Components of End to End Khmer Speech Synthesis

Seanghay Yath

About me

A researcher on Khmer Natural Language Processing and End-to-end Khmer Speech Synthesis. Experienced in Android development, Node.js and some frontend frameworks. I worked for Pigeon Media, Pi Pay, Koh Santepheap Media, and MPTC.

—

Website: seanghay.com

GitHub: github.com/seanghay

Twitter: twitter.com/seanghay_yath

Hugging Face: hf.co/seanghay

Telegram: t.me/seanghay_yath

Overview

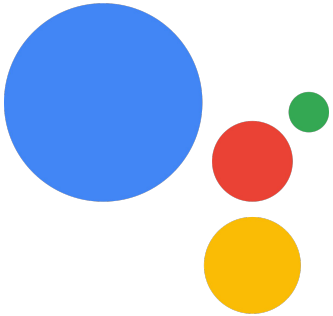
1. What is a Text to Speech System?
2. TTS Frontend
3. TTS Backend
4. Speech Dataset Preparation
5. Training & Inference
6. Deployment
7. Challenges
8. Wrap up

What is a Text to Speech System?

Text-to-speech (TTS) refers to the ability of computers to read text aloud. A TTS engine converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound.

https://en.wikipedia.org/wiki/Speech_synthesis

Well-known Text to Speech Systems

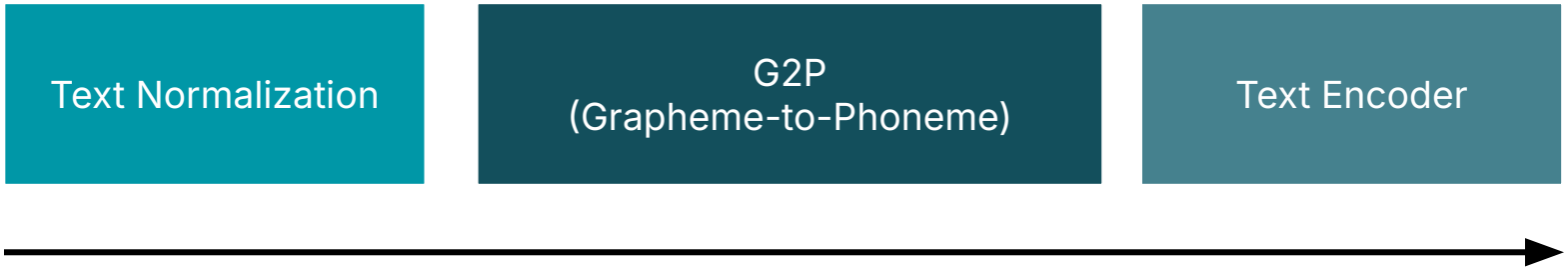


How Speech Synthesizer works

It takes raw text as an input and produce an audio speech.

TTS Frontend

Turns text into integer representations.



TTS Backend

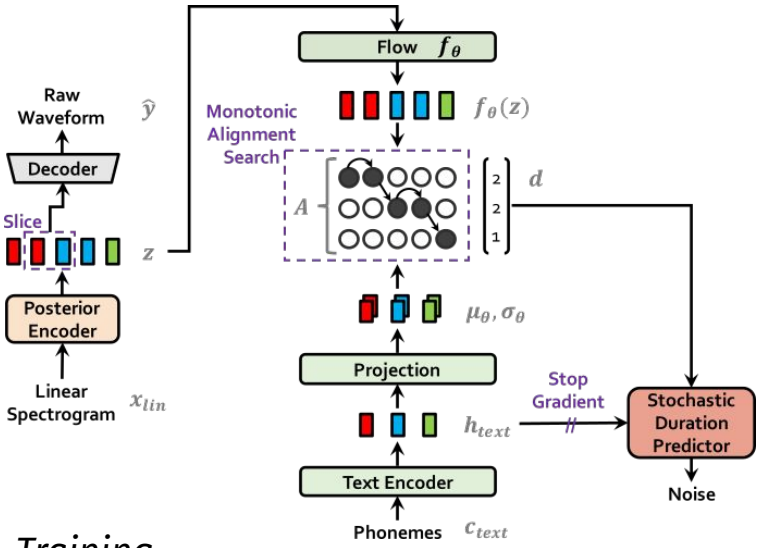
There are many backends such as **NaturalSpeech 2**, **Tacotron 2**, **WaveGlow**, **YourTTS**, **TorToiSe**, **FastSpeech 2** and many more but I chose **VITS** for this talk. Each backend has its own strengths and weaknesses so choose it based on your need and requirements.

