Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 9, Issue. 6, June 2020, pg.165 – 173

SPEECH RECOGNITION TECHNOLOGY: AN OVERVIEW AND PRACTICAL IMPLEMENTATION APPROACH

[1]Azubuike N. Aniedu; [2]Jennifer C. Olibie; [3]Sandra C. Nwokoye
[1,2,3] Electronic and Computer Engineering, Nnamdi Azikiwe University Awka, Anambra State, Nigeria
[1] an.aniedu@unizik.edu.ng; [2] jenniferolibie@gmail.com; [3] sandrachiomanwokoye@gmail.com

Abstract— There are constant and on-going research and innovations to enhance speech processes for ease of communication for all, but most especially those with disabilities. With recent technology advancements, voice recognition has gained a massive acceptance particularly, when it comes to making communications with machines more natural and seamless. Voice technologies is increasingly being applied in many electronic devices most especially phones, personal computers, and web browsers. This is achieved mainly using a Speech Recognition System (SRS). In this write-up, some of the models, approaches, Application Programming Interfaces (APIs), performance metrics used in SRS are reviewed. Thereafter, a sample implementation of one of the common APIs, JavaScript Web Speech API, was presented; noting particularly the simplicity with which speech recognition using APIs we can achieved, with just a basic knowledge of programming language.

Keywords— Voice recognition, speech recognition systems, APIs, JavaScript, Speech-to-text, text-to-speech

I. INTRODUCTION

In recent years, there has been a heightened need for human-machine interaction. These interactions are becoming increasingly mainstream with our daily lives as gadgets and devices become smarter and more intuitive. This trend is spurred on by many underlying factors chief among them being the need for humans to give commands to machines in natural communication means.

One of the technologies that facilitate this interaction is the voice technology mainly achieved through the speech recognition system (SRS). Speech recognition basically put, enables the recognition and translation of spoken language into text by computers. It is the process by which a computer takes a speech signal (recorded using a voice sensor e.g. microphone) and converts it into words in real-time.

Every physical system or agent needs an interface with which to interact with its environment. Systems which accept (sense) voice/sound data, process it and produce a response based on the data are known as Speech Recognition System (SRS) and the devices used to interface the system with the voice input is called Voice-User Interface (VUI). Some examples of VUIs are Microsoft's Cortana and Google's Voice Search.

This paper presents models, approached, and performance metrics used in speech recognition systems. Also presented here is a simple Web Speech Application Programming Interface (API) to demonstrate the ease of achieving an SRS integration in a system.

A. Processes of Speech Recognition

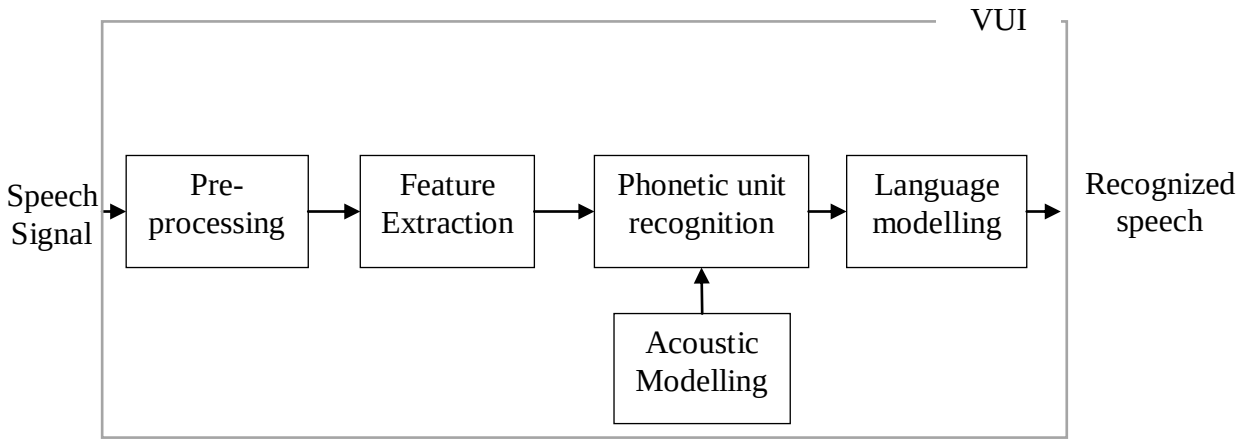

Fig I General block diagram of a speech recognition system (Sarma & Sarma, 2015)

