6th Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU 2018)
29-31 August 2018, Gurugram, India

A Step-by-Step Process for Building TTS Voices Using Open Source Data and Frameworks for Bangla, Javanese, Khmer, Nepali, Sinhala, and Sundanese

Keshan Sodimana, Pasindu De Silva[2], Supheakmungkol Sarin[1], Knot Pipatsrisawat[1], Oddur Kjartansson[1], Martin Jansche[1], Linne Ha[1]

[1]Google,[2]Google (on contract from Teledirect Pte Ltd)

{ksodimana,pasindu,mungkol,thammaknot,oddur,mjansche,linne}@google.com

Abstract

The availability of language resources is vital for the development of text-to-speech (TTS) systems. Thus, open source resources are highly beneficial for TTS research communities focused on low-resourced languages. In this paper, we present data sets for 6 low-resourced languages that we open sourced to the public. The data sets consist of audio files, pronunciation lexicons, and phonology definitions for Bangla, Javanese, Khmer, Nepali, Sinhala, and Sundanese. These data sets are sufficient for building voices in these languages. We also describe a recipe for building a new TTS voice using our data together with openly available resources and tools.

Index Terms: text-to-speech, linguistic resources, low-resourced languages, open source

1. Introduction

As natural human-machine communication becomes more prevalent, text-to-speech (TTS) systems play an equally important role as speech recognition systems. From our experience building TTS voices for many low-resourced languages, we have observed the following common obstacles that prevent more rapid research: the lack of language resources (e.g., audio, lexicon, linguists) and the complexity of voice building process.

Without resources for the language, voice building becomes a much more challenging problem. Even when some data are available, we still need a native speaker (usually a linguist) to help curate various resources and to provide feedbacks on the resulting voices. Because voice building tools often require non-trivial amount of technical proficiency to configure and use, this working scheme may create a heavy dependency on the engineering team. This problem is further amplified if the operations are scaled across many languages in different time zones.

In this paper, we aim to address some of these obstacles. First, we present a set of data that we open sourced for the public to use freely. These data consist of audio recordings of short phrases/sentences, a pronunciation lexicon, and a phonology definition for each of the following 6 low-resourced languages: Bangla (Bangladeshi Bengali), Javanese, Khmer, Nepali, Sinhala, and Sundanese. Moreover, to help guide researchers who are interested in experimenting with our data, we also present a step-by-step recipe for building TTS voices from our data with a web-based open source tool. The tool abstracts away the technical burdens of configuring and maintaining complex systems. We hope that our contributions here will help spur more interests in natural language processing for low-resourced languages.

In the next section, we discuss related work. In Section 3, we describe the open source resources that we are contributing to the community and discuss TTS tools that we will utilize for building TTS voices. Then, in Section 4, we present a step-by-step guide for building a new TTS voice using the tools. In Section 5, we discuss the quality of our data. Finally, in Section 6, we wrap up with some discussions about future directions and some conclusions.

2. Related Work

All the languages discussed in this paper are considered low-resourced languages. Therefore, work in this area is minimum for some languages. However, some work has been done on building Sinhala [1] and Bangla [2] voices using Festival [3]. Both of these voices are based on the unit selection technique [4]. These Bangla and Sinhala Festival voices have been open sourced and can be downloaded from their respective websites[1]. In a more recent work, a parametric voice building technique has been used to build a Bangla voice from audio collected from multiple speakers via a crowdsource approach [5]. A voice building process for any new language based on the MaryTTS system has been discussed in [6].

Our key contributions in this work are (1) addressing the lack of TTS data for several low-resourced languages by making available linguistic resources for Bangla, Javanese, Khmer, Nepali, Sinhala, and Sundanese. The data will be made available under free and a flexible license. (2) providing a recipe for utilizing these resources through freely available TTS tools.

3. Available Resources and Tools

In this section, we describe the resources that we are open sourcing in details and also discuss a few tools that we will use in our voice building recipe in the next section.

3.1. Linguistic resources

Typically, the bare minimum types of data required to build a new TTS voice are (i) audio data (with associated transcript) (ii) a pronunciation lexicon and (iii) a phonology definition. We are making all these resources available for all 6 languages mentioned earlier. Unless mentioned otherwise, all the data have been released under the Creative Commons Attribution 4.0 international license (CC BY 4.0) [7].

