AI VOICE ASSISTANT ANDROID APP USING KEYWORD MATCHING AND NLP

Ms. Kuramana Yasodha
Student
Master of Computer Applications
Andhra University College of Engineering
Visakhapatnam, India
Email: kuramanayasodha9581977919@gmail.com

Mr. M. Bala Naga Bhushanamu
Assistant Professor
Department of Computer Science
Andhra University College of Engineering
Visakhapatnam, India
Email: balu91@gmail.com

Abstract: Human voice interaction with a device is revolutionized in this day of digital world to make use of voice assistants like Google Assistant, Amazon Alexa and Apple Siri; as a result, the user can communicate with these devices by entering voice commands, and hence interaction with the device using hands is wavered. But these commercial helpers highly rely on the availability of uninterrupted internet and processing in the cloud, which makes them all but useless out of reach to the internet or where connectivity is poor. The proposed project attempts to deal with this limitation by proposing an Android-based AI voice assistant that has been built with Java in Android Studio and is meant to operate offline with a lightweight rule-based system. This assistant is able to perform important functions of the smartphone-calling, sending messages, alarm, photo taking, and opening apps-without using the internet. It uses the native Google Android API SpeechRecognizer to handle voice input, and Text-to-Speech (TTS) to show feedback as well as a minimal say-one-thing-get-another Natural Language Processing (NLP) pipeline consisting of tokenization, normalization and pattern matching. The detection of commands is done based on predetermined keyword formations that infer minimal calculations and stability on under powered devices. The assistant mainly offline though supports the optional online capabilities of weather and Wikipedia queries when they are connected to a network, rapidly switching modes depending on connectivity. Its user interface remains simple and open to access, and it provides voice feedback input and feedback in visual and sound categories. It is worth noting that everything is done locally so that the response is timely and privacy is enhanced relative to the cloud-based systems. The present project shows that primordial to the use of AI and NLP, the proposed voice assistant is made functional and safe by covering all areas of efficiency that a voice assistant should have, it can be made user-friendly, and with the resources available and common in rural, developing, or connectivity-limited areas, a feasible and optimized voice assistant can be fashioned.

Keywords: Artificial Intelligence, Text-To-Speech, Speech Recogintion, Java, XML, Android Studio, Natural Language Processing (NLP) Techniques: Tokenisation, normalization, pattern matching, keyword matching, rule-based decision making.

I. INTRODUCTION

This AI Voice Assistant project offers a low profile, performance, and privacy-aware solution to popular voice-controlled systems by emphasizing off-line functionality and resource-use. In contrast to the assistants with AI models based on the internet and remote cloud servers, this application conducts all commands locally through the mixture of rule-based AI and simple NLP methods. Fundamentally, the assistant can detect voice command entered with Android SpeechRecognizer and tokenize it followed by normalization to find commands using pattern recognition and keyword matching. Every command has its orderly route: after the voice signal gets transcribed into a written text, the system looks through keywords and phrases indicating an action, such as call, message, alarm, open or capture. They are compared with the pre-configured functions to activate Android intents- phone call, messaging, starting the camera, etc. Regular expressions increase the performance of the assistant to parse parameters of the command, and the TextToSpeech engine will give a clear voice response to verify the actions completion.

The navigation of the app is also modular which allows simple extension and customization of commands. Being not computationally expensive, it can be run on older or entry-level Android devices. Besides, although the current version of the app is mostly intended to work in offline access, there are some selectively online-enabled elements of the app that are currently supported in the case of a network connection. This two-mode feature is like a best of both worlds- making sure that there is reliable performance in disconnected set ups but having the added functionality when connected to the internet. All of it considered, the voice assistant project can illustrate the ability to integrate the core AI principles, simplified NLP, and built-in Android APIs that can be assembled into a versatile yet user-friendly product that is easy to implement in a low-resource setting.

II. LITERATURE REVIEW

The voice assistants have evolved from early days when they were rule-based assistants to hybrid and deep learning

assistants with a wide range of connection dependence on internet connectivity.

[1] Weizenbaum (1966) came up with a finding on rule-based conversational agents, ELIZA. This agent used basic pattern matching, keywords and trigger responses to resemble conversation with humans. It did not have any actual understanding but still worked entirely without being connected to the Web and provided the basis of voice assistants that use rules.