A speech recognition system upon reception of the sound signal into the VUI, typically through an embedded microphone goes through the following processes before outputting the result as seen in fig I:

B. Speech Detection & Processing:

The first thing an SRS system does is to *pre-process* the received signal so as to recognise the presence of a speech signal. An analog-to-digital converter (ADC) translates this analog signal to digital signal (Fig II).

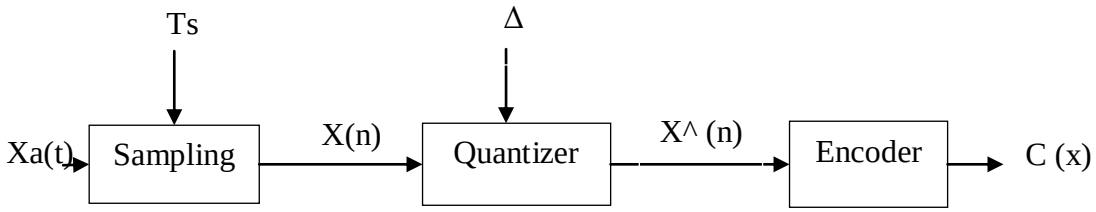

Fig II Block diagram of an analog to digital converter (Unpublished source)

Where:
Xa(t) = speech continuous signal
Ts = sampling period
X(n) = discrete time sequence
Δ = quantization interval
X^ (n) = digital signal
C (x) = sequence of binary code words

It does this by sampling – dividing the continuous signal into discrete unit at specified frequencies, quantizing – dividing the amplitude of the signals into discrete units based on a specified quantization interval to obtain the digital signal and then encoding the signal.

C. Speech Recognition

This is the most important and difficult part of the process of the SRS, the signal undergoes *feature extraction*, obtaining relevant characteristics of the signal and consequently, divided into small segments and matched to known phonemes according to the language setting input: a phoneme is the smallest element of a language, it is a representation of the sounds we make and put together to form meaningful expressions (Grabianowski, Speech Recognition: Speech to Data, 2006). There are a varying number (approximately 44) of phonemes in English, representing all the vowels and consonants used for speech. The process of matching the digital signal to known phonemes is known as *acoustic modelling* and decoding into a sequence of words, i.e. *language modelling* is done using an algorithm according to Hidden Markov Model, Neural Network or Dynamic Time Wrapping. These models basically have a large database of known words of which the phonemes are decoded from and subsequently matched.

The models use powerful and complicated statistical modelling systems that leverage on probability and mathematical functions to determine the most likely outcome of the speech (Grabianowki, 2006).

Some of these models are covered in section II.A.1.

D. Semantic Recognition

This can also be called *language modelling*. The meaning of the recognized words is obtained by higher-level processing using dynamic knowledge representation to modify the syntax, semantics and pragmatics (Hope, 2019). In simpler words, the system checks the language structure of its interpretation using previous knowledge and errors are corrected.

E. Output

The system then responds to the user in the form of the requested action being performed, voice output, text or its equivalent.

II. CLASSIFICATION OF SRS

Hope (Hope, 2019) classified SRS as follows:

1. Speaker dependent system: in which the system has been trained to recognise a single speaker. This is easier to develop, more accurate and cheaper but less flexible.
2. Speaker independent system: This system recognises multiple speakers. It does not require training and is much harder to develop but more flexible.
3. Discrete speech recognition: Here, the system requires that the user pauses between each word, as this enables efficiency in the system's speech recognition.
4. Continuous speech recognition: This is more convenient as the system recognises continuous speech. The user is not required to specifically stop, rather, the user speaks at one's normal speech rate.
5. Natural language: This SRS system feels like a full round communication with a fellow human as the computer recognises the voice and responds equally in response to the question being asked.