3.1.1. Audio data

This set of data contains audio-transcript pairs for each language. Each entry consists of an audio file and a (normalized) text transcript of the audio. Audio recordings are in RIFF WAVE format. The accompanying `line_index.tsv` file has

[1]https://github.com/firojalam/Katha-Bangla-TTS and http://ucsc.cmb.ac.lk/ltrl/projects/si/textonly_old.htm

10.21437/SLTU.2018-14

Language	Gender	Speaker count	Number of audio files	Total Duration (hours)	Location
Bangla	M	6	1819	2:56:18	http://www.openslr.org/37/
Javanese	F	20	2864	3:31:13	http://www.openslr.org/41/
	M	21	2958	3:28:13	
Khmer	F	17	2906	3:58:00	http://www.openslr.org/42/
Nepali	F	19	2064	2:47:45	http://www.openslr.org/43/
Sinhala	F	13	2064	3:22:49	http://www.openslr.org/30/
Sundanese	F	21	2401	3:12:21	http://www.openslr.org/44/
	M	22	1812	2:10:22	

Table 1: *A catalogue showing information about recorded audio files in different languages and genders.*

Language	Number of lexicon entries
Bangla	69277
Javanese	54322
Khmer	70671
Nepali	66119
Sinhala	52573
Sundanese	42816

Table 2: *Number of lexicon entries for each language.*

the normalized transcript of the recorded audio along with the ID of the corresponding audio file. Table 1 shows the properties and location of the audio corpus of each language.

The audio data were collected from a group of volunteers in each language. The volunteers were between the ages of 20 and 35. They were asked to read short sentences, each of which contains 5 - 20 words. The texts used for recording were either extracted from wikipedia, general websites, or were declarative sentences created by native speakers of the target languages. The recording was conducted in quiet environments: either a sound studio or a quiet room with a soundproof booth. Moreover, all audio files have passed through a QC process to ensure good audio quality, absence of background noise, and match between recorded audio and text transcript.

3.1.2. Pronunciation lexicon

A pronunciation lexicon is a map from words to their pronunciations written in a certain convention (e.g., IPA) [8]. Lexicons are important in a TTS system because they allow the system to turn training text or input text into the underlying pronunciations (e.g., sequences of phoneme symbols). The system can then synthesize the voice based on the resulting pronunciations.

These lexicon files can be found in Google Internationalization team's language resources repository [9] for each language. For example: the Javanese lexicon can be accessed at `https://github.com/googlei18n/language-resources/blob/master/jv/data/lexicon.tsv`. Other lexicons can be found in their respective language's folder. Table 2 lists the amounts of lexicon entries we have made available for each language.

We created these pronunciation lexicons by working with native speaker linguists to create guidelines for phonemic transcriptions. Then, we utilized teams of linguists (or trained native speakers) to manually transcribe words based on the guidelines. Every word entry has also gone through a QC process to ensure correctness. Words in these lexicon were obtained by crawling wikipedia pages and websites in these languages. We aggregated and ranked the words according to their frequencies of occurrences. The linguists then transcribed the top words in the lists.

3.1.3. Phonology

The phonology definition of a language defines the set of phonemes and their properties. For example, it lists all possible consonants, vowels, and indicates properties such as place of articulation, nasality, and voicing properties. Phonology definitions are important for voice building because they provide features and clues for voice building algorithms to accurately learn an acoustic model for the language. A phonology definition for each language is available at the language resources Github repository [9]. For example, the phonology file for Javanese is at `https://github.com/googlei18n/language-resources/blob/master/jv/festvox/phonology.json`.

3.2. Voice building tools

In the subsequent section, we will present a recipe for building a TTS voice. The underlying voice building tools used are Festival [3] and Merlin [10].

Festival [3, 11] is a popular TTS engine used by many researchers. Festival provides routines to build parametric voices using its clustergen algorithm. It also contains some useful preprocessing sub-components that other engines rely on.

Merlin is a TTS engine which uses neural networks to build voices. It relies on parts of Festival to preprocess input data (e.g., to generate features).

Setting up either Festival and Merlin for voice building and maintaining such systems are not trivial tasks. As a result, we presented, in an earlier work, Voice Builder, a framework that simplifies voice building process by wrapping both Festival and Merlin in an easy-to-use web frontend [12]. Voice Builder is also an open source tool that can be used by any interested researchers to build voices and manage experiments. Currently, Voice Builder is configured with both Festival and Merlin as underlying voice building engines[2].

4. Step-by-step voice building process

Now we will provide detailed instructions on how to build a TTS voice from the data we described in the previous section. We will use Voice Builder as a tool for building the voice, instead of interacting directly with voice building engines like Festival or Merlin. As a prerequisite, please download language resources (LR) repository from [9]. This repository contains some useful scripts that we will utilize in this section.

