© 2017 IJSRST | Volume 3 | Issue 3 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology

Speech Synthesizer for English Audio with Indian Accent

Shweta Pardeshi, Nilisha Mahale, Priya Pandhare, Bhagyashree Kamble

Computer, Savitribai Phule Pune University/K.K Wagh College of Engineering/Cognifront Pvt Ltd, Nashik, Maharastra, India

ABSTRACT

The main aim of the project is to generate synthetic Speech in Indian accent. The synthesizer will take input as a transcript and produce audio file as a output. Primarily we will learn theoretical aspects of speech synthesis and apply them to develop complete code to synthesis audio. We will use Text processing which is responsible for determining all knowledge about the text that is not specifically phonetic and Phonetic analysis which focuses on the phone level within each word, tagging each phone with information about what sound to produce and how to produce it. General issues such as NLP, DSP, different voices, accents and multiple languages, HMM model, phonemes are discussed. As we know Indian Accent is different from that of American or British Accent. Hence our focus is on Indian accent. This would largely help even for those people who cannot understand the American accent. The speech synthesizer has enormous applications such as reading for blind people, telecommunication services, language education, aid to handicapped person, talking books and toys, call centre automation.

Keywords: Speech synthesis, Text-to-speech, Transcript, Phonemes, NLP, DSP, Hmm, Audio sampling, Audio Pitch, Audio file formats**.**

I. INTRODUCTION

Our brain is wired to hear Indian accent. The accent connects a person to human psychology. Hence we are developing a system in which the goal of synthesizer is to convert runtime arbitrary input text into natural sounding speech as **Indian accent**.

The objectives of our project are

1) To learn theoretical concepts of Speech synthesis, Text-to-speech, Transcript, Phonemes, NLP, DSP, Audio sampling, Audio Pitch, Audio file formats.
2) To apply all the concepts to create fully functional systems
3) To provide tools for Graphics Designer, Language experts, Students and end users which are discussed further.

Text-to-speech synthesis is a research field that has received a lot of attention and resources during the last couple of decades for excellent reasons. One of the most interesting ideas is the fact that a workable TTS system, combined with a workable speech recognition device, would actually be an extremely efficient method for speech coding. This field of study is known both as Speech Synthesis that is the "synthetic" (computer) generation of speech, and Text-To-Speech or TTS. It is the process of converting written text into speech. In the process of speech synthesis, mainly two processing components are used; they are NLP (natural language processing) and DSP (digital signal processing) modules.

We will first see the ideal text to speech working in Natural language processing

A Text-To-Speech (TTS) synthesizer is a computer-based system that should be able to read *any* text aloud, whether it was directly introduced in the computer by an operator or scanned and submitted to an Optical Character Recognition (OCR) system. Let us try to be clear. There is a fundamental difference between the system we are about to discuss here and any other talking machine (as a cassette-player for example) in the sense that we are interested in the automatic production of new_sentences. This definition still needs some refinements. Systems that simply concatenate isolated words or parts of sentences, denoted as *Voice Response Systems*, are only applicable when a limited vocabulary is required (typically a few one hundreds of words), and when the sentences to be pronounced respect a very restricted structure, as is the case for the announcement of arrivals in train stations for instance. In the context of

TTS synthesis, it is impossible (and luckily useless) to record and store all the words of the language. It is thus more suitable to define Text-To-Speech as the automatic production of speech, through a grapheme-to-phoneme transcription of the sentences to utter.

At first sight, this task does not look too hard to perform. After all, is not the human being potentially able to correctly pronounce an unknown sentence, even from his childhood ? We all have, mainly unconsciously, a deep knowledge of the reading rules of our mother tongue. They were transmitted to us, in a simplified form, at primary school, and we improved them year after year. However, it would be a bold claim indeed to say that it is only a short step before the computer is likely to equal the human being in that respect. Despite the present state of our knowledge and techniques and the progress recently accomplished in the fields of Signal Processing and Artificial Intelligence, we would have to express some reservations. As a matter of fact, the reading process draws from the furthest depths, of the human intelligence.

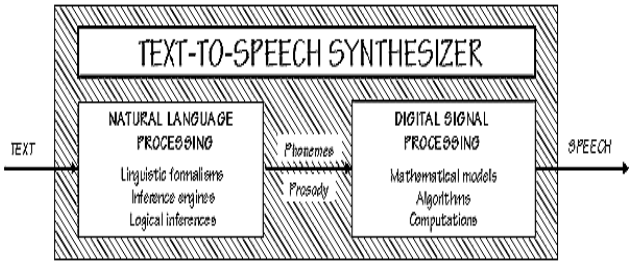

Figure 1: Basic Text to speech model

Figure 1 introduces the functional diagram of a very general TTS synthesizer. As for human reading, it comprises a Natural Language Processing module (NLP), capable of producing a phonetic transcription of the text read, together with the desired intonation and rhythm (often termed as *prosody*), and a Digital Signal Processing module (DSP), which transforms the symbolic information it receives into speech. But the formalisms and algorithms applied often manage, thanks to a judicious use of mathematical and linguistic knowledge of developers, to short-circuit certain processing steps. This is occasionally achieved at the expense of some restrictions on the text to pronounce, or results in some reduction of the "emotional dynamics" of the synthetic voice (at least in comparison with human performances), but it generally allows to solve the problem in real time with limited memory requirements.