A. Models and Approaches in SRS systems

Several authors have developed different models and approaches to building a software recognition system. Some of these models and approaches include:

1) Models

1. Hidden Markov Model – Here, each model is like a link in a chain where the chain is a word. While trying to match the phoneme to the words, different words come up and each phoneme is assigned a probability based on the built-in dictionary and user training.

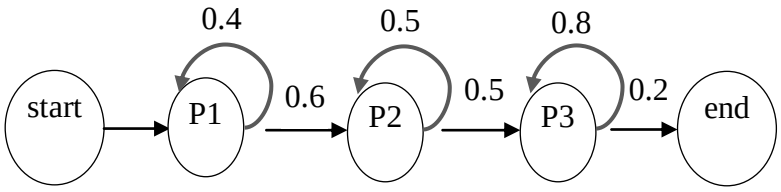

Fig III Sample Markov model for Word Processing

From (Fig III) above, we can deduce sequences and their probability of occurrence as:
P1P2P3 = 0.6 x 0.5 x 0.2 = 0.06
P1P1P2P2P3 = 0.6 x 0.4 x 0.5 x 0.5 x 0.2 = 0.012
We see that the probability of P1P2P3 being the outcome is more and thus, it is used as the output.
Using a more practical example for the pronunciation of the name 'chidera':

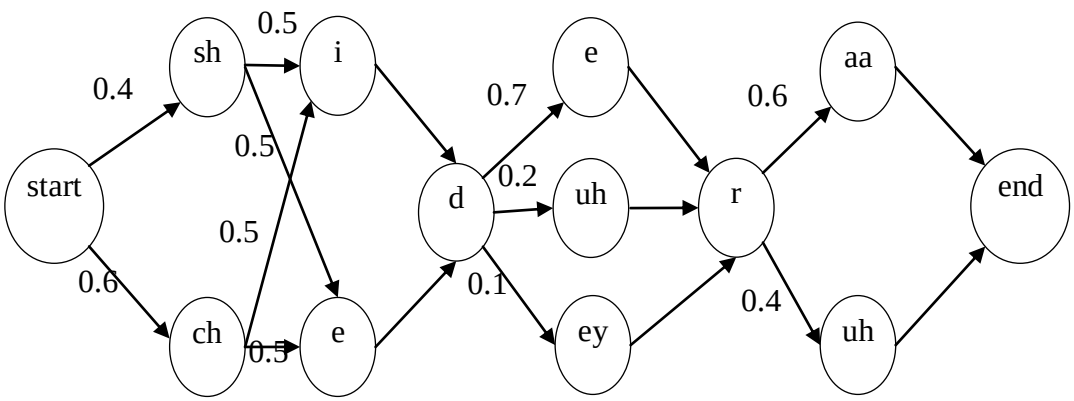

Fig IV Sample Markov Model for the word 'Chidera'

The above trained probability statistics (Fig IV) accommodates for the following pronunciations:
Ch I de r aa: 0.6 x 0.5 x 0.7 x 0.6 = 0.126
Sh e duh r aa: 0.4 x 0.5 x 0.2 x 0.6 = 0.024
Ch e d ey r uh: 0.6 x 0.5 x 0.1 x 0.4 = 0.012

We notice that the most probable word pronounced would be 'chideraa' followed by 'shedeuhraa' and lastly 'chedeyruh'.

