Malayalam Text-to Speech system

Guide: Jiby J P
Project Coordinator: Manilal D.L

Group 16
Kurian Benoy 34
kurianbenoy@mec.ac.in

Objective

- To build a Text to Speech(TTS) system in Malayalam
- Obtain the state of art result

- Module1 : EDA, dataset collection
- Module2: Train first TTS system in Malayalam
- Module3: Fine tune TTS system
- Module4: User Interface

Work Done

Mozilla TTS on top implementation

This project is a part of Mozilla Common Voice. TTS aims a deep learning based Text2Speech engine, low in cost and high in quality.

How can I train my own model?

1. Check your dataset with notebooks under dataset_analysis. Use this notebook to find the right audio processing parameters. The best parameters are the ones with the best GL synthesis.(completed)
2. Write your own dataset formatter in datasets/preprocess.py or format your dataset as one of the supported datasets like LJSpeech.(completed)
    - preprocessor parses the metadata file and converts a list of training samples.
3. If you have a dataset with a different alphabet than English Latin, you need to add your alphabet in utils.text.symbols.(completed)
    - If you use phonemes for training and your language is supported here, you don't need to do that.
4. Write your own text cleaner in utils.text.cleaners. It is not always necessary to expect you have a different alphabet or language-specific requirements.
    - This step is used to expand numbers, abbreviations and normalizing the text.
5. Setup config.json for your dataset. Go over each parameter one by one and consider it regarding the commented explanation.(completed)
    - 'sample_rate', 'phoneme_language' (if phoneme enabled), 'output_path', 'datasets', 'text_cleaner' are the fields you need to edit in most of the cases.
6. Write down your test sentences in a txt file as a sentence per line and set it in config.json test_sentences_file.
7. Train your model.
    - SingleGPU training: python train.py --config_path config.json
    - MultiGPU training: CUDA_VISIBLE_DEVICES="0,1,2" python distribute.py --config_path config.json
        - This command uses all the GPUs given in CUDA_VISIBLE_DEVICES. If you don't specify, it uses all the GPUs available.

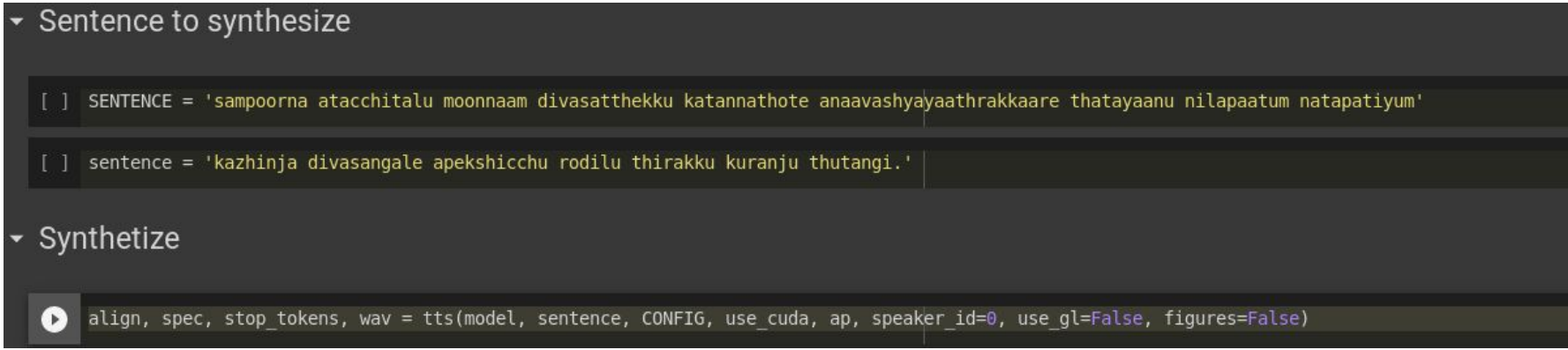
Project code can be found here:

https://github.com/kurianbenoy/MTTS

Experiment: Malayalam transliterated to TTS

I transliterated Malayalam text and obtained english like representation and passed it to English TTS and obtained sound samples.

The transliteration to Malayalam to English TTS was done using LibIndic library. First voice sample was generated



Experiment: Malayalam transliterated to TTS

Malayalam Transliteration

```
[ ] ! pip install  git+git://github.com/libindic/Transliteration
```

```
Collecting git+git://github.com/libindic/Transliteration
  Cloning git://github.com/libindic/Transliteration to /tmp/pip-req-build-itv7niyy
  Running command git clone -q git://github.com/libindic/Transliteration /tmp/pip-req-build-itv7niyy
Requirement already satisfied (use --upgrade to upgrade): libindic.transliteration==0.4.1 from git+git://github.com/libindic/Transliteration in /usr/
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from libindic.transliteration==0.4.1) (46.0.0)
Building wheels for collected packages: libindic.transliteration
  Building wheel for libindic.transliteration (setup.py) ... done
  Created wheel for libindic.transliteration: filename=libindic.transliteration-0.4.1-cp36-none-any.whl size=2566 sha256=5e73070cdaacf994eea28f9fae19
  Stored in directory: /tmp/pip-ephem-wheel-cache-p3h34fsn/wheels/54/bf/f4/8948d2bad9229ff9258fc143a647011c5dbef5b809769fdec9
Successfully built libindic.transliteration
```

```
[ ] ! git clone https://github.com/libindic/Transliteration.git
```

```
[ ] ! ls
```

```
[ ] %cd Transliteration
```

```
[ ] ! pip install -r requirements.txt
```

```
[ ] from libindic.transliteration import getInstance
    t = getInstance()
    text = u"നമസ്കാരം"
    t_text = t.transliterate(text, "ml_IN")
    print(t_text) #"നമസ്കാരം"
```

Using Espeak for TTS

The eSpeak NG is a compact open source software text-to-speech synthesizer for Linux, Windows, Android and other operating systems. It supports 107 languages and accents. It is based on the eSpeak engine created by Jonathan Duddington.

eSpeak NG uses a "formant synthesis" method. This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but is not as natural or smooth as larger synthesizers which are based on human speech recordings. It also supports Klatt formant synthesis, and the ability to use MBROLA voices.

eSpeak NG is available as:

- A command line program (Linux and Windows) to speak text from a file or from stdin.
- A shared library version for use by other programs. (On Windows this is a DLL).
- A SAPI5 version for Windows, so it can be used with screen-readers and other programs that support the Windows SAPI5 interface.
- eSpeak NG has been ported to other platforms, including Solaris and Mac OSX.

Sound samples generated can be found in github link

Thank you!