[2] Microsoft (2000) launched the Speech API (SAPI), which allowed the creation of desktop applications that could support offline voice activities. The system also had keyword recognition, which would correspond user speech and execute commands and was popular mainly because of its efficient and local processing nature.

[3] In 2011, 2012, and 2014, Apple Siri, Google Now and Microsoft Cortana banked on cloud-based NLP to introduce their smart voice assistants. Although being smarter and able to recognize more complex queries, they were based on detection of specific keywords to be used along the major functionalities. Most of the tasks with these assistants though needed constant access to the internet.

[4] Myer and Tomar (2018) recently proposed a Time Delay Neural Network (TDNN) on embedded feature to spot the keyword. Their two-tier model was both lightweight and optimised offline voice command recognition and was aimed at devices that have low computer resources and could include mobile and IoT devices.

[5] Choi et al. (2019) suggested Android voice assistants with a Temporal Convolutional Network (TCN). Their model provided a better performance in terms of speed and efficiency compared to conventional convolutional networks, therefore, enabling successful organizational offline wake-word detection and thus it can serve as the best contender to be incorporated into mobile applications.

[6] The use of voice in Android smart phones was under development by Mahajan et al. (2019) to create voice-enabled Android assistant on behalf of the visually impaired users. Whenever the assistant needed to perform basic tasks such as calling and navigation, it relied on cloud-based speech-to-text services with a feedback keyword match. Nonetheless, it required full use of internet connection to recognize the speech.

[7] Rani et al. (2021) engaged a virtual assistant, which is the online speech recognition API of Google. It used simple logic of keywords of task execution in form of an if-else statement like reading time or calls. Although it was simple in the functional sense, it did not have an offline capability at all.

[8] An assistant developed by Vedika Jain et al. (2021) was fitted to Python with an API (Speech Recognition Service API) offered by Google that enabled speech-to-text conversion and performed a task based on the rules of matching keywords. It had the ability to turn on websites, display time but need internet connection, in order to identify it.

[9] Patil and Patil (2021) created a chatbot that could be used in academics on the RASA framework. The intent classification and entity extracting systems were used on training data which was loaded with keywords. Although it made accuracy higher, it had to be continually online and needed machine learning infrastructure.

[10] K. Gupta et al. (2021) built an android voice assistant in Java programming language and Google speech API. The system practiced the use of keyword matching and conditional logics in task execution but nonetheless it entirely depended on cloud-based voice recognition, a factor that limits offline application.

[11] Basha and Anuradha (2022) presented a hybrid assistant that allows recognizing speech offline with Vosk and in the cloud with the help of the API of Google. The app performed a keyword matching to attend actions and maintained the service during connection variations.

[12] Siddiqui et al. (2023) suggested involving rules and keyword matching to use an Android voice assistant offline and more advanced cases at the cloud level.

[13] Vu et al. (2023) created the Android equivalent of a voice assistant, Voicify, which is a deep learning voice assistant. It pulled out the UI features through voice commands and enabled users to control mobile applications without any need to use hands. Most processing was local but there was a need to update metadata and command handling periodically online.

III. EXISTING SYSTEM

The current voice-based assistants have been transformed into online solutions through the mobility-based technology. A desktop-based assistant has been created on the basis of Python libraries, including SpeechRecognition and pyttsx3, as well as cloud-based assistance, Google Speech-to-Text. Such systems utilized a multi-stage pipeline that included wake word, speech recognition (ASR), natural language understanding (NLU), dialogue management, and output (text-to-speech (TTS).

Then the voice assistants became compatible with Android mobile devices to provide more convenience and portability. Most of these Android applications enabled the user to make calls, send messages as well as search through a voice and these were all made possible by SpeechRecognizer and Assistant APIs provided by Google. Although these mobile systems made access and use on the move reasonable, the systems still required extensive use of the internet so that the systems could process commands. It means that they were not good in offline settings and could not be flexible or customized, depending only on third-party cloud services

IV. PROPOSED SYSTEM

The offered system is a voice assistant with artificial intelligence running offline on Android with the implementation of Java through Android Studio whose technology is based on a keyword-based call method in identifying and actioning user instructions. In contrast to classic desktop-based voice assistant and modern mobile assistants based on cloud capabilities, this one both transcribes

voice input directly in hardware with the help of the native Android SpeechRecognizer and locally executes the tasks through Java-based Android API packages, including AlarmManager and SMSManager.