The above metrics is also adaptable to sentences. For example, in (Fig V):

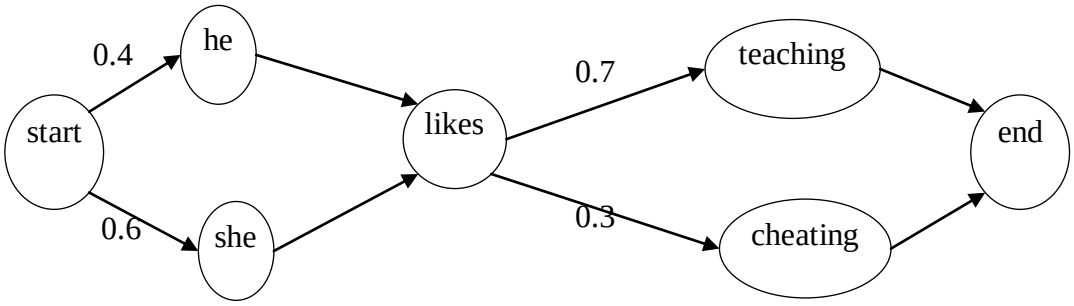

Fig V Sample Markov Model for a sentence

Sentences such as:
He likes teaching: 0.4 x 0.7 = 0.28 - relatively probable
He likes cheating: 0.4 x 0.3 = 0.12 – very improbable!
She likes teaching: 0.6 x 0.7 = 0.42 – very probable!
She likes cheating: 0. 6 x 0.3 = 0.18 – relatively improbable

Due to this probability metrics the Markov model provides, the SRS is more intelligent and able to differentiate between similar sounding words/sentences such as 'I want it' and 'I won't eat'.

2. Dynamic Time Warping: This approach has been widely replaced by the HMM model. According to Vintsyuk (Vintsyuk, 1968), it is an algorithm for measuring similarities between two sequences that may vary in time or speed. It allows the computer find optimal match between two given sequences with certain restrictions. For example, a received voice signal is matched in the database for signals with similar wave shapes.

Other models used in SRS include: Neural networks, end-to-end automatic speech recognition, Gaussian mixture model, etc.

B. Approaches to SRS

There have been several approaches used in the development of an SRS. It can currently be broadly classified into 3:

i) Acoustic-Phonetic Approach

This was the initial approach to SRS development. In this approach, the machine attempts to decode the speech signal in a sequential manner based on the observed acoustic (sound) feature and the known relationship between these features and the phonetic symbols (Rabiner & Juang, 1993). The phases involved in this approach include:

- Segmentation and Labelling phase: This is the first phase and it involves segmenting the signal into discrete regions each representing a phoneme and attaching one or more labels to each region accordingly.
- Word/Sentence Extraction: Here, the machine attempts to determine a valid word/sentence from the sequence of phonetic labels.

Although this has been the prevalent approach, it has not however been without its problems which include:

- Lexical Access Problem – this is the term used to describe the difficulty in decoding phonetic units into word strings.
- Difficulty in getting reliable lattices (group of labels).

ii) Pattern recognition approach

In this approach, speech patterns are used directly without clear feature determination and segmentation. It involves two steps:

- Pattern training – Here, the system is trained on the possible patterns to be found in speech.
- Pattern comparison – in which the system does a direct comparison of the unknown speech with each possible pattern learned in the training phase.

iii) Artificial intelligence

This approach tries to incorporate both the acoustic-phonetic approach and the pattern recognition approach. It integrates expert systems into the segmentation and labelling phase of the acoustic and phonetic approach. It also incorporates learning and adaptation, all which is done with the aid of neural networks.

III. PERFORMANCE IN SPEECH RECOGNITION

In order to determine the performance of an SRS system, speed and accuracy are considered. Whereas the Word-Error-Rate (WER) is used to measure accuracy, speed is measured using real-time factor.

1) Word-Error-Rate

WER is a measure of the error tendency of the system and it is measured with respect to a reference transcript. It is produced by applying a dynamic alignment between the output of the ASR system and a reference transcript (italk2learn, 2015). Here, three types of errors can be distinguished:

- Substitution error (sub) – here, a word is being substituted for another. For example, murder might be substituted for mother.
- Deletion error (del) – this involves omission of word(s) from the sentence
- Insertion error (ins) – This occurs when a word has been wrongly inserted into the sentence.

