Development of a Kyrgyz Language Speech Synthesizer:

A Demonstration of the Ossian Frontend and the Merlin Neural Network Speech Synthesis System

Joshua Meyer
Department of Linguistics
University of Arizona
Tucson, Arizona, 85721
jrmeyer.github.io

***Abstract*—This paper describes the rapid and efficient development of a Kyrgyz language speech synthesizer using open source technologies created by The Centre for Speech Technology Research at the University of Edinburgh. The synthesizer was trained on recordings from an open license audio book recorded by the Kyrgyz Language Audiobooks initiative from the Bizdin Muras group.**

I. Introduction

The creation of new Text-To-Speech (TTS) technologies typically require either (1) expert, linguistic knowledge of the language or (2) a sizeable collection of audio with transcriptions. Open source software exists for either of these two approaches, for example eSpeak NG allows linguists with some programming knowledge to create a synthesizer by coding orthography –¿ pronunciation rules. Practically speaking, the linguist need not be an expert programmer to create a useable voice quickly with eSpeak NG. However, a major drawback of such rule-based speech synthesizers is that they inevitably sound robotic. For certain applications this is not an issue, and even benefitial (screen readers like NVDA can more easily speed up this kind of speech). However, most people prefer human-like voices.

Unfortunately, there is currently no way to create a human-like, natural voice without first collecting a corpus of speech. The second approach is exactly this: collect speech and train a statistical model to replicate that voice. While this approach sounds best, it typically requires access to an audiobook as well as more expert programming knowledge. The current paper demonstrates the development of a Kyrgyz lagnuage synthesizer using Ossian and Merlin, which are designed to be user friendly and allow the non-expert to create a good voice quickly.

Furthermore, I demonstrate how it is possible to create accurate alignments at the utterance level for an entire audiobook, by training a speech recognizer on only a small subsection of that same book. Typically audiobooks come in audiofiles which are too large to use directly for training a synthesizer. By splitting large files on silence and then generating transcriptions, we can use more audio with less time hand-aligning segments.

II. Data

The audio book used to train this system was spoken by a female native Kyrgyz speaker from the north of the country. The audio files made available to the author were formatted as `stereo MPEG 1.0 Layer III` with a sampling rate of `44 kHz` and a `160 kbps` bit rate. The recordings were available in `30 minute` segments.

The original recordings were split on silence longer than `.55 seconds`, and the resulting utterance-sized files were then hand-aligned to text by the author. By using .55 seconds of silence as the delimiter, the audio was split on what mostly corresponded to complete sentences in the text file. After splitting the audio on silence, the first `1.5 hours` were hand-aligned to text by the author. This 1.5 hours was used to train the speech recognizer described next.

III. Speech Recognizer: Kaldi

A. Training

The original source was down-sampled to 16kHz and the two original channels were merged (to convert stereo to mono). All acoustic models were generated and trained using the Kaldi speech recognition toolkit. The features used were 13-dimensional MFCCs. Delta and Delta Delta features were also extracted, for a total of 13 MFCCs + 13 Δ + 13 $\Delta\Delta$ = 39 features. Cepstral mean and variance normalization was then applied as normalization.

Initially, monophones were trained from a flat start. Then, context dependent triphones were trained using the alignments given by the monophones. The context-dependent triphones were then used to generate alignments for the audio, and those alignments were used to train the final DNN. Since we're working with a single speaker, there was no speaker adaptation applied.

The language model incorporated into the decoding graph was a tri-gram backoff model trained on the text of the audiobook itself. This means the decoder was highly biased to predict strings from the audiobook, which is good for our application.

B. Decoding

An additional `10 hours` of the audio book (which had been split on silence) was then decoded with the DNN ASR

system trained on the first 1.5 hours of the audiobook. The generated transcriptions for these 10 hours were treated as correct and used in the training of the speech synthesizer.

IV. TTS Frontend: Ossian

Training in Ossian was performed with the standard `naive_01_nn` recipe.

V. TTS Backend: Merlin

Merlin

VI. Conclusion

Acknowledgment

I would like to thank the researchers at CSTR for welcoming me to Edinburgh and helping me understand their fabulous toolkits; in particular Oliver Watts, Simon King, and Srikanth Ronanki.

The author was in part supported by a Graduate Research Fellowship from the US National Science Foundation during this research.

References

[1] bib goes here