Voice Activated Systems

Seminar Paper

Sachin Bhalekar

Sumit Jawale

Abstract

A voice activated system is a system that performs actions and is controlled by means of the human voice. Speech is the primary means of communication between people. By removing the need to use buttons, dials and switches, consumers can easily operate appliances with their hands full or while doing other tasks. Today speech technologies are commercially available for a limited but interesting range of tasks. These technologies enable machines to respond correctly and reliably to human voices and provide useful and valuable services. Speech-based user interfaces represent one of the next major disruptions in computing and provides you with an opportunity to take advantage of this new form of interaction.

"Voice is the most natural form of user interaction, and recent advancements in speech recognition and natural language understanding technology are now enabling us to control our connected devices with speech. We believe that voice can simplify the modern lifestyle, providing us with a unifying interface across devices so we can be entertained, stay informed, and organize our lives by just asking." - Amazon

Table of Contents

Introduction and Background

Statement of Problem Area

To Integrate voice capabilities into existing systems. Providing voice commands as an additional feature of interaction to existing system.

Previous and Current Work, Methods and Procedures

❖ Previous and Current Work:

Speech recognition development started from early 1877 and kept improving till date.

Period	Key Development
1877–1971	Speech recognition is at an early stage of development. Specialized devices can recognize few words and accuracy is not very high.
1971–1987	Speech recognition rapidly improves, although the technology is still not commercially available.
1987–2014	Speech recognition continues to improve, becomes widely available commercially, and can be found in many products.

Table 1: Overview of speech recognition development.

A brief timeline on how the work got improved is given below:

- In 1877, Thomas Edison's phonograph becomes the first device to record and reproduce sound. Later in 1879, he invents the first dictation machine, a slightly improved version of his phonograph.
- In 1936, A team of engineers at Bell Labs, led by Homer Dudley, begins work on the Voder, the first electronic speech synthesizer. It was granted a patent in 1939.
- Later, IBM brings the Shoebox (1962), Automatic Call Identification (1971), Tangora (1980), MedSpeak (1996).
- Microsoft integrates speech in Office (2002), Windows Vista (2007) and launches Cortana (2014).
- Google introduces GOOG-411 in 2007.
- Apple announces Siri in 2011.
- Amazon launches Alexa in 2014.

❖ Methods and Procedures:

- Custom Speech-recognition systems:
    - The acoustic modeling and language modeling are important parts of statistically-based speech recognition algorithms. Many systems use the Hidden Markov models (HMMs) and language modeling for natural language processing.

- Cloud-based Speech-recognition systems:
    - With the advent of cloud, speech-recognition systems have improved to provide more accurate speech-to-text and text-to-speech services. These services can be used directly in your systems to incorporate speech-recognition service.

Background

Speech-recognition was first invented by Thomas Edison in 1877. As years passed by, the system kept on improving. By the help of cloud, many service providers such as Amazon, Google, Apple, Microsoft, IBM and Facebook started providing speech-recognition services that can be incorporated within any application or system. These cloud-based services keep on evolving with time and help in providing more accurate results. Majority of the systems use speech-recognition services that are hosted in cloud.

Brief Project Description

To develop an Alexa skill that will offer customers a more intuitive way to interact with the CSUN University Calendar using voice commands. This will help the customers to know about upcoming events at CSUN by asking Alexa about events based on the time-frame provided by the user. Also, user can ask detailed information of the event further.

Reference Architecture

The reference architecture includes 3 major components:

1. Device to process input request
2. Speech Recognition system
3. Endpoint

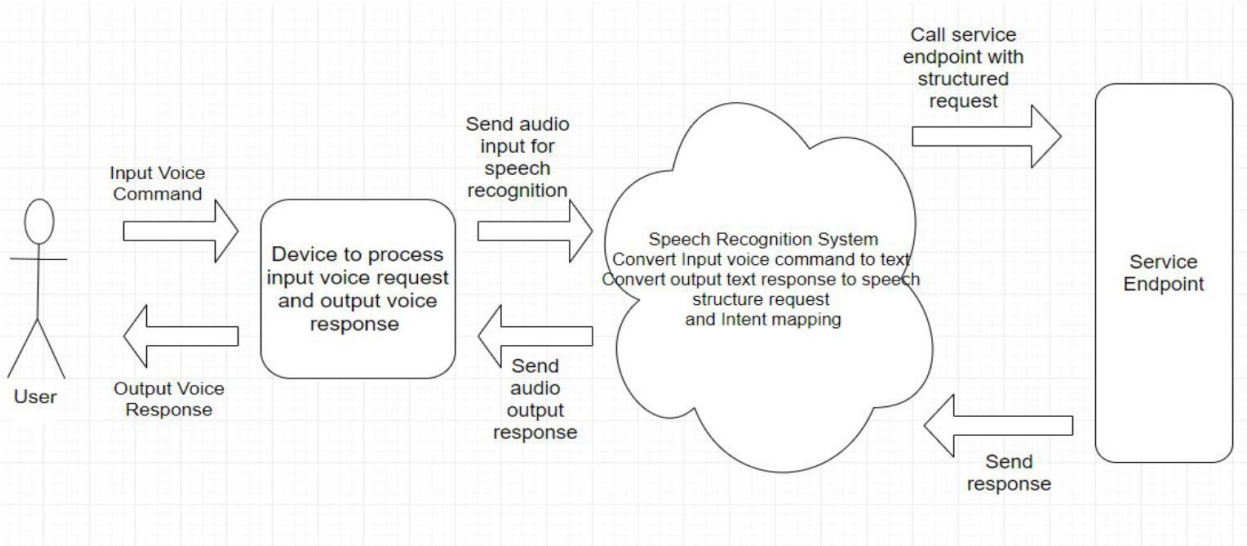

Figure 1: Reference Architecture

1. Device to process input request:

   This is a dumb-device that receives the audio as an input from user or the speech recognition system and sends or plays it back to speech-recognition system or the user.

2. Speech Recognition system:

   This is the heart of the voice activated system. The audio received from user is converted into text and send for processing to the endpoint. The text-based response is taken from endpoint and converted into audio and sent to the dumb-device.

3. Endpoint:

   The endpoint is where the business logic is written to process the request from the user.

Flow of the system:

i. The user invokes the system with the wake-word along with the invocation name for the system. (e.g. *Alexa, open Uber!*)
ii. The user then speaks or ask the device the request (known as utterances), he or she wants.
iii. The dumb-device takes the audio input and sends it to the speech recognition system.
iv. The speech recognition system, converts the audio input into text and structure/format the request such that the endpoint can understand.

v. Based on the intent, the request is send to the endpoint.
vi. At the endpoint, the request is processed based on the business logic written and the response is sent back to the speech recognition system. Additional services such as database, third-party API can be attached to the endpoint for processing the request.
vii. The response received by speech recognition system is converted into an audio file through speech synthesized markup language (SSML). This audio response is sent back to the dumb device.
viii. Upon receiving the response in audio format, the dumb-device then plays the audio to the user, fulfilling the user's request.

Implementation Guides

Implementation Guide Details #1

Amazon:

❖ Amazon's AVS:

- Alexa Voice Service (AVS) is an Amazon cloud service which helps in building Alexa skills and Alexa-enabled products easily and rapidly.
- Alexa is always getting smarter with new capabilities through machine learning.
- Complex technologies like Automatic Speech Recognition and Natural Language Understanding are handled in the cloud.
- AVS provides you easy-to-use APIs for core functionality like audio playback, volume control, or text-to-speech.
- AVS helps build your own custom skill with the Alexa Skills Kit (ASK).

