Open-Source Consumer-Grade Indic Text To Speech

Andrew Wilkinson[1], *Alok Parlikar*[1], *Sunayana Sitaram*[1],
Tim White[2], *Alan W Black*[1], *Suresh Bazaj*[2]

[1]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA
[2]Hear2Read, Indians for Collective Action, Palo Alto, CA
{aewilkin, aup, ssitaram, awb}@cs.cmu.edu; {tim, suresh}@hear2read.org

Abstract

Open-source text-to-speech (TTS) software has enabled the development of voices in multiple languages, including many high-resource languages, such as English and European languages. However, building voices for low-resource languages is still challenging. We describe the development of TTS systems for 12 Indian languages using the Festvox framework, for which we developed a common frontend for Indian languages. Voices for eight of these 12 languages are available for use with Flite, a lightweight, fast run-time synthesizer, and the Android Flite app available in the Google Play store.

Recently, the baseline Punjabi TTS voice was built end-to-end in a month by two undergraduate students (without any prior knowledge of TTS) with help from two of the authors of this paper. The framework can be used to build a baseline Indic TTS voice in two weeks, once a text corpus is selected and a suitable native speaker is identified.

Index Terms: speech synthesis, Indian languages, low-resource languages, open source, Android text to speech

1. Introduction

According to [1], in 2010, India had 15-22 million blind and over 100 million visually impaired persons. An adult goes blind every five minutes, and a child goes blind every 60 seconds. Without intervention, the number of people with vision loss could double from 2010 levels by 2020. Worldwide (including neighboring countries Bangladesh, Nepal, and Pakistan), the population of visually challenged persons for whom the primary language is an Indian language is probably 50% higher. Intelligible and natural-sounding TTS systems exist for many languages. As of the submission date of this this paper, Google TTS for Android supports 22 languages, of which only two (Hindi and Bengali) are from the Indian subcontinent.

This work will help bridge the digital divide for visually challenged persons who have been deprived of education and employment opportunities due to lack of consumer-grade TTS software for low-cost devices, such as sub-$100 (USD) Android phones and tablets.

The system uses the Indic frontend for grapheme-to-phoneme (g2p) conversion described in [2]. The Indic frontend was used for g2p conversion to build TTS synthesizers for various Indian languages for use with the Flite ("Festival Lite") Speech Synthesis engine [3]. The Indic frontend handles many phenomena common to Indian languages such as schwa deletion, contextual nasalization, and voicing. It also handles multi-script synthesis between various Indian-language scripts and English.

1.1. Relation to Prior Work

Most prior work on Indic TTS has focused on specific languages. Furthermore, it is generally proprietary, or available for research purposes only. The Festvox/Festival and Flite implementation described in this paper is the most widely available open-source TTS software for Indian languages, with the exception of eSpeak, which uses formant synthesis that requires significant listener training to comprehend.

Commercially available TTS software for Indian languages includes Google Android TTS for Hindi and Bengali and Apple iOS TTS for Hindi, which uses Nuance Vocalizer. Some of these commercial implementations have used elements of the Festvox framework.

Indian-language TTS work available for research purposes only includes work submitted at various Blizzard conferences and Indic TTS Consortium projects funded by the Technology Development for Indian Languages (TDIL) program of the Indian government. The TDIL consortium used the Festvox framework and the HTS-based engine, as described at [4].

For the 2013 Blizzard challenge, a team from the Indian Institute of Science, Bangalore, developed a unit selection–based TTS system for Kannada and Tamil, which does not use Festvox or Festival [5].

We used the open-source Kannada corpus published by IIIT Hyderabad [6] and the Marathi corpus published by the Center for Indian Language Technology at IIT Bombay [7].

1.2. Novel Contributions

We provide a repeatable process to generate new voices suitable for an end user, by supporting the steps from finding prompts, to recording, to building the voice, to making it available and practical to use on Android.

Historically, TTS systems have been developed one language at a time. Efforts by different institutions to develop TTS for Indic languages in this way have been ongoing for some years. This approach requires significant manpower and financial resources. Our approach expands upon the Festvox framework by leveraging the commonalities between Indian languages. Data from the 2001 census show that about 93% of the population of India use one of the country's top 12 languages as their primary language. With our framework, we have built

voices for eight of these 12 languages, representing 80% of the population; and our framework can be expanded for use in synthesizing all the official languages of India.