- To use the SpeechRecognizer of the Android system to do live speech-to-text translation and apply simple NLP methods to identify any keywords.
- Such that voice controls the core Android operations such as alarm clock set up, sending SMS, starting a call, using a camera, and getting access to locations, and accompanied with both visual and audible feedbacks.
- To facilitate online functionalities to display weather (on multiple locations using OpenWeatherMap API), news, headlines (using News API) and general queries (using Wikipedia API) when network coverage is provided.
- To keep history of user interaction.
- In order to facilitate accessibility by implementing an option of text input, so that the assistant can be used by deaf or hard of hearing people.
- So as to impose security and privacy by means of managing permissions properly.
- It should also have good offline support so that people can make a call, send a text message, set an alarm, make use of a camera, and so on without being connected to the internet.

V. TECHNOLOGIES USED

The power of the application of this AI Voice Assistant (Android) is achieved through the synergistic combinations of powerful, traditional applications tools, native Android APIs, and low-intensity processing techniques to support offline processing, high-speed responsiveness, and cross-compatibility.

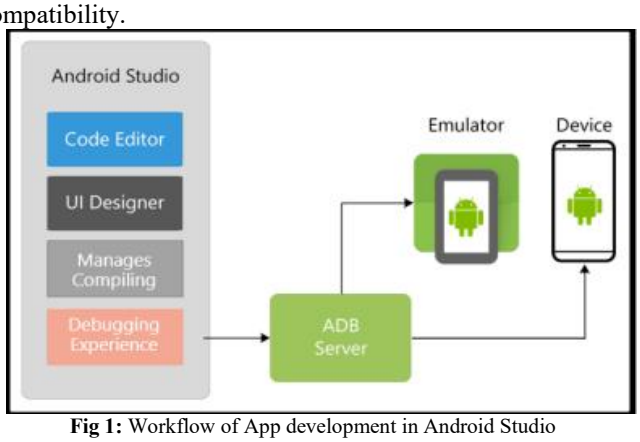

Fig 1: Workflow of App development in Android Studio

- ***Java:*** Java is an object-oriented programming language which allows developer to design the logic, flow control and interaction with the Android system within the app.
- ***Android Studio:*** Android studio is the official android application development IDE that allows development of, testing and debugging of android applications. It offers tools such as emulators, layout editors, and Gradle to do development in an efficient way.
- ***AndroidSDK & APIs:*** The SDK& APIs applied are Android SDK and Intent APIs to utilize native mobile features in the app. It also uses online APIs for providing various information like wikipedia, openweathermap, news APIs.
- ***Regular Expressions (Regex):*** A Regex can be applied to identify a pattern like time, date or keywords in user commands to make the parsing of these commands simplified.
- ***XML:*** XML provides access to developer to create layout and interface elements of the app such as the buttons and input fields.
- ***Text-To-Speech (TTS) engine:*** The native TTS engine of android will process the text response into a spoken format giving the audible response.
- ***SpeechRecognizer API:*** This API records and translates what the user is said into text format for future processing.

VI. METHODOLOGY

The process embraced in this Android software AI Voice Assistant aims at integrating basic Natural Language Processing (NLP) and keyword-based intent detection in the context of determining the meaning behind the user commands. The main work cycle is the transformation of speech to the text with the help of the speech recognition apparatus which is built-in as a part of Android operating system. The app then applies tokenization, keyword detection, pattern matching to get intention in the user input. According to this established desire, related activities are aligned based on Android system APIs and the overall feedback is given through Text-To-Speech (TTS).