—

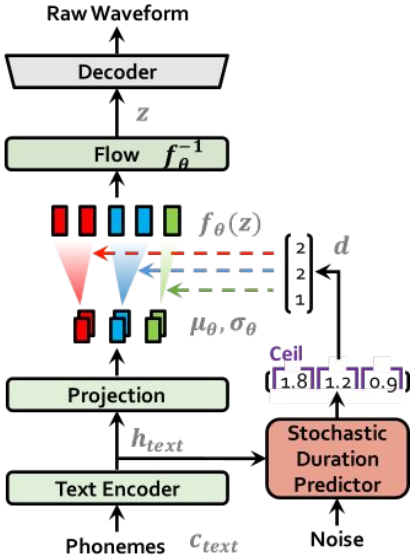
Learn more:
https://paperswithcode.com/sota/text-to-speech-synthesis-on-ljspeech

VITS: Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech

Jaehyeon Kim, Jungil Kong, and Juhee Son



Training



Inference

Get started

Text Normalization

Text Normalization / Text Verbalization

A text preprocessor that transforms things like numbers, abbreviations, symbols, date, time and ambiguities to a spoken text. The output must be all spoken forms.

Input

មានមនុស្ស២នាក់បានដួលលើផ្លូវជាតិលេខ6Aថែមទាំងស្រែកថាជួយផងៗ

Output

មានមនុស្សពីរនាក់បានដួលលើផ្លូវជាតិលេខប្រាំមួយអាថែមទាំងស្រែកថាជួយផងជួយផង

Text Normalization / Text Verbalization

1. Cardinal number
2. Decimal number
3. Currency / Money
4. Percentage
5. Khmer Repeater (ៗ) Expansion
6. Single Latin Character
7. Date and time
8. Phone numbers (010 123 123, 012 3333 12)
9. Plate numbers (1A-1234, 1A-4444)
10. Abbreviations (គជប, /ស.?ស.?យ.?ក/)
11. Address (6A -> ជ្រាំមួយ អា)
12. Fraction (1.5 តោន -> មួយ តោន កន្លះ)
13. Measurements (គ.ក -> គីឡូក្រាម, គ.ម -> គីឡូម៉ែត)

Text Normalization / Cardinal Number

- ១ -> មួយ
- ២២ -> ម្ភៃពីរ
- ១២០ -> មួយរយម្ភៃ
- ១,000,000 -> មួយលាន

Text Normalization / Decimal Number

១,២៣ មួយ ក្បាស ម្ភៃបី
១.២៣ មួយ ចុច ម្ភៃបី
-២.២៣ ដក ពីរ ចុច ម្ភៃបី
-២.៣២១១ ដក ពីរ ចុច សាមពីរ ដប់មួយ

Text Normalization / Currency

$100 -> មួយ រយ ដុល្លា
២០០៛ -> ពីរ រយ រៀល
USD ២០០ -> ពីរ រយ ដុល្លា

Text Normalization / Percentage

100% មួយ រយ ភាគ រយ
8.34% ប្រាំបី កណ្តាក សញ្ញា សាម សិប បួន ភាគ រយ

Text Normalization / Abbreviations

ព.ស. -> ពុទ្ធសករាជ
គ.ស. -> គ្រិស្តសករាជ
ប.ជ -> បន្ទាយមានជ័យ

គជប -> គ ជ ប
សសយក -> ស ស យ ក

Text Normalization / Date and Time

01/01/2023 -> ថ្ងៃទី មួយ ខែ មករា ឆ្នាំ ពីរពាន់ ម្ភៃ បី
01-01-2023 -> ថ្ងៃទី មួយ ខែ មករា ឆ្នាំ ពីរពាន់ ម្ភៃ បី
01/01/23 -> ថ្ងៃទី មួយ ខែ មករា ឆ្នាំ ពីរពាន់ ម្ភៃ បី
2023/01/01 -> ថ្ងៃទី មួយ ខែ មករា ឆ្នាំ ពីរពាន់ ម្ភៃ បី

