Applications of AI Virtual Assistants https://kambria.io/blog/real-worldapplications-of-ai-virtual-assistants/
- Virtual Assistant Apps: Benefits and Abilities https://agilie.com/en/blog/virtual-assistantapps-benefits-and-abilities