❖ Amazon Alexa's Architecture:

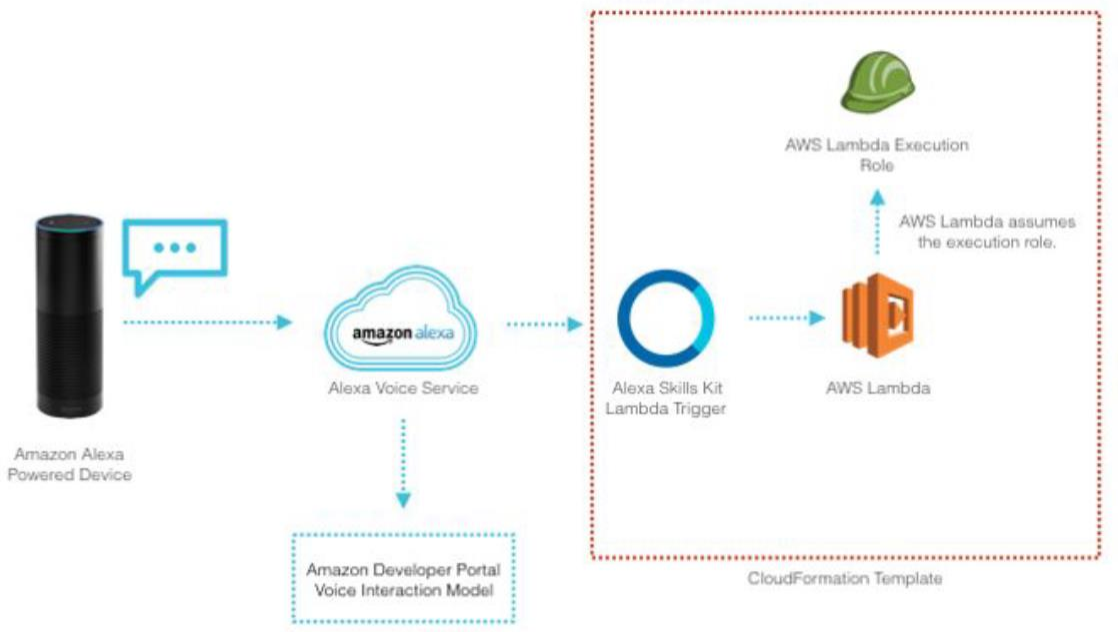

Figure 2: Amazon Alexa Architecture

- This architecture can be traced back to the reference architecture.
- The dumb-device in this architecture is the Amazon Alexa powered device.
- The speech-recognition system used here is the Amazon's Alexa Voice Service.
- The endpoint in this architecture is the AWS Lambda, which handles the business logic.

❖ Requirements:

- Alexa Skill Kit - Developer Account:
  - This is the portal provided by Amazon to build and test your skills.
- Amazon Web Service - Account (AWS Lambda)
  - An AWS account will be required to host your endpoint of the skill on AWS Lambda.
  - Custom endpoint can be used through HTTP.
- AWS CLI
  - If the endpoint used is AWS Lambda, then AWS CLI is helpful to deploy your code easily on AWS Lambda.

❖ Steps to build an Alexa Skill:

- Login into the Alexa Skill Kit (ASK).
- Go to the skills console and select 'Create new skill'.
- Provide the name for the skill. This is name that will be displayed in the Amazon Skill store.
- Choose the model for the skill:
  - Custom – To develop the skill from scratch
  - Flash Briefings – To utilize the built-in intents for a flash briefing skill which can be enabled by the user for his flash briefings.
  - Smart Home – To develop skills that can be integrated with smart home devices.
  - Video – To develop a skill which will display a video as an additional output with audio
- Skill Builder Checklist:
  - Invocation Name:
    - Invocation name is the word(s) that will be used by the user to invoke and begin interaction with the skill.
    - Requirements of Invocation name:
      - 2 or more words.
      - only lower-case alphabetic characters, spaces between words, possessive apostrophes, or periods used in abbreviations.
      - characters like numbers must be spelled out. For example, "twenty-one".
      - cannot be Alexa launch phrases: "launch", "ask", "tell", "load", "begin", and "enable".
      - Wake words including "Alexa", "Amazon", "Echo", "Computer", or the words "skill" or "app" are not allowed.
  - Intents, Samples and Slots:
    - An intent represents an action that fulfills a user's spoken request.
    - Choose from Amazon's 144 built-in intents or create your own.
    - Provide sample utterances to the intent
    - Use slots for re-usability
  - Build Model:

        - Now build the interaction model.
        - Check for any inconsistency or errors.
        - Resolve and build again.
    - Endpoint:
        - The Endpoint will receive POST requests when a user interacts with your Alexa Skill.
        - The request body contains parameters that your service can use to perform logic and generate a JSON-formatted response.
        - Service Endpoint Type:
            - AWS Lambda ARN
            - HTTPS

❖ **Testing:**

- Create sample JSON requests and test the cases in ASK.
- Use Alexa Simulator to test the skill using voice-commands or by typing the requests.
- Create JSON request to test the AWS Lambda endpoint in AWS console.
- Use Voice & Tone feature to test Alexa's response.
- Directly test the skill by enabling it on Alexa product. [The skill in development will require the Alexa product logged-in with the developer account credentials]

❖ **Publishing:**

- Skill preview in store:
    - Required: Public name, description, example phrases, small and large icons, category
    - Optional: keywords, privacy policy, terms of use
- Privacy and Compliance:
    - Requirement of purchases, user information, age limit, advertisement
- Availability:
    - Public or Business
    - Global or specific regions
- Submit for review.

Implementation Guide Details #2

Google:

❖ **Google Assistant:**

- The Google Assistant is a virtual personal assistant developed by Google that is primarily available on mobile and smart home devices.

- Google Assistant uses Dialogflow for natural language understanding and machine learning.
- Dialogflow' s Actions on Google integration allows you to reach users on voice-activated speakers like Google Home, eligible Android phones and other services.

❖ Google Assistant Architecture:

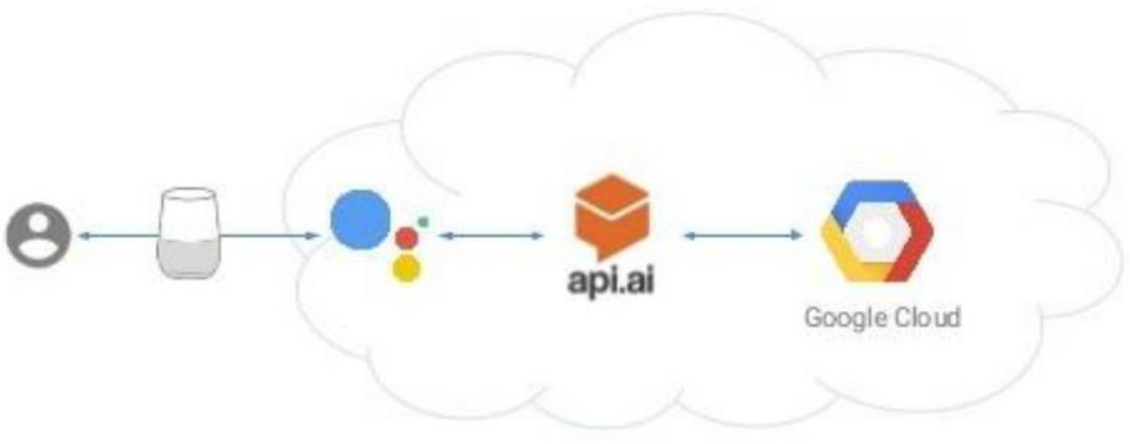

Figure 3: Google Assistant Architecture

- This architecture can be traced back to the reference architecture.
- The dumb-device in this architecture is the Google Assistant powered device.
- The speech-recognition system used here is the Google's Dialogflow (api.ai).
- The endpoint in this architecture is the Google cloud, which handles the business logic.

❖ Requirements:

- Google Assistant Actions - developer account:
  - A portal provided by Google to build and test your apps.
- Google dialog flow - developer account:
  - A portal provided by Google to host your natural language processing intelligence.

❖ Steps to build a Google App:

- Login into the Google Actions console.
- Select a new project.
- Provide the name for the app. This is name that will be displayed in the Google store.
- Choose the model for the skill:
  - Custom – To develop the app from scratch using dialog flow
  - Smart Home – To develop apps that can be integrated with smart home devices.