[2]Readers interested in voice building tool boxes may also check out [13, 14].

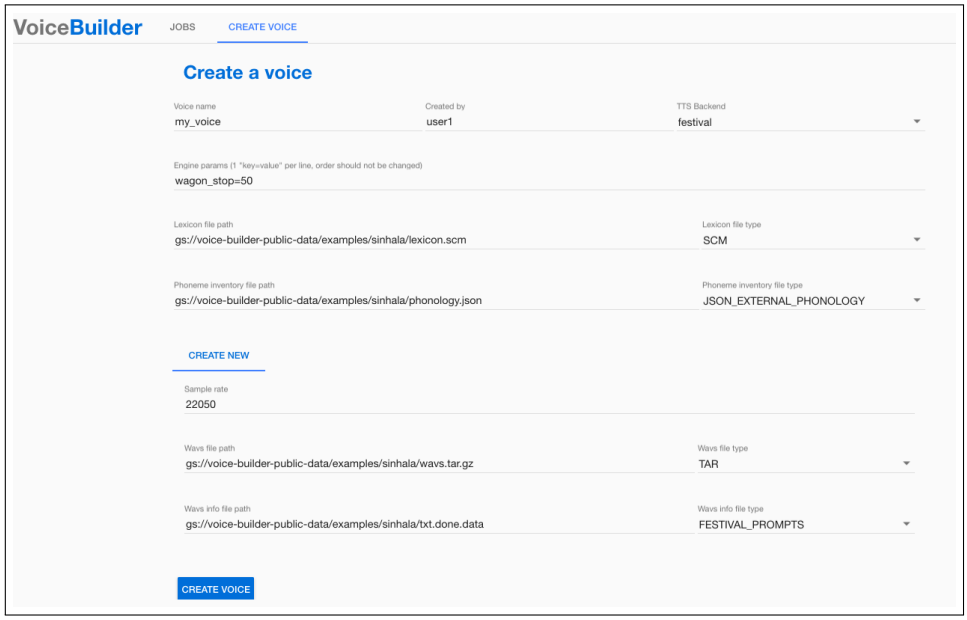

Figure 1: *A screenshot of a page for creating a voice in Voice Builder. Note that all data paths used in the form are GCS paths.*

Voice Builder is a web-based tool that simplifies voice building process by abstracting away low-level interactions with voice engines (such as Festival or Merlin). Behind the scene, Voice Builder utilizes TTS engines (wrapped in Docker container) on Google Cloud Platform (GCP). The use of GCP, which is a publicly accessible platform, removes a lot of technical burdens from the users. However, before we can use it for building a voice, we need to make the data accessible by Voice Builder. This is achieved by uploading the data to Google cloud storage (GCS) [15]. We will now present a step-by-step process for preparing the data and uploading them to GCS.

1. Lexicon: This contains all the words and their transcriptions. They are typically formatted as TSV, because this format is simple and easy to understandable. The first column of the TSV file is the word (i.e., spelling) and the second column is the pronunciation (e.g., in IPA). The syllable boundaries are marked with "." in the pronunciation. For example,

   අංකනය /a ŋ . k ə . ɳ ə . j ə/

   In order for the lexicon to be used by Festival or Merlin, it needs to be in the Festival lexicon format. We have included lexicon in both the TSV and Festival format in our open source repository. However, if any user would like to make modifications to the lexicon, editing the TSV file is probably easier. The edited file can then be converted into the Festival format by using `festival_utils/festival_lexicon_from_tsv.py`, which can be found in LR. For example, running

   ```
   $ cat si/data/lexicon.tsv | python
   festival_utils/festival_lexicon_from
   _tsv.py > si/festvox/lexicon.scm
   ```

   will convert the Sinhala TSV lexicon into the Festival format (`lexicon.scm`).

2. Audio data: These are the recordings that will be used for training a voice model. Each file should be in the RIFF WAVE format. Then, the folder containing all the audio files should be compressed in the tar format. For example, to compress all the audio file in folder "audio" into an archive named "audio.tar", run

   ```
   $ tar -cf audio.tar audio
   ```

3. Audio and transcript: This is the mapping between audio files and their corresponding text transcripts. It is stored in a TSV file where the first column is the audio file name (without the ".wav" extension) and the second column is the corresponding (segmented) text. For example,

   sin_2241_0397568785 මෙහි මුද්‍රාව අභය

   In our data sets, each audio file name also encodes the speaker ID. In the above example, 2241 is the speaker ID, and 0397568785 is a random string associated with that audio entry. Thus, all audio files recorded from this same speaker will have the prefix sin_2241. This mapping file needs to be in the Festival format for voice building. Again, we have made the mapping files available in both formats in our repository. If needed, the TSV file can be converted to the Festival format using `festival_utils/prepare_prompts.py`, which can be found in LR as well. For example, running

   ```
   $ cat si/data/line_index.tsv | python
   festival_utils/prepare_prompts.py >
   si/festvox/txt.done.scm
   ```

   will convert the TSV mapping file into a mapping file in Festival-compatible format.