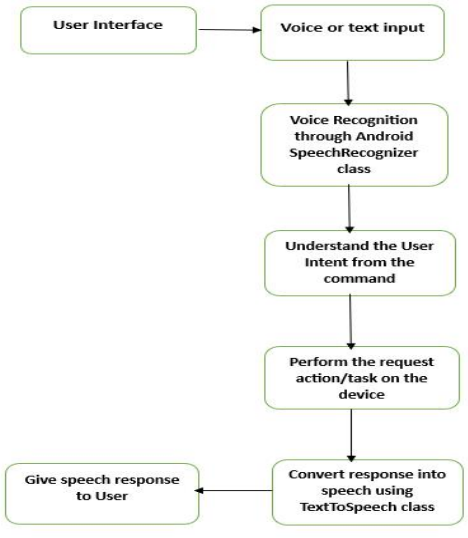

Fig 2: Workflow of Android Voice Assistant

A. Speech to Text:

The application takes the help of the inbuilt SpeechRecognizer and RecognizerIntent classes provided by Android platform with the help of which audio input is obtained in written form. On clicking on the microphone button, the app asks for the audio recording permission and proceeds to open the speech recognition screen.

B. Normalization of Text:

The normalization of the input is made after the receivable text of the speech recognizer or the input field. This includes capitalizing text to lowercase to allow non-case matching of keywords, strip leading and trailing whitespace and normalize the text to gain a more convenient pattern search. Normalization confirms that the command is not influenced by the differences as to how the users are speaking or typing.

C. Tokenization and Keyword matching:

The normalized text has implicit tokenization based on the string operations. The app looks to see the keywords or phrases like the word call, the set alarm, the sending SMS, the word weather, or the opening of the YouTube. The app can determine user intent using rule-based approach without using heavy or complex NLP or machine learning. This lightweight module makes it faster to process on low-resource devices.

D. Regular Expression Pattern Matching:

In case of structured commands such as setting an alarm, the app employs regular expressions. As another example, the regex ALARM_PATTERN retrieves the hour, minute, and period (AM/PM) in a command such as: “set alarm at 6:30 AM.” Such an exact matching allows an accurate parameter extraction which is required during the scheduling of alarms.

E. Rule Based and Intent Recognition Decision Making: After the input is compared with particular keywords or patterns, the app applies rule-based decision in order to detect the intent. All the intents are tied to a certain mode of operation: calling launches the contact picker, SMS initiates message interface, alarm establishes time, even activity such as weather, news, Wikipedia, or maps.

F. Execution of Action with Android APIs:

After reading out the intention of the user, the app will perform the action upon using the Android APIs. As an example, it can schedule alarms via AlarmManager, , send text messages via SmsManager, launch other apps with the Intent. Everything is closely connected with the system of permissions and intent of Android to guarantee safe and effective performance. Error handling is also installed to cope with such cases as missing permissions or services.

G. Response Generation of Text-to-Speech:

Once an action is done with a command, the assistant verbally responds with the help of the TextToSpeech built-in engine in Android. This makes the user interaction stronger as actions are confirmed on an audible level

H. Management of History:

To make it easier to use and track, every interaction (both what the assistant responds with and what are the users responding with) is logged in a list of history. This history is presented by custom UI dialog, so that user can see previous commands and answers.

VII. USE CASE DIAGRAM

.