II. METHODS AND MATERIAL

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

1. Text to words

Reading words sounds easy, but if you've ever listened to a young child reading a book that was just too hard for them, you'll know it's not as trivial as it seems. The main problem is that written text is ambiguous: the same written information can often mean more than one thing and usually you have to understand the meaning or make an educated guess to read it correctly. So the initial stage in speech synthesis, which is generally called **pre-processing** or **normalization**, is all about reducing ambiguity: it's about narrowing down the many different ways you could read a piece of text into the one that's the most appropriate.

Preprocessing involves going through the text and cleaning it up so the computer makes fewer mistakes when it actually reads the words aloud. Things like numbers, dates, times, abbreviations, acronyms, and special characters (currency symbols and so on) need to be turned into words and that's harder than it sounds. The number 843 might refer to a quantity of items ("eight hundred and forty three"), a year or a time ("eight forty three"), or a padlock combination ("eight four three"), each of which is read out slightly differently. While humans follow the sense of what's written and figure out the pronunciation that way, computers generally don't have the power to do that, so they have to use statistical probability techniques. So if the word "year" occurs in the same sentence as "843," it might be reasonable to guess this is a date and pronounce it "eight forty three." If there were a decimal point before the numbers ("0.843"), they would need to be read differently as "eight four three."

Preprocessing also has to tackle **homographs**, words pronounced in different ways according to what they mean. The word "read" can be pronounced either "red" or "reed," so a sentence such as "I read the book" is immediately problematic for a speech synthesizer. But if it can figure out that the preceding text is entirely in the

past tense, by recognizing past-tense verbs ("I got up... I took a shower... I had breakfast... I read a book..."), it can make a reasonable guess that "I read [red] a book" is probably correct. Likewise, if the preceding text is "I get up... I take a shower... I have breakfast..." the smart money should be on "I read [reed] a book."

2. Words to phonemes

Having figured out the words that need to be said, the speech synthesizer now has to generate the speech sounds that make up those words. In theory, this is a simple problem: all the computer needs is a huge alphabetical list of words and details of how to pronounce each one (much as you'd find in a typical dictionary, where the pronunciation is listed before or after the definition). For each word, we'd need a list of the **phonemes** that make up its sound.In theory, if a computer has a dictionary of words and phonemes, all it needs to do to read a word is look it up in the list and then read out the corresponding phonemes, right? In practice, it's harder than it sounds. As any good actor can demonstrate, a single sentence can be read out in many different ways according to the meaning of the text, the person speaking, and the emotions they want to convey (in linguistics, this idea is known as prosody and it's one of the hardest problems for speech synthesizers to address). Within a sentence, even a single word (like "read") can be read in multiple ways (as "red"/"reed") because it has multiple meanings. And even within a word, a given phoneme will sound different according to the phonemes that come before and after it.

An alternative approach involves breaking written words into their graphemes (written components units, typically made from the individual letters or syllables that make up a word) and then generating phonemes that correspond to them using a set of simple rules. This is a bit like a child attempting to read words he or she has never previously encountered (the reading method called phonics is similar). The advantage of doing that is that the computer can make a reasonable attempt at reading any word, whether or not it's a real word stored in the dictionary, a foreign word, or an unusual name or technical term. The disadvantage is that languages such as English have large numbers of irregular words that are pronounced in a very different way from how they're written (such as "colonel," which we say as kernel and not "coll-o-nell"; and "yacht," which is pronounced "yot" and not "yach-t") —exactly the sorts of words that cause problems for children learning to read and people with what's known as surface_dyslexia (also called orthographic or visual dyslexia).[1]

3. Phonemes to sound

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

Concatenate

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

The figure below shows the Existing structure which tells us the history of our project innovation idea.

Earlier we needed a narrator to convert transcript to audio signal. In our transcript we are removing the narrator and converting transcript directly into audio signals. The audio signals will then be converted into graphics by a graphic designer. This further will be

delivered to teacher or student. All the extracted phoneme will be stored in the master database.