    - Templates:
        - Trivia – An app where user can create their own trivia game for other users.
        - Personality Quiz – Build a personality quiz which can suggest user which city they should live or what celebrity they resemble.
        - Flash Cards – Build an app that can help users to learn through flash cards.
- Conversation Design:
    - Select a persona:
        - Personas can range from happy, sad, self-deprecating, formal, and anything in between.
    - Write dialogs:
        - Construct the statements that will be used by user as a request.
    - Define intents with entities:
        - Create the intents for the dialogs and add entities as required.
    - Define actions:
        - Map the intents to actions/fulfilment (endpoint/webhook).
    - Build the model

❖ Testing:

- Use real devices:

  This lets you experience your apps the way users will and gives you a better idea of how well your app's user experience is designed.

- Use Actions simulator:

  The simulator lets you switch virtual surfaces easily (such as a phone or voice-activated speaker) to see how the experience works on different devices and lets you specify user input with text and voice.

❖ Publishing:

- Skill info:
    - Required: Public name, pronunciation, description, example phrases, small and large icons, category
    - Optional: keywords, privacy policy, terms of use
- Privacy and Compliance:
    - Requirement of purchases, user information, age limit, advertisement
- Availability:
    - Public or Business
    - Global or specific regions
- Submit for review

Implementation Guide Details #3

Microsoft:

❖ Microsoft's Cortana:

- Cortana and Bing use Language Understanding Intelligent Service (LUIS) as service.
- LUIS is a machine learning-based service to build natural language into apps, bots, and IoT devices.
- LUIS helps building model by integrating with Azure Bot Service to implement a voice activated system.
- Once the model starts processing input, LUIS begins active learning, allowing you to constantly update and improve the model.

❖ Cortana's Architecture:

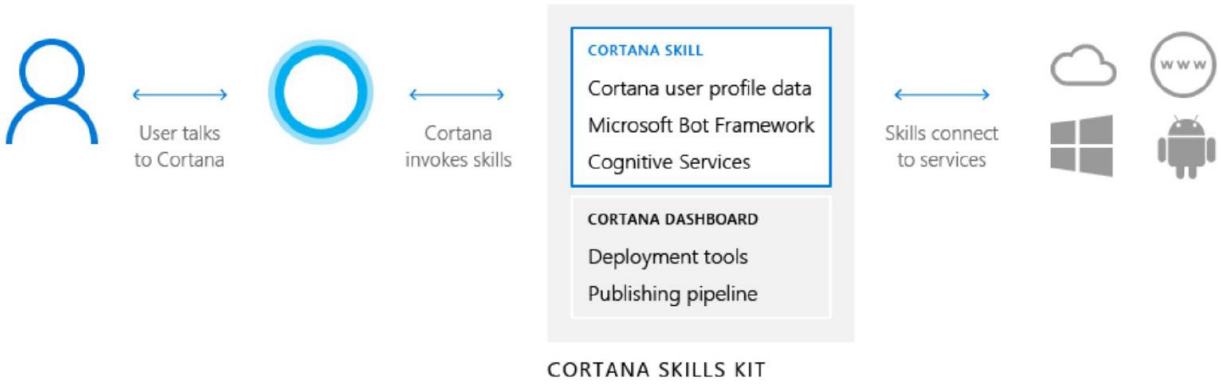

Figure 4: Cortana's Architecture

- This architecture can be traced back to the reference architecture.
- The dumb-device in this architecture is any device powered by Microsoft's Cortana.
- The speech-recognition system used here is the Language Understanding Intelligent Service (LUIS).
- The endpoint in this architecture is the Microsoft's Azure cloud, which handles the business logic.

❖ Requirements:

- Microsoft Cortana - developer account:
  - A portal provided by Microsoft to build and test your apps using Cortana.
- Microsoft Azure/LUIS - developer account:
  - A portal provided by Microsoft to host your natural language processing intelligence in cloud.