Recently, the baseline Punjabi TTS voice was built end-to-end in a month by two undergraduate students (without any prior knowledge of TTS) with help from two of the authors of this paper. The framework can be used to build a baseline Indic TTS voice in two weeks, once a text corpus is selected and a suitable native speaker is identified. The availability of sufficient open-source corpora in Unicode format varies significantly from language to language. Auditioning and selecting a suitable speaker is a function of the location and availability of a recording studio.

Our framework does not require the creation or use of a pronunciation lexicon for any of the Indian languages. In English TTS, either very complicated letter-to-sound rules are needed, or a lexicon of tens of thousands of entries, or both. This is because the relationship between English orthography and pronunciation is relatively convoluted. The Indian languages are simpler in this regard. Thus, through a combination of taking advantage of commonalities between the scripts and encoding a manageable number of language-specific letter-to-sound rules, we avoid having to spend time specifying the pronunciations of individual words.

Most of the published work on Indic TTS has used Festival, which is a runtime system designed to run on Unix servers. We use Flite (described in the next section) in a consumer-friendly app that runs on sub-$100 (USD) Android phones and tablets using less than a third of the available processing power. This leaves sufficient processing power and memory for other applications to run simultaneously without loss of performance.

2. Flite Overview

Flite is a small, fast-runtime open-source TTS engine developed at Carnegie Mellon University. It is an alternative to the Festival TTS engine, for voices built using the Festvox suite of voice building tools. Flite is part of the suite of free speech synthesis tools which include Edinburgh University's Festival Speech Synthesis System and Carnegie Mellon University's Festxox project, which provides tools, scripts, and documentation for building new synthetic voices. Flite is less intensive in memory and computation than Festival.

Flite is written in ANSI C, and is designed to be portable to almost any platform, including low-end smartphones, tablets, PCs, and servers which must serve synthesis to many users.

2.1. Offset-Based Indic Frontend

The Flite Indic frontend uses an offset-based approach instead of creating explicit support for each Indian script. Chapter 12 of the Unicode specification [8] specifies offset-based character tables for each script, with each containing up to 128 characters. Within each script, there is a fixed sequence of characters which makes it easy to build general rules for phenomena common to Indian languages such as schwa deletion, contextual nasalization, and voicing. This makes it possible to have a single mapping with offsets for all scripts for Indian languages.

For example, the Devanagari block begins at (hexadecimal) codepoint 0900, the Bengali block begins at 0980, and the Tamil block begins at 0B80. The Devanagari letter ⟨क⟩ ⟨k⟩ occupies position 22 in the Devanagari block, or 0915; the corresponding Bengali letter ⟨ক⟩ occupies position 22 in the Bengali block, or 0995; and the corresponding Tamil letter ⟨க⟩ occupies position 22 in the Tamil block, or 0B95. By identifying the range in which each character falls, and subtracting the beginning value of its block, we map corresponding characters in different scripts to one representation, which is used as a basis for further processing.

2.2. Android Flite port

The "Flite for Android" project is an open-source project that provides a wrapper for the Flite engine to run in the Android environment. It is available to build from source, and is also available in the Google Play store as "CMU Flite Text to Speech."

The project integrates into Android's TTS API so that once the app is installed on an Android device, the Flite engine is available as an alternative to the default synthesizer. The project is mostly Java code that contains callbacks into Flite via the Java Native Interface. The app supports CLUSTERGEN voices as downloadable voice files, and currently supports English, Gujarati, Hindi, Marathi, Kannada, Tamil, and Telugu. It is possible to use unit selection voices (Festival "clunit") by compiling the app with the appropriate voice libraries. The app defaults to streaming synthesis under the Android framework, but also supports batch synthesis.

The Indian languages that have received the most support to date are Hindi, Tamil, and Marathi. Ongoing efforts support periodic upgrades to all voices, which are available for download within the app without the need to update the app itself.

3. Building Indic Voices

CLUSTERGEN is a method for building statistical parametric synthesizers from databases of natural speech. Although the result is not as crisp as a well-crafted unit selection voice, this method makes it much easier to build a clear synthetic voice that models the original speaker well.