The WER is calculated thus:

$$WER = \frac{100*(sub+ins+del)}{total\ number\ of\ words\ in\ reference\ script} \quad (1)$$

For example: Given Table I,

TABLE I: ALIGNMENT AND TYPES OF ERRORS FOR TESTING

Ref	My	name	is	Chidera	Olibie.	I'm	awesome	and	New
ASR	My	name	is	Children	Olivia.	-	awesome	-	anew
Error Type				sub	sub	del		del	ins

$$WER = \frac{100 * (2 + 2 + 1)}{9}$$

$$= \frac{100 * 5}{9}$$

$$= 55.56\%$$

We see that the word-error-rate of this system is high and needs a lot more training to reduce it. This above equation can also be written in weighted form depending on the needs of the company and the priority they place on each error. For example: a substitution error might have a weight of 0.5, an insertion error a weight of 1.5 and a deletion error a weight of 1, thus the equation is rewritten as

$$WER = \frac{100*(0.5sub+1.5ins+del)}{total\ words} \quad (2)$$

Equation (2) is not standard and varies depending on requirements placed.

2) Real Time Factor (RTF)

This is a common metric for measuring speed of an Automatic Speech Recognition (ASR) and it is given as:

$$RTF = \frac{P}{I} \quad (3)$$

Where P = time taken to process input

I = duration of input.

For example: If it takes 50 secs to process an input of 30 sec duration, then its RTF is given as:

$$RTF = \frac{50}{30} = 1.667$$

From the equation (3), we notice that an RTF of 1 indicates a processor working in real-time and this indicates a high speed.

A. APIs used in Speech Recognition

Application Programming Interface (API) is an interface developed to abstract the difficulties of having to develop systems from scratch. It gives you the functionality of those systems in a much simpler, abstracted, and faster way. In this context, these APIs abstract the speech recognition pre-processes and processes while giving you quality and easy-to-use interface. These interfaces are not for the consumer end but for the development of the applications that would then incorporate these systems.

These APIs can support several languages and have general functions that include text-to-speech conversion, speech-to-text conversion, transcription, etc. Some examples of these APIs are Google Cloud Speech-to-Text, Twilio's ASR, Microsoft Speech API, Watson Speech to Text, Watson Text to Speech, Amazon Transcribe, Microsoft Azure Bing Speech API, Amazon Polly, VoxSigma API, Speechmatics ASR, Nexmo Voice API, Dialogflow, CMU Sphinx, Kaldi, Vocapia, etc.

1) Implementing a sample Web-based voice recognition system using JavaScript Web Speech API

An advantage of API is the abstraction of the underlying process from the developer. This JavaScript Web Speech API is, as the name implies, a speech API made specifically for the web using the JavaScript language. A small sample program has been implemented to illustrate the simplicity with which one can achieve speech recognition using this API. However, this API works only for Chrome and Mozilla Firefox as of 2019.
For the program, Visual Studio Code (text editor), Chrome Web Browser applications were used.

To illustrate, 3 files were created which include speech.html, main.css and main.js.

The speech.html file contains the mark-up for the application (Fig IV).

speech.html × | main.js | main.css

speech.html ▸ html ▸ body ▸ div#wordsDiv ▸ img

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8" />
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <title>Speech Recognition</title>
7    <meta name="viewport" content="width=device-width, initial-scale=1">
8    <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
9
10 </head>
11 <body>
12     <p>Jenny's Mini Speech-Text</p>
13     <div id="wordsDiv">
14       <img src="./microphone.svg" alt="mic" title="Click on Mic to record">
15     </div>
16
17     <script src="main.js"></script>
18 </body>
19 </html>
```

Fig III Speech.html

To explain the code in Fig VI, a typical html code contains a header and a body. The header content is not usually visible to the user except for the title on line 6, which shows on the browser tab. On line 8, is the link tag to link the CSS file to the html. Then follows the body of the page, which is the visible part of the code. This contains the <p> tag which has been used as the application header, tag that contains a <p> tag which would be created to contain the transcribed words and an <img> tag on line 14 which contains the microphone image to be clicked on to start recording. Next is the script tag on line 17, to reference the JavaScript file to be called upon.

The CSS file to style the above mark-up is shown in Fig VII.

speech.html | main.js | main.css

main.css ▸ #wordsDiv