9:30 AM -> ម៉ោង ប្រាំ បួន សាម សិប នាទី ព្រឹក
9:30 ម៉ោង -> ប្រាំ បួន កន្លះ
8:00 PM -> ម៉ោង ប្រាំ បី យប់

Text Normalization / Address

ផ្លូវជាតិលេខ ៦A -> ផ្លូវជាតិលេខ ប្រាំ មួយ អា
ផ្ទះលេខ 8Eo -> ផ្ទះលេខ ប្រាំបី អឺ សេរ៉ូ

Text Normalization / Phone Number

012 123 123 -> សូន្យ ដប់ ពីរ មួយ រយ ម្ភៃបី មួយ រយ ម្ភៃបី
012 32 31 33 -> សូន្យ ដប់ ពីរ សាម(សិប)មួយ សាម(សិប)ពី សាម(សិប)បី
012 3333 56 -> សូន្យ ដប់ ពីរ ការេ បី ហា(សិប)ប្រាំមួយ

Text Normalization / License Plate

1A-3324 មួយ អេ សាម (សិប) បី ម្ភៃ បួន

Text Normalization / Khmer Iteration Mark(ៗ)

មួយថ្ងៃៗ -> មួយ ថ្ងៃ មួយ ថ្ងៃ
មួយព្រឹកៗ -> មួយ ព្រឹក មួយ ព្រឹក
គាត់ស្រែថាចោរៗ -> គាត់ស្រែថា ចោរ ចោរ

Text Normalization / Tools

- Pynini
- OpenGrm Thrax
- Regex (find and replace)

G2P
(Grapheme-to-Phoneme)

What is a Grapheme-to-Phoneme?

Transforms text into more stable representations so we can feed them into a TTS backend so it can generalize it well.