This section describes the process and tools developed to build Indic voices using CLUSTERGEN, though many of the steps are the same for other synthesis methods. The steps described through subsection 3.7 are carried out on Unix server(s) and Apache web server(s).

3.1. Open-Source Indic Corpus

We follow the techniques and requirements described in [3] for designing a good corpus for obtaining recordings for use in speech synthesis. The challenge is in gathering open-source corpora that can be published for use by others in conformance with an MIT X11–type license. This is accomplished by using source materials:

- With expired copyright (e.g., Hindi novels by Munshi Premchand).
- Released as open source (e.g., Bengali Wikipedia).

- With written permission from the copyright holder (e.g., [9]).

3.2. Qualifying Voice Talent for Recording

CLUSTERGEN uses a diphone database (referred to here as "Voice") to synthesize speech for arbitrary text. This database is built from speech recorded by a native speaker, as described in [3].

The quality of recorded prompts selected from the open-source corpus plays a critical role in the quality of synthesized speech. All recordings must be made in an anechoic chamber with a high-quality microphone using a pop filter and without dynamic range compression. The speaker should be relaxed; keep a constant distance from the microphone; and not be suffering from a cold, a cough, or allergies, etc. These conditions are necessary but not sufficient. The speaker should speak in a natural style; that is, they should not speak differently just because they are reading rather than speaking extemporaneously.

At the same time, for reasons of consistency, the speaker should limit prosodic variations as much as possible, and maintain a reasonably constant rate of speech. Accomplishing this is not easy for most people, as it requires them to speak differently than their normal speech pattern and yet sound natural. Doing so, however, helps with fundamental frequency (f0) extraction and phoneme duration calculations.

We ask potential speakers to record 15-20 test sentences selected from the list of prompts that will be recorded later to build the Voice. These test sentences are evaluated by researchers who have been working in this field and can recognize prosodic variation, changes in rate of speech, and other properties, based on their prior experience. This step is quite subjective and is more art than science. The test recording need not be made in a studio, since it is not used to build a Voice.

The next step is to record the first 100-200 prompts in a studio, as described above. These recordings are used to build a test Voice. The purpose of building a test voice is twofold:

- Ensure that the f0 variance is minimal.
- Ensure that the synthesized speech built using the test voice has reasonable resemblance to the recorded speech.

Speech generated using the test voice may not necessarily be intelligible, depending on the number of prompts recorded, the language, and the speaker. Once again, we relied on the subjective judgment of the researchers to decide if we should proceed with the next step to record all the prompts needed to build a production-quality Voice.

3.3. Building CLUSTERGEN Voices

The selected Voice talent then records as many prompts as necessary to generate intelligible speech. We limit each recording session to an hour. For most people, the vocal cords are tired by the end of the hour, such that the voice quality degrades. Most speakers are able to record about 250 sentences in an hour, with rest breaks after every 50 sentences. For an average duration of six seconds per sentence, this results in 25 minutes of recorded speech.

Having a second native speaker listen in during the recording session allows the speaker to receive feedback on mispronunciations, unintended noise (such as coughing, paper shuffling, or foot tapping), and intonation. It is much easier to rerecord a sentence immediately, or to skip it if there are spelling errors that make the word difficult to pronounce, than to address the problem later on.

If a listener is not available during the recording session, then the recordings must be played back by a native speaker later to make sure that they match the prompts and, if necessary, either edit the text to match the recording or remove the recording where editing of the prompt is not possible.

The amount of recording needed to build good CLUSTERGEN Voices depends on the language, the prompt list, and the speaker. For some languages, we have built a good Voice with about an hour of recorded speech, while others have required up to two hours.

We have experimented with deleting recordings with the highest distortion as measured by the mel-cepstral distortion (MCD) metric. In some cases it improved the Voice quality, while in other cases it did not. We believe it is worthwhile to try it.

3.4. A/B Listening Tests

Speech research often requires scientists to conduct subjective listening tests to compare different methods against each other. For example, A/B tests are used to determine if a new synthetic voice is better than an earlier voice that might have been built using a different algorithm or dataset.