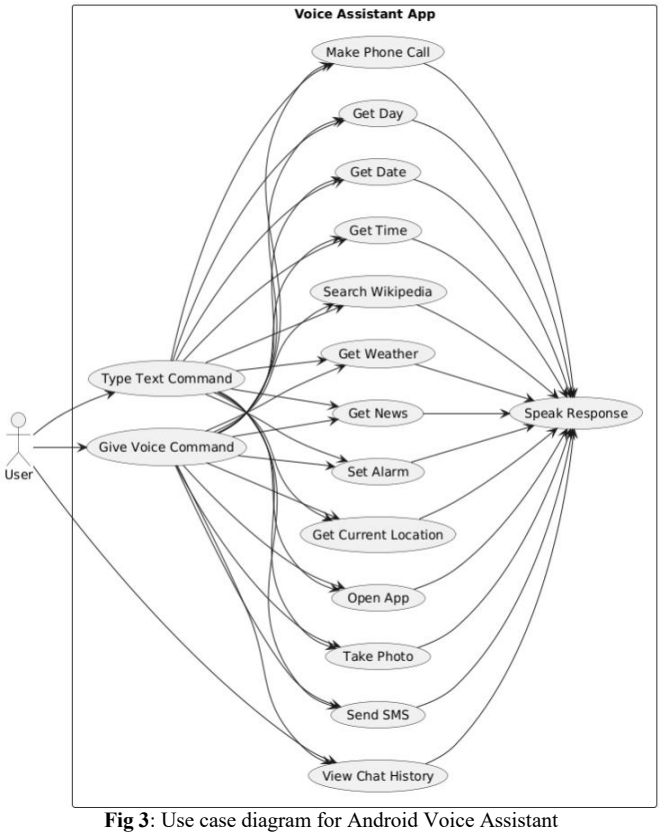

Fig 3: Use case diagram for Android Voice Assistant

VIII. RESULTS AND ANALYSIS

The Sophia AI Voice Assistant was tested using numerous parameters such as the precision of the command recognition, promptness of the execution, offline usage, resources and the user satisfaction.

A. Functionality-Testing:

The offline functionality that the assistant was tested in core of the functionality comprised the following:

Table I: Sample Offline Test cases

Function	Test Command	Expected Result	Output
Initiate call	“Call [contact name]”	Opens dialer and calls specified person	Successful

Function	Test Command	Expected Result	Output
Send SMS	"Message to [contact]"	Sends SMS to specified person with message specified	Successful
Alarm	"Ring alarm at 7 AM"	Rings alarm at specified time	Successful
Get time/date	"What is the time or date?"	shows the current time/date	Successful
Launch app	"Open Camera" / "Open WhatsApp"	Starts the asked application	Successful
Photo	"Capture an image"	Camera is opened to take photo	Successful
Offline TTS Response	Any command like "Hi"	Speaks preloaded answer	Successful

It could be recognized that the system could accept 95-100 per cent of the voice commands that were related to the offline functions in the case of quiet environment.

B. Precision of Speech Recognition:

- Quiet Environment: Per Cent accuracy of speech recognition was 96 -98 on tested indoors with minimum sounds.
- Medium Noise: With the noise level, when there was a little amount of fan noise or there was a distant conversation, accuracy reduced by a small percentage of 85 – 90.

C. Response period:

Table II: Average response time of assistants

Operation	Mean Response Time
Voice to text conversion	0.8 – 1.2 seconds
Command parsing & matching	Less than 0.5 seconds
Execution of Action	1.0 – 2.0 seconds

The overall average time of response varied between 2 and 3 seconds, making the assistant swift enough to be of a real-time interaction.

D. Efficiency and consumption:

- RAM memory: memory-averaged: 4060 Mb.
- Consumption/battery: Usage low during short periods (draining less than 3 per cent in 30 minutes of use).
- App size: The size of the in-built APK file was less than 20 MB as there were no external ML libraries and cloud SDKs.

E. Internet-enabled Online Feature Performance:

All the online attributes were reliant on the availability of the network and exhibited a little slower latency (5-7 seconds).

Table III: on-line sample test cases

Control	Command	Output	Result
Weather Info	"What's the weather in Visakhapatnam?"	Retrieves real weather data	Successful
Wikipedia Search	"About India on Wikipedia"	Reads brief intro from Wikipedia	Successful
News Headlines	"Get latest news"	Reads first 5 news headlines	Successful
Location Finding	"Where do we find Taj Mahal?"	Google Maps Launches	Successful

F. Examples of Figures:

To demonstrate in a representative way how the work of Sophia AI Voice Assistant and its interface is carried out, a series of screenshots were made during the various tasks of realization.