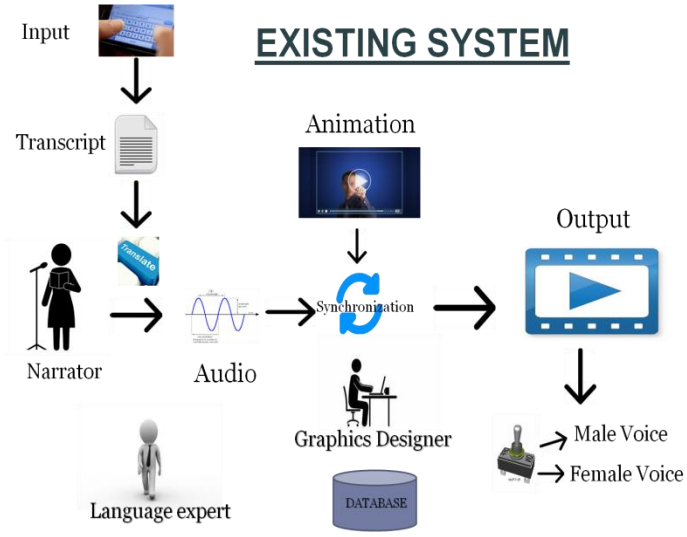

Figure 2: Existing system of speech mapping

Text paragraphs will be broken into sentences, sentences will be broken into words and words will broken into phonemes. The input text will be finally converted into English audio with Indian accent. There would also be options of male and female voices with different variations.

The figure below elaborates our System Architecture:

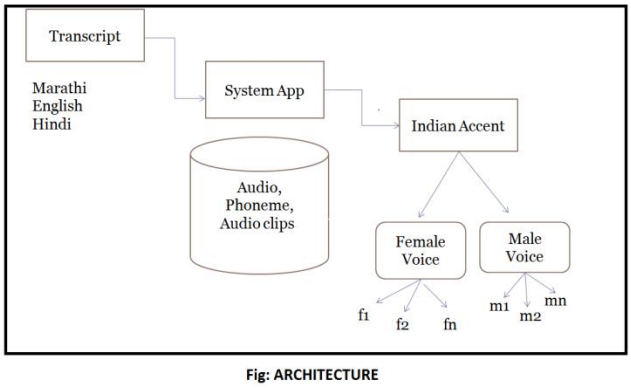

Fig: ARCHITECTURE

Figure 3: Architecture of speech synthesizer

Since in our system the transcript is the only input, we giving this input to the system application like audio, phonemes, audio clips, Hmm model, Stitching algorithm, etc. This would be converted into audio form in .wav file format in Indian accent. It would be changed as per user choice in different variations of male and female voices. Input firstly will be in English text .It would further be changed into Hindi or Marathi languages as per users choice

III. RESULTS AND DISCUSSION

We will see the success and failure conditions:

Success Conditions: Indian accent English audio is produced.

Failure Conditions:

1) Mixed language.
2) Database crash.
3) Sound card not working.

This is our Work Breakdown Structure (WBS)
(E001) Text Analysis
(E002) Organize the input sentences into manageable lists of words
(E003) Words are decomposed into their elementary units Phonemes.
(E004)Determination of the phonetic transcription of the incoming text
(E005)Analyzing phonemes
(E006)Detection of vowels, barakhadi, etc (depending on language)
(E007)Matching word with phonemes
(E008)Storing phonemes into database
(E009)Playing audio through API
(E010)Generation of waveforms
(E011)Testing for successful inputs and enormous inputs
(E012)Testing on multiple platforms and WBSS

IV. CONCLUSION

We proposed a system of speech synthesis to convert runtime arbitrary input text into natural sounding speech as Indian accent. We learnt various theoretical concepts of Speech synthesis, Text-to-speech, Transcript, Phonemes, NLP, DSP, Audio sampling, Audio Pitch, Audio file formats and how to apply all the concepts to create fully functional systems. We will basically provide tools for Graphics Designer, Language experts, Students and end users

V. REFERENCES

[1]. Puhesynteesis, "Speech Synthesis".

[2]. "Speech Synthesis System for Indian Accent using Festvox" MOHAMMED WASEEM1, C.N SUJATHA2

[3]. Chomsky, N.; Halle, M. (1968), the Sound Pattern of English, Harper and Row, OCLC 317361

[4]. Harris, Z. (1951), Methods in Structural Linguistics, Chicago University Press, OCLC 2232282

[5]. Abrantes et al. 91 A.J. ABRANTES, J.S. MARQUES, I.M. TRANSCOSO, "Hybrid Sinusoïdal Modeling of Speech without Voicing Decision", EUROSPEECH 91, pp. 231-234.

[6]. A. Acero, L. Deng, T. Kristjansson, and J. Zhang, "HMM adaptation using vector Taylor series for noisy speech recognition," in Proceedings of ICSLP,Beijing, China, 2000.

[7]. M. J. F. Gales and S. J. Young, "7 Cepstral parameter compensation for HMM recognition in noise," Speech Communication, vol. 12, no. 3, pp. 231– 239, 1993.

[8]. S. J. Young and L. L. Chase, "Speech recognition evaluation: A review of the US CSR and LVCSR programmes," Computer Speech and Language, vol. 12, no. 4, pp. 263–279, 1998.