TestVox is an open-source web-based framework for running subjective A/B listening tests. It allows one to quickly set up listening tasks.

Typically, TestVox is configured to play 15-20 sentences of approximately 5-6 seconds each. The order in which the sentences are played, and the order in which the synthesized speech alternatives are played, can both be randomized.

3.5. Numbers to Text Transliteration

Indian language texts may employ the numerals native to the script of the language, or may employ the standard numerals common to most of the world today (known as "Arabic" or "Indo-Arabic" numerals; we refer to them as "English" numerals for simplicity). In modern Indian-language texts, English numerals are more commonly used than native numerals.

Speaking numbers in Indian languages requires use of a pronunciation lookup table for all numbers between zero and 100, because these numbers take idiosyncratic forms that cannot be deterministically generated. Combination rules are used to compose higher-order numbers.

Surprisingly, the complete number vocabulary information (for integers through 100, plus fractions and higher powers of 10) is readily available only for some of the Indian languages, such as Hindi, Marathi and Gujarati (e.g., at [10]). Finding reliable information is particularly daunting in cases where many or most native speakers have switched to writing and speaking numbers in English instead of the native language, such as Tamil. It required consulting elderly speakers to find

out how numbers were spoken in Tamil at one time, and sometimes different speakers have differing versions of the same number.

For this reason, we synthesize numbers written with English numerals in English, and numbers written in a native script in the corresponding language. This reflects a compromise between respecting the desires both of authors who use English numerals and wish the text to be accessible to a wide audience (including people who may not have full familiarity with the native number system), and of authors who use native numerals and wish to continue the traditions of the language. We plan to make these representation options (English, native, or mixed numbers) a choice in the future for the user.

Further design decisions on this subject are described in [2].

3.6. Collecting Actionable Feedback

For each language, 10 or more volunteer native speakers were recruited to provide actionable feedback on the Voice. This is an iterative process where rules are added or edited based on the feedback, until an acceptable quality level is reached. A web-based application was built to collect feedback on accuracy of pronunciations in the synthesized speech.

The web application allows the volunteers to listen to synthesized speech for 50 representative sentences and comment on awkward or mispronounced words, one sentence at a time. The user interface allows the volunteers to identify mispronounced words and to write a description of why a word is wrong. The best actionable feedback is when the comment identifies the incorrect syllable. Multiple comments can be entered for a sentence if multiple syllables are mispronounced. The comments are captured via email sent to the researcher(s) for analysis and follow-up action. For example:

From: [Volunteer's Name] [Volunteer's Email]
Voice: sun400_ta.flitevox
Test sentence 13:
உங்கள் கல்வி பின்னணி, **மற்றும்** தற்போதைய வேலை **இவற்றைப் பற்றிச் சற்று** கூற முடியுமா?
Comments: Correction - in all these words tra sounds like ra
Test sentence 2: **இதற்கான** வரவேற்பு எப்படி இருந்தது அங்கே?
Comments: Correction - Idarkan**a** (a is missing in the pronunciation)

The biggest challenge in this process is training the volunteers to provide actionable feedback to the researcher who may not be a linguist. Simply stating that certain pronunciations are bad does not help. Similarly, sending an .mp3 recording of how the volunteer speaks the sentence is generally not very helpful.

Feedback is qualified by ensuring that multiple volunteers have responded the same way with only a few (or no) dissenters.

3.7. Exception Rules and Language-Specific Lexicon

After collecting feedback from testers on a voice, we use the feedback to make corrections and improvements to the voice. This can take several forms.

In some cases, the letter-to-sound rules for the language need to be updated to reflect regular rules that the language exhibits in the mapping from orthography to pronunciation, which can be predicted by context. Such rules are intrinsically understood by native, literate speakers, such that when they are violated by the synthesized voice it is apparent to the speaker; but they are not evident to a nonnative speaker.

In other cases, one or a small number of words provide an exception to a rule, and these words can be added to a pronunciation lexicon for the language. In still other cases, a letter commonly maps to more than one different phoneme in a way that is not systematically predictable from context, so we must discover the variants and input them into the lexicon to train the system to achieve the correct pronunciation.