Fig 4: UI Design

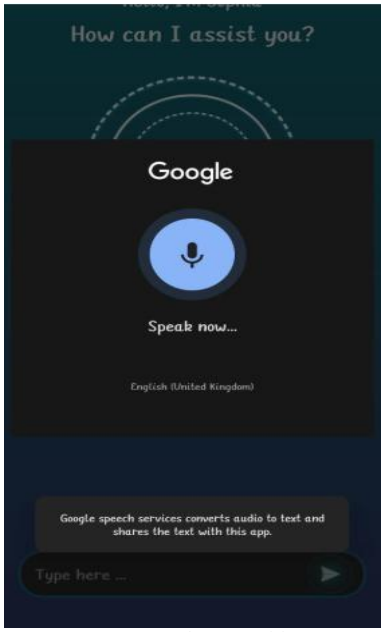

Fig 5: Speech Recognition

CONCLUSIONS

The android app that implements the AI Voice Assistant is a Java program that was designed in Android studio since it offers a low-weight, secure and effective voice interface framework. It is based on rule-driven AI as well as relatively uncomplicated NLP techniques such as tokenization, normalisation & pattern matching to comprehend the user commands. Key tasks of calling, messaging, setting of the alarms and opening apps can be performed offline despite having all offline features turned off. It will be suitable to users in low connectivity regions or those who appreciate privacy due to its offline feature. Keyword matching and regular expressions execute the commands fast, deterministically, and with minimal resource consumption. The application is playable on even low-end android devices. It runs on an easy to use and understand interface and has no requirements on cloud storage yet presents a session-based history tracking. Offline commands take 1-2 second and online features such as weather updates are a little slower. It can easily operate by incorporating Android life such as SpeechRecognizer and TextToSpeech. All in all, the application is simple enough and functional at the same time to provide a stable offline-first voice assistant experience.

FUTURE SCOPE

There is the AI Voice Assistant app which is already quite effective in connecting with users offline in voice mode, but there is huge opportunity in improvement. The future enhancements may include the support of multiple languages, the offline language packs, as well as personalization of the voice commands. Other capabilities such as an offline knowledge base, hotword detection, gestures and safety features such as emergency mode, chat history export and many more would increase its functionality immensely. These advancements are supposed to make the app a better, more varied, smarter, and more accessible helper, particularly to offline and privacy-oriented settings.

REFERENCES

[1] J. Weizenbaum, 1966. ELIZA -A Program to Study Natural Language Communication between man and machine. 18 (9):36, 45), 1975.

[2] Microsoft Corporation, 2000. SAPI Microsoft Speech API. Microsoft DevNet.

[3] Apple Inc., Google LLC, and Microsoft corp., 2011-2014. Siri, Google Now and Cortana: Voice applications on Smartphones. Different product documents and releases.

[4] Myer A. and Tomar D., 2018. Time Delay Neural Network with an efficient Keyword Spotting in the Embedded Systems. International Journal of Computer Applications 182 (48), pp. 1

[5] H. Choi, J. Kim and S. Lee, 2019. Lightweight Temporal Convolutional Networks on Keyword Spotting in Mobile Devices. Interspeech, 1-5.

[6] A. Mahajan, P. Patil, S. Kale, 2019. Android Voice Enabled Application (The Visually impaired). International Research Journal of Engineering and Technology (IRJET). 2020, 6 (4) Article (3294-3297).

[7] Rani, S., Verma, M., and Goel, A., 2021. Android Based Voice-Controlled Virtual Assistant. International Journal of Engineering Research & Technology (IJERT), Vol.10, No. 5, p.108- 112.

[8] V. Jain, S. Desai and K. Shah, 2021. Automate Python-Based Personal Voice-Based Assistant. International Journal

of Research in Engineering, Science and Management; Vol. 4, No. 6; pp. 241-244.

[9] P. Patil and P. Patil, 2021. Chatbot or RASA College Enquiry Chatbot. International Journal of Engineering Research and Technology (IJERT), Vol. 10, No. 3, 234- 237.

[10] K Gupta, R Jain, and S Verma, 2021. Voice Assistant Android-Based Application Google Speech API. International Journal of Computer Sciences and Engineering 9 (2) 34-39.

[11] Basha, S. and Anuradha, V., 2022. Andoid Voice Assistant (Hybrid) Vosk Google Speech API. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 8: 4, 98-103.

[12] Siddiqui, I., Kumar, A and Mehta, R., 2023. Andro VArray I.C.H. - Meio-Nome Cele Mil. 2011International Journal of Innovative Technology and Exploring Engineering (IJITEE) No.12, Vol.1, pp.110-115.

[13] T. Vu, M. Le and D. Tran 2023. Voicify: Deep Learning based Mobile Voice Assistant. In: Proceedings of the IEEE AIMC Conference, pp. 67 72.