```css
body{
  color: #222;
  text-align: center;
  font-size: 30px;
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
  margin-top: 50px;
}
#wordsDiv{
  width: 400px;
  height: 50px;
  margin: auto;
  border-radius: 10px;
  box-shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);
  color: #222;
  text-align: left;
  font-size: 16px;
  padding: 2px 10px;
}
img {
  height: 50px;
  width: 25px;
  margin: 0 5px;
  float: right;
}
```

Fig VII main.css

In Fig VII, the whole of the application was styled highlighting the various designs added to the application. This will reflect as seen in Fig IX. Some of the designs added include: reduction in the microphone size, setting of the font and its size, setting of the output box and centralising it with the 'margin: auto' rule, setting the width and height of the box and so on.

Then the application is powered using the Speech Recognition API in the main.js file (Fig VIII).

speech.html | main.js | main.css

main.js ▸ ...

```js
window.SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;

const recognition = new SpeechRecognition(); // instantiating the speech recognition interface

recognition.interimResults = true; // show results that are not yet final

let p = document.createElement('p'); //paragraph to hold the words

const words = document.querySelector('#wordsDiv');
const img = document.querySelector('img');

words.appendChild(p);

recognition.addEventListener('result', e => {
  console.log(e);
});

const start = () => {
  recognition.start(); //start the SR service on the instance
  recognition.onresult = (event) => {
    const speechToText = event.results[0][0].transcript;
    p.textContent = speechToText; // add the transcribed text to the DOM
  }
}