ចាក់ -> c a k

ចក្រ -> c a k

អ្នក -> n ea k

នាក់ -> n ea k

ឲ្យ, អោយ -> ʔ ao j

How to Build a G2P

1. Original Text

គាត់បានធ្វើដំណើរទៅ ហានព្រះចន្ទវិញ

2. Word Tokenized

គាត់|បាន|ធ្វើដំណើរ|ទៅ| |ហាន|ព្រះចន្ទ|វិញ

3. Phonemized

k oa t|ɓ aa n|t w əə ɗ a m . n aə|t ɨ
w| |th aa n|p r ea h c a n|w ɨ ɲ

Khmer Word Boundary Detection

```
pip install khmercut
```

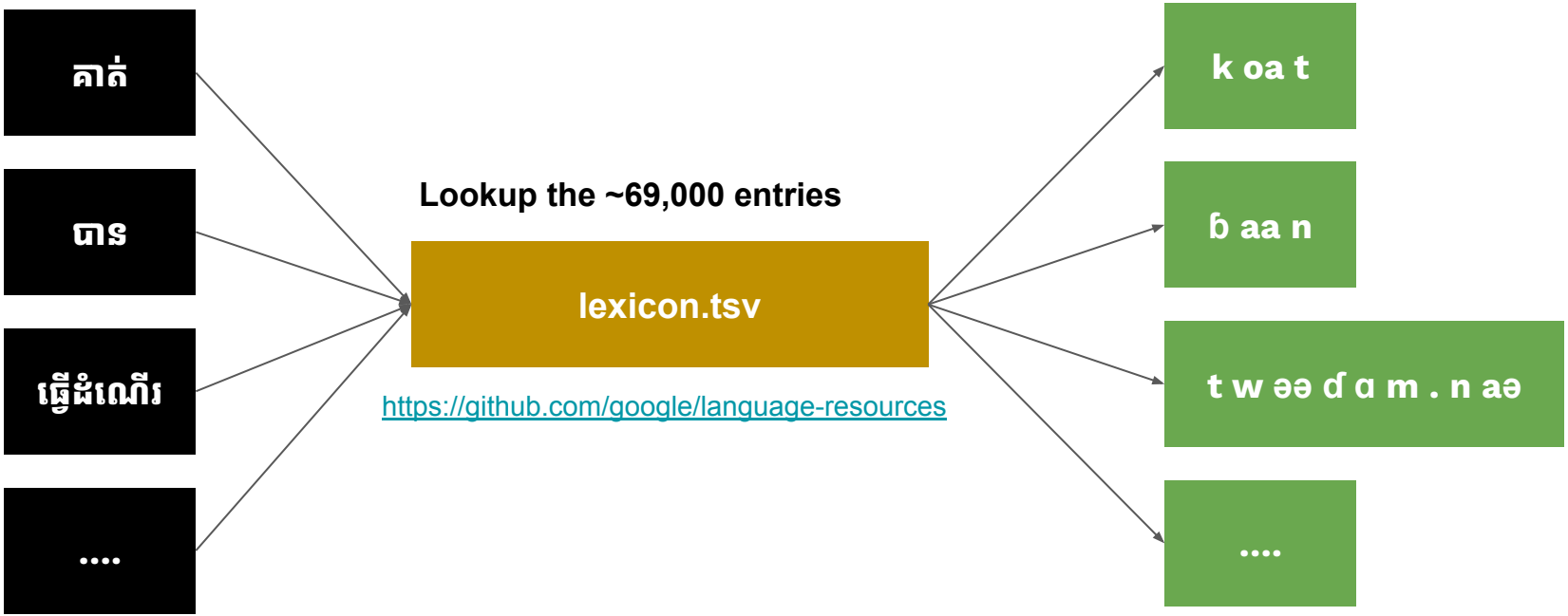
```
from khmercut import tokenize

tokens = tokenize("គាត់បានធ្វើដំណើរទៅ ហានព្រះចន្ទវិញ")

print(tokens)
```

```
["គាត់","បាន","ធ្វើដំណើរ","ទៅ"," ","ហាន","ព្រះចន្ទ","វិញ"]
```