❖ Steps to build a Cortana Skill:

- Login into the Microsoft Cortana developer console.
- Select a new project.
- Provide the name for the skill.
- Choose the programming language (C# OR NodeJS)
- Choose the model for the skill:
    - Custom – To develop the skill from scratch and write custom code.
    - Smart Home – To develop apps that can be integrated with smart home devices.
- Conversation Design:
    - Write dialogs:
        - Construct the statements that will be used by user as a request.
    - Define intents with entities:
        - Create the intents for the dialogs and add entities as required.
    - Define endpoint:
        - Map the intents to the endpoint.
    - Build the model

❖ Testing:

- Use real devices:

  This lets you experience your apps the way users will and gives you a better idea of how well your app's user experience is designed.

- Use Skill Test simulator:

  The simulator lets you test the skill Interactive Testing and Batch Testing. Interactive Testing helps in testing the skill by using the utterances one after another. Batch Testing lets you test multiple utterances at a time.

❖ Publishing:

- Skill info:
    - Required: Public name, pronunciation, description, example phrases, small and large icons, category
    - Optional: keywords, privacy policy, terms of use
- Privacy and Compliance:
    - Requirement of purchases, user information, age limit, advertisement
- Availability:
    - Public or Business
    - Global or specific regions
- Submit for review

Conclusion

Summary

Speech recognition has a long history with several waves of major innovations. Most recently, the field has benefited from advances in deep learning and big data. The advances are evidenced not only by the surge of academic papers published in the field, but more importantly by the worldwide industry adoption of a variety of deep learning methods in designing and deploying speech recognition systems. These speech industry players include Amazon, Google, Microsoft, IBM, Apple, Baidu, Nuance, SoundHound, iFLYTEK many of which have publicized the core technology in their speech recognition systems as being based on deep learning in cloud.

Problems Encountered and Solved

- NPM Alexa-skill:
  - Problem: Unable to run the project due to missing package (Alexa-skill)
  - Solution:
    i. Required to download and install the Alexa-skill package for the project.
    ii. Command used: *npm install alexa-skill –save*
    iii. Zip the complete project along with the node_modules (consisting Alexa-skill package)
    iv. Upload the zip to AWS Lambda for running successfully.

- Session termination:
  - Problem: Unable to invoke the skill due to previous open session.
  - Solution: After completion of the request, close the session or provide a prompt message so that new request can be handled in the same session by the user.

Suggestions for Better Approaches to Problem/Project

- Use AWS Lambda as the endpoint while building an Alexa skill:
  - Helps to integrate with the skill easily
  - No additional development of JSON-based request/responses
  - No SSL certification required

- Bespoken service:
  - Helps to test the skill end-to-end
  - Use sample inputs/outputs to test the skill

Suggestions for Future Extensions

- Account Linking:
  - Allow users to link their CSUN profiles with the skill and fetch user specific events and notifications from the CSUN profile for each user.

- Set Reminders:
  - Allow users to fetch events and set reminders for their event of interest.

- Refine Utterances:
  - Add and modify the utterances accepted by the skill to incorporate users with varied level of proficiency in English language.

Appendices/References

[1] https://en.wikipedia.org/wiki/Voice_command_device

[2] https://en.wikipedia.org/wiki/Speech_recognition

[3] http://www.newsweek.com/2017/09/22/alexa-google-home-smart-phones-illiteracy-technology-voice-recognition-662282.html

[4] https://developer.amazon.com/docs/alexa-voice-service/get-started-with-alexa-voice-service.html

[5] https://developer.amazon.com/docs/ask-overviews/build-skills-with-the-alexa-skills-kit.html

[6] https://www.youtube.com/watch?v=qEYbjCXOU7Q

[7] https://developers.google.com/actions/extending-the-assistant

[8] https://developers.google.com/actions/reference/nodejsv2/overview

[9] https://docs.microsoft.com/en-us/azure/cognitive-services/luis/home

[10] https://docs.microsoft.com/en-us/cortana/skills/