4. Phonology: This contains the phoneme inventory of the language. This file is in JSON format, which is described in details here:

   `https://github.com/googlei18n/language-resources/blob/master/docs/JSON_PHONOLOGY.md`[3]

[3]Going over the format of this file is beyond the scope of this paper.

This JSON file can be used directly by the voice building process.

Once all the data are in the right format, we simply need to upload them onto a Google cloud storage folder (also known as a "bucket"). We will use the `gsutil` command line tool to upload the data. To download this tool, which is packaged as "Cloud SDK", and get started, please follow the instruction at `https://cloud.google.com/storage/docs/quickstart-gsutil`. Once you finish this step, you should have Cloud SDK installed on your local machine and have created a project on Google Cloud. The next step is to create a new storage bucket. This can be done by running

```
$ gsutil mb gs://<bucket>/
```

where `<bucket>` is an arbitrary bucket name. Once done, we can upload data onto this bucket with the following command:

```
$ gsutil cp <file> gs://<bucket>/<path>
```

This command will upload `<file>` from a local machine onto GCS bucket `<bucket>` at `<path>` relative to the root of the bucket. For example, to upload a lexicon file in Sinhala onto `my_bucket` at path `si/lexicon`, run

```
$ gsutil cp si/festvox/lexicon.scm
gs://my_bucket/si/lexicon.scm
```

All uploaded files need to be made publicly accessible (so that Voice Builder can access them). The following command can be used to make a file in a bucket public:

```
$ gsutil acl ch -u AllUsers:R
gs://<bucket>/<path>
```

Once all the above data are in Google cloud storage, we can simply go to Voice Builder demo's "create voice" page at `http://tinyurl.com/voice-builder` [4]. We then need to fill in the paths to the resources that have been uploaded to GCS. Each GCS path has the format `gs://<bucket>/<path>/<to>/<file>`. Figure 1 shows some example paths in a bucket called `voice-builder-public-data`. Currently, Voice Builder supports 2 options for voice building engine; Festival or Merlin. Once an engine is selected and the rest of the form filled out, you can simply click the "create voice" button at the bottom of the page. This will initiate a "job" that trains a voice model based on the data and parameters in the form. Figure 1 shows a screenshot of the "create voice" page in Voice Builder. Notice the field called "Engine params" on the second row of the form. This field allows users to adjust parameters to be passed to the underlying voice building engine. Users may leave the value of this field unchanged or experiment with other values for better results. Please consult the manual of the respective engine for complete explanation of the options.

The "jobs" page of Voice Builder lists all the jobs in the system and indicates their statuses. Once training is completed, you can click on the job ID to visit the page for that job. On this page, you can click a button to deploy the trained voice model for audio synthesis. After a few seconds, once the model is fully deployed, you can enter arbitrary text in the target language into the text box at the bottom of this page and click a button to synthesize the voice.

[4]The use of computing resources on the demo is free. If any users want to run their own instance of Voice Builder, please follow the instructions at `https://github.com/google/voice-builder`. The pricing of GCP resources can be found at `https://cloud.google.com/pricing`.

5. Data quality

The audio data sets released as part of this paper are all multi-speaker. We have used Google's internal TTS system to build voices from these data. In fact, these voices have been made available through services such as Google Translate [16]. We have also evaluated the quality of these voices by measuring the Mean Opinion Score (MOS), which ranges from 1 (unnatural speech) to 5 (natural speech) [17]. Table 3 shows the MOS of these voices, which were built using Google's proprietary algorithm similar to that described in [5]. All voices listed here are monolingual, multi-speaker voices.

Language	Mean opinion score
Bangla (M)	3.403 ± 0.098
Javanese (F)	3.998 ± 0.103
Khmer (F)	3.512 ± 0.144
Nepali (F)	3.705 ± 0.139
Sinhala (F)	3.285 ± 0.161
Sundanese (F)	3.669 ± 0.097

Table 3: *Mean opinion score for a voice of each language. The genders of the tested voices are indicated in parentheses.*

It is important to note that different voice building tools may produce voices with different quality and characteristics, even from the same training data set. Moreover, each voice building engine contains a number of training parameters that can be tweaked for different results. We encourage interested researchers to experiment with our open source data to produce voices that fit their use cases. Our data here show that the resources discussed in this paper are more than sufficient for building intelligible and good quality voices for these languages. In our other work, we have explored combining the Javanese and Sundanese data sets together with an Indonesian corpus to create an even higher quality voice for Javanese and Sundanese [18].