Khmer Word Phonemizer



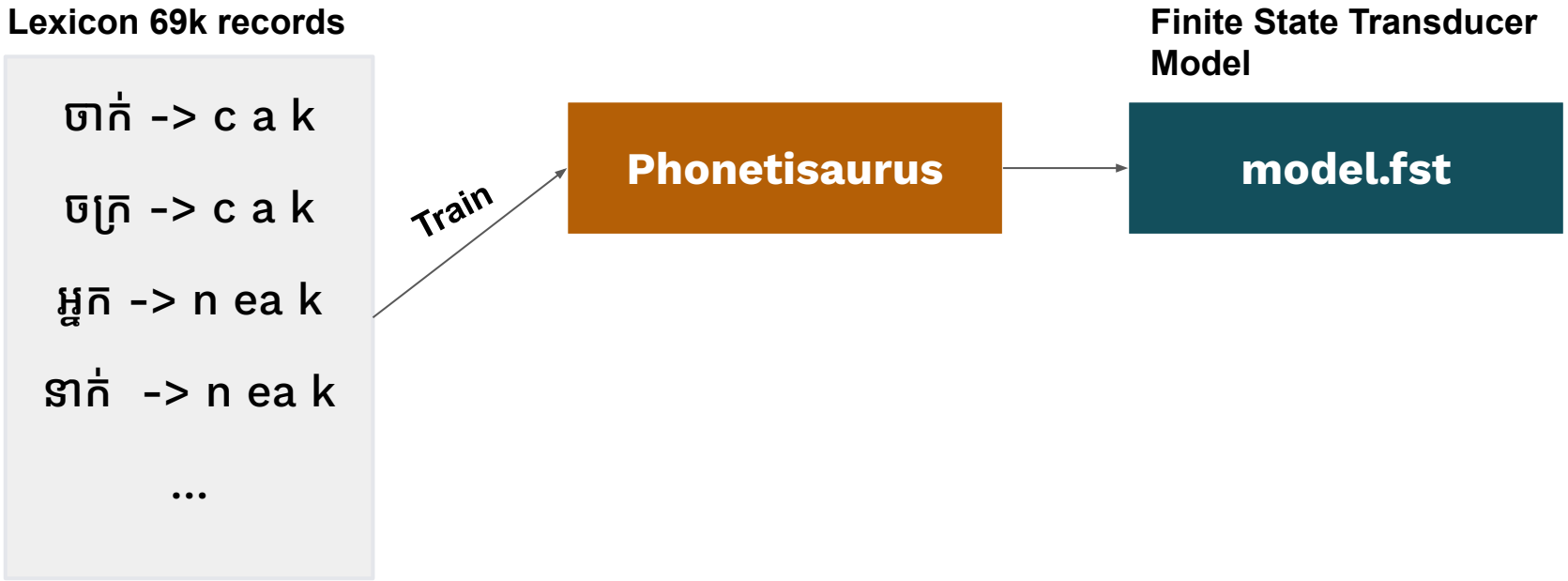
Out-of-Vocabulary (OOV) Handling

We have two options

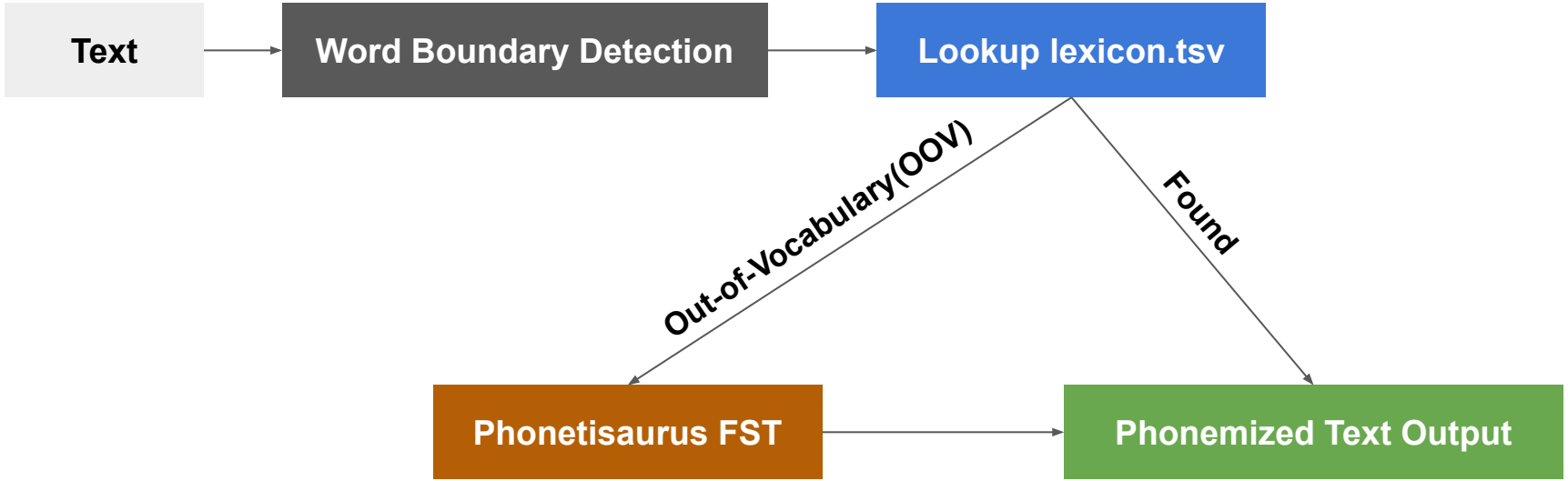
1. **Rule-based G2P (https://gitlab.com/mkrlab/automatic-phonemic-and-phonetic-transcription)**

2. **Train the lexicon with Phonetisaurus (https://github.com/AdolfVonKleist/Phonetisaurus)**

Train a G2P


Lexicon 69k records
ចាក់ -> c a k
ចក្រ -> c a k
អ្នក -> n ea k
ន្ទាក់ -> n ea k
...
Train
Phonetisaurus
Finite State Transducer Model
model.fst

Phonemizer Pipeline



Khmer Phonemizer

https://github.com/seanghay/khmerphonemizer

```
from khmerphonemizer import phonemize

text = "នៅលើលោកនេះមិនមានមនុស្សណាម្នាក់ចេះអស់ទេ"
result = phonemize(text)

print(result)
```

Text Encoder

Text Encoder

Convert Phonemes into an array of numbers since the backend only understand numbers.