As an example of a regular rule, in Tamil, in words written with a double ⟨ற⟩ ⟨r⟩, the pronunciation is as /tr/ for those characters. E.g., ⟨மற்றும்⟩ ⟨marrum⟩ "and" is pronounced as /matrum/.

As an example of letters representing multiple phonemes, in Marathi, the letter ⟨ज⟩ can represent either /z/ or /ʤ/, as in ⟨दरवाजा⟩ /dərəvaza/ "door" and ⟨समाज⟩ /səmaʤ/ "society." In Hindi, ambiguity for this and other letters is resolved with the use of the nukta diacritic, but Marathi does not use the nukta, so we use native-speaker feedback initially to learn which phoneme applies where, and use that information in building a new version of the voice.

3.8. Porting to Android After Successful QA

After testing for a version of a voice is completed and the voice has passed quality assurance criteria, the voice is made available for download in the Android app. No additional processing is necessary at this stage; the same file is used in the app as for the UNIX-based Flite platform.

3.9. Bilingual (Indic & English) TTS

We are now looking into creating versions of the Indic voices that can speak both the primary language and also high-quality English, for the common situation in which English text is encountered in a primarily Indian-language document. Currently, English synthesis is supported by our Indic voices by mapping English phonemes to the Indian-language phonemes and synthesizing as if the English word were a word in the Indian language. The resulting English synthesized speech sounds heavily accented, since using this mapping is inferior to performing a voice build using English prompts with the English lexicon and rules.

Several of our voice talent speakers have recorded sets of English prompts in addition to the Indian-language prompts. We have built separate English voices using these prompts. The goal is to use both the English prompts together with the Indian-language prompts recorded by the same speaker in a unified voice build that will synthesize both languages well. In addition to predicting correct pronunciations for English based on rules, this approach will improve the English quality by incorporating English phonemes that are not present in the other language. At the same time, it will retain pro-

nunciation characteristics of Indian English, such that when the voice switches between languages, the result naturally reflects the comfortable code-switching exhibited by many speakers and does not switch to a jarring American accent, for example.

4. Availability

The current version of Flite has support for Indic voices created using Festvox. Voices for Hindi, Gujarati, Marathi, Tamil, and Telugu are available for download, with Kannada and Bengali to come next.

The Flite TTS for Android application is also built with support for Indian languages, and the same voices can be downloaded in the app.

Documentation is provided in the Festvox manual for building voices using the Indic frontend, and for adding support for new voices. [11]

Flite binaries and voices (Indic and US English):
`http://www.festvox.org/flite/packed/flite-2.0`

Indic voice demos:
`http://tts.speech.cs.cmu.edu:8084`

Festival Indic voices:
`http://www.festvox.org/cmu_indic/index.html`

Festival and Festvox binaries:
`http://www.festvox.org`

CMU Flite TTS Android app:
`https://github.com/happyalu/Flite-TTS-Engine-for-Android/`

`http://play.google.com/store/apps/details?id=edu.cmu.cs.speech.tts.flite&hl=en`

TestVox:
`http://bitbucket.org/happyalu/testvox/wiki/Home`

Hear2Read demo:
`http://www.hear2read.org/demo/app.php`

5. Conclusions

We described the design and development of an open-source TTS framework for 12 Indian languages. The feedback on Release 1.0 of the first six languages has been positive, though more work is needed. We have received many requests to port the Flite engine to the Windows operating system using the Speech API, which confirms the technology and reinforces our mission to empower the visually challenged to become productive members of society.

Since starting this project, we have learned that many users with good eyesight are also interested in using Indic TTS. These include:

- People living in areas where the native language is different from their own native language. This is a very common situation in any multilingual country. It is certainly true in India, which has 22 official languages. People learn to understand the spoken words after living in the area for a while. However, they rarely learn to read it. TTS gives them the ability to read local language content.
- People in an environment where reading on a screen is not feasible. Examples include crowded trains and buses, exercising in the gym or running, sitting on a plane when other passengers want all lights to be off, etc.
- People who want to learn a new language. This is especially true for children of immigrants to a new country. The United States and Canada are good examples.

6. Acknowledgements

This research was partially supported by Hear2Read donors and volunteers. We would like to thank all those who helped us over the past several years with design, development, open-source corpus collection, recording the prompts, multiple rounds of feedback, and the website.