6. Conclusions

In this paper, we aim to address some challenges in the area of TTS research for low-resourced languages. We presented data sets of recorded audio in 6 low-resourced languages and linguistic resources necessary to build TTS voices in these languages. Moreover, we describe a relatively simple method for utilizing the presented data to create a TTS voice through a web application called Voice Builder. By using Voice Builder, a lot of technical complexities are abstracted away from the voice building process, allowing TTS research and experimentation to be more accessible to a wider set of audience. We hope that this work will invite more interests in natural language processing research on low-resourced languages.

7. Acknowledgements

We would like to thank all the volunteers that have contributed their time and donated their voices to the collection.

8. References

[1] R. Weerasinghe, A. Wasala, V. Welgama, and K. Gamage, "Festival-si: A sinhala text-to-speech system," in *Text, Speech and Dialogue, 10th International Conference, TSD 2007, Pilsen, Czech Republic, September 3-7, 2007, Proceedings*, 2007, pp. 472–479.

[2] F. Alam, S. M. Habib, and M. Khan, "Bangla text to speech using festival," in *Conference on Human Language Technology for Development (HLTD 2011), Alexandria, Egypt*, 2011, pp. 02–05.

[3] A. W. Black and P. A. Taylor, "The Festival Speech Synthesis System: System documentation," Human Communciation Research Centre, University of Edinburgh, Scotland, UK, Tech. Rep. HCRC/TR-83, 1997, avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

[4] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proceedings of the Acoustics, Speech, and Signal Processing, 1996. On Conference Proceedings., 1996 IEEE International Conference - Volume 01*, ser. ICASSP '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 373–376. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.1996.541110

[5] A. Gutkin, L. Ha, M. Jansche, O. Kjartansson, K. Pipatsrisawat, and R. Sproat, "Building statistical parametric multi-speaker synthesis for bangladeshi bangla," in *SLTU-2016 5th Workshop on Spoken Language Technologies for Under-resourced languages, 09-12 May 2016, Yogyakarta, Indonesia; Procedia Computer Science*, 2016, pp. 194–200.

[6] S. C. Pammi, M. C. Oliva, and M. Schrder, "Multilingual voice creation toolkit for the mary tts platform," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. ELRA, 5 2010.

[7] "Creative commons attribution 4.0 international license," https://creativecommons.org/licenses/by/4.0.

[8] A. Brown, "International phonetic alphabet," *The Encyclopedia of Applied Linguistics*, 2013.

[9] "Google internationalization language resources," https://github.com/googlei18n/language-resources.

[10] Z. Wu, O. Watts, and S. King, *Merlin: An Open Source Neural Network Speech Synthesis System*. ISCA, September 2016, pp. 218–223.

[11] P. A. Taylor, A. Black, and R. Caley, "The architecture of the festival speech synthesis system," in *The Third ESCA Workshop in Speech Synthesis*, Jenolan Caves, Australia, 1998, pp. 147–151.

[12] P. D. Silva, T. Wattanavekin, T. Hao, and K. Pipatsrisawat, "Voice builder: A tool for building text-to-speech voices," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, may 2018.

[13] T. Schultz, A. W. Black, S. Badaskar, M. Hornyak, and J. Kominek, "SPICE: Web-based tools for rapid language adaptation," in *INTERSPEECH*, 2007.

[14] A. Conkie, T. Okken, Y.-J. Kim, and G. Di Fabbrizio, "Building text-to-speech voices in the cloud." in *LREC*, 2012.

[15] "Google cloud storage," https://cloud.google.com/storage/.

[16] "Google translate," https://translate.google.com.

[17] I. Rec, "P. 800.1, mean opinion score (mos) terminology," *International Telecommunication Union, Geneva*, 2006.

[18] J. A. E. Wibawa, S. Sarin, C. F. Li, K. Pipatsrisawat, K. Sodimana, O. Kjartansson, A. Gutkin, M. Jansche, and L. Ha, "Building open javanese and sundanese corpora for multilingual text-to-speech," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 7-12 May 2018, Miyazaki, Japan, 2018, pp. 1610–1614. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2018/pdf/8888.pdf