1. Define the characters set.
2. Define a unique ID for each token and special tokens.

Text Encoder / Basic

Let’s say we have a map of { a=1, b=2, c=3, d=4, e=5, <UNK>=0 }

Input

```
text = “aacccbbbb!!”
tokens = [c for c in text]
# tokens = [ ‘a’, ‘a’, ‘c’, ‘c’, ‘c’, ‘b’, ‘b’, ‘b’, ‘b’, ‘!’, ‘!’]
```

Output

```
[ 1, 1, 3, 3, 3, 2, 2, 2, 2, 0, 0]
```

Text Encoder / Special Tokens

Reserved characters used internally for different purposes.

Common special tokens are [UNK],[PAD],[BLANK],[SEP], <pad>, <blank>, <sep>, <unk> etc.

Sometimes, special tokens are just a symbol such as [“_”, “\u2581”] etc.

Speech Dataset Preparation

Speech Audio File

Speech must be clear

Sample Rate → **22050 Hz**

File Format → **wav / flac**

Max Audio Duration → **30 seconds**

Each audio must have a denormalized transcription.

Total Duration → **20+ hours**

Forced Alignment

Find corresponding speech segment in audio to a given text.

Montreal Forced Aligner
https://montreal-forced-aligner.readthedocs.io/

Forced Alignment for Multilingual Data
https://pytorch.org/audio/master/tutorials/forced_alignment_for_multilingual_data_tutorial.html

Khmer Forced Aligner
https://github.com/seanghay/khmer-forced-aligner

Dataset Folder Structure

1. Place all audio files inside a folder called `wav`
2. Create 3 text files such as `val_filelist.txt`, `test_filelist.txt` and `train_filelist.txt` with the content like below

audio-01.wav|គាត់បាននិយាយថា គាត់នឹងទៅផ្ទះវិញឆាប់ៗនេះ។
audio-02.wav|ស្រាប់តែពេលនោះគាត់ក៏បានចុះទៅក្រោមផ្ទះ

Dataset Files Tree

```
my_dataset
├── test_filelists.txt
├── train_filelists.txt
├── val_filelists.txt
└── wav
    ├── audio-01.wav
    └── audio-02.wav
```

Training

Training / Hardware

Graphic Card

NVIDIA RTX 3090 24GB
NVIDIA A10 24GB ($0.60 / hr on Lambda Labs)

RAM

32 GB

Time

~1-2 days

Training / Code

The training code for Khmer language will be available at: https://github.com/seanghay/khmer-vits

It will be as simple as `python train.py`

TensorBoard

We monitor the training via TensorBoard, a web based visualization tool. Once the training is started you will be able to inspect the graph and the generated audio output for each evaluation at http://localhost:6006

After you satisfied with the generated audio output, you can stop the training and backup your checkpoints to use in the inference.

Inference

VITS inference can be run on CPU and CUDA fairly quickly.
To run the inference:

```
python inference.py input.txt \
    --checkpoint G_10000.pth \
    --output audio.wav
```

Inference with ONNX

VITS inference can be run on CPU and CUDA fairly quickly.
To run the inference:

```
python export_onnx.py --checkpoint G_10000.onnx

python inference_onnx.py input.txt \
    --checkpoint G_10000.onnx \
    --output audio.wav
```

Deployment

Remaining Challenges

1. The current Khmer Word Boundary tokenizer is not perfect.
2. Khmer Sentence Tokenizer
3. Punctuation Prediction
4. Prosody Prediction
5. Text Correction / Spell check
6. Text normalization
7. G2P still needs more work for complex words

Wrap up

References

awesome-khmer-language
https://github.com/seanghay/awesome-khmer-language
Phonetisaurus
https://github.com/AdolfVonKleist/Phonetisaurus
khmernormalizer
https://github.com/seanghay/khmernormalizer
khmercut
https://github.com/seanghay/khmercut
khmercut-rs
https://github.com/seanghay/khmercut-rs
phonetisaurus-js
https://github.com/seanghay/phonetisaurus-js
automatic-phonemic-and-phonetic-transcription
https://github.com/seanghay/automatic-phonemic-and-phonetic-transcription

Thanks to

H.E Chanty Sothy
Mr. Rina Buoy
Mr. Vitou Phy
Mr. Makara Sok
Mr. Sovichet Tep
Mr. Phan Viet Hoang
Mr. Kyle Gorman, Google
Mr. Richard Sproat, Google
Meta FAIR

Python Cambodia

Thank you!
Stay curious.

Seanghay Yath