img.addEventListener('click', () => start()); //add click event to the Mic button
```

Fig VIII main.js

From figure VIII, Line 1 enables one to set the speech recognition interface irrespective of the browser one is on.

Line 3 creates an instance of the speech recognition service.

Line 5 sets interim results to true i.e. it should show the results of the speech recognition in process even if they are not final.

Line 7 creates the html element that will hold the words.

In line 9 and 10, the tags needed for the Html DOM was queried; first the 'div' to add the words into and then the image that would be monitored for a click event.

Line 12 adds this element as a child of the tag.

In line 18, a function that starts the recognition service instance and adds the result of the transcription to the <p> tag was created.

Line 26 sets up the function 'start' to be called on the click event of the microphone image.

Finally, on running the application, when one clicks the microphone icon and then speak some words, the output would be as shown in Fig IX:

Fig IX Web output

Notices that with the API, one does not have to 'worry' about the underlying principles of software recognition as these have been abstracted. It is however relevant to understand these underlying principles in case one wants to develop his/her own API or run into errors that will require editing the API codes itself.

B. Setbacks in speech recognition

The following factors can be seen as affecting the accuracy of an SR system:

- The voice signal is often accompanied with noise which makes recognition a bit trickier. Sometimes, low quality sound cards also introduce noise to the signal.
- Different speakers have different accents which the system may not have been trained to recognise.
- The rate of speech varies from speaker to speaker, thus making individual words hard to recognise.
- Difficulty in separating simultaneous speech from multiple users.
- These statistical models in use take a lot of processor power. Also, the stored word dictionaries take up a lot of memory, although this is being eliminated with cloud computing.
- There is no possible way for SRS to differentiate homonyms (Grabianowski, Speech Recognition: Weaknesses and Flaws, 2006). However, these models have incorporated contextual decoding that greatly improves the results.
- There is need for a large vocabulary of words varying in syntax, semantics, and pragmatics. This lack of widespread vocabulary leads to some inconsistencies in SRS output.

C. Advantages, Disadvantages and Applications

Several Advantages of Speech Recognition Systems can be seen of which include increased productivity, closer and easier interaction with computers, assistance of people with disabilities, enabling hands-free technology, diminishing spelling mistakes etc.

The major disadvantages of SRS are that it takes time to setup/train and it might lead to vocal strain due to continuous use by the user.

The applications of Speech recognition can be seen in video games, car speech recognition, voice assistant, data entry, documentation, speaker identification, automation at call centers, medical industry, military, etc.

So many companies have seen the advantages of having voice control enabled devices and thus have delved into it. Some of the most popular ones are Google Assistant, Amazon's Alexa, Apple's Siri, and Microsoft's Cortana. A brief comparative summary of these are given in Table II.

TABLE II: COMPARATIVE SUMMARY OF VIRTUAL ASSISTANTS (AZURE, 2019) (GOOGLE, 2019) (TUNG, 2019)

	Assistant	Alexa	Siri	Cortana
Company	Google	Amazon	Apple	Microsoft
Year released	2016	2014	2011	2014
Language	C++	Multi	Java, JavaScript, Objective C	C# (C/C++)
Privacy	No	No	Possibly yes	No
Accuracy	90.6%	87%	62.2%	81.9%
SRS	AI (RNN)	Long short-term memory ANN	Sophisticated machine learning techniques (e.g. convolutional neural networks and long short-term memory)	AI

IV. CONCLUSION

Speech recognition has evolved massively especially in the 21st century with milestones achieved within a short period of time. It aims to revolutionize the interaction between humans and their gadgets and to provide a more fluid communication in a natural language.

Most technology companies are fast incorporating this system into their system and also creating enabling fora to encourage others to incorporate it too. However, there is a strong advocacy for care to avoid the violation of privacy while training their models.

REFERENCES

[1] K. Sarma and M. Sarma, "Acoustic Modelling of Speech Signal using Artificial Neural Network: A Review of Techniques and Current Trends - Scientific Figure," 01 July 2015. [Online]. Available: https://www.researchgate.net/figure/Basic-block-diagram-of-a-speech-recognition-system_fig1_281670062.

[2] E. Grabianowski, "Speech Recognition: Speech to Data," How Stuff Works, 10 November 2006. [Online]. Available: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition1.htm.

[3] E. Grabianowki, "How Speech recognition Works: Speech recognition and statistical modelling," How Stuff Works, 10 November 2006. [Online]. Available: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition2.htm.

[4] C. Hope, "Voice Recognition," Computer Hope, 05 January 2019. [Online]. Available: www.voxtab.com/trancription-blog.

[5] T. K. Vintsyuk, "Speech Discrimination by Dynamic Programming," Kibernetika, pp. 81-88, 1968.

[6] L. Rabiner and B.-H. Juang, Fundamentals of Speech Recognition, USA: PTR Prentice Hall Inc., 1993.

[7] italk2learn, "Evaluating the performance of automatic speech recognition systems," 7 May 2015. [Online]. Available: www.italk2learn.eu. [Accessed 15 April 2019].

[8] E. Grabianowski, "Speech Recognition: Weaknesses and Flaws," How Stuff Works, 10 November 2006. [Online]. Available: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition3.htm.

[9] M. Azure, "Speech Services," 2019. [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/speech-services. [Accessed 27 April 2019].

[10] Google, "Cloud Speech-to-Text," 2019. [Online]. Available: https://cloud.google.com/speech-to-text.

[11] L. Tung, "Move over Siri, Alexa: Google's offline voice recognition breakthrough cuts response lag," 13 March 2019. [Online]. Available: https://www.zdnet.com/google-amp/article/move-over-siri-alexa-googles-offline-voice-recognition-breakthrough-cuts-response-lag.

