Deep Neural Net Speech Synthesis for the Chuvash Language

Dante Razo
Indiana University, Bloomington, IN
Department of Linguistics
drazo@indiana.edu

Abstract

Chuvash is a minority language spoken by roughly one million people in European Russia (RBS, 2012). It is a Turkic language that utilizes the Cyrillic alphabet. For this project, I trained an Ossian/Merlin speech synthesis model on Chuvash-language news clips. The former is a front-end for Merlin, which is a neural net based speech synthesis system. The performance of Ossian was to be compared to other popular solutions such as Mozilla TTS, Mozilla LPCNet, Festival, and eSpeakNG. Due to the time constraint, I was only able to completely train the Ossian/Merlin model.

1 Introduction

Despite the relatively large number of speakers, Chuvash is still considered a minority language. This project aimed to train popular speech synthesizers to produce intelligible Chuvash from written samples. The sparsity of data made this more of a challenge than if I had used another, more popular language.

Text-to-speech works by accepting text, conducting linguistic analysis on the input, then producing audio waveforms. Data preprocessing techniques include normalization and tokenization (Wikipedia, 2019a), though the latter is meant for text corpora used as input. Audio could be "preprocessed" by normalizing volume and adding silence to the beginning of files. While this seems counterproductive, establishing a noise floor to helps the system filter out background sounds like fans or other peoples' conversations.

Most of the systems featured in this project are trained on audio samples. I used clips from Chuvash-language news programs as my primary corpus. Due to the small number of samples ($\sim$ 546) for the task at hand, *transfer learning* should be used to get the best results. Unfortunately, given the time restraint, I was unable to implement this technique with Ossian or the other systems. Transfer learning would entail training a **Deep Neural Net** (DNN) on a language with copious amounts of data (like English, for example), then carrying over neuron weights to the Chuvash DNN. Despite phonological and syntactical differences, there are enough similarities between the languages (or any two for that matter) for transfer learning to be effective.

Speech synthesis is the production of artificial human speech (Wikipedia, 2019b). Text-to-speech (TTS) is a subset of the field which focuses on taking text as input and returning synthesized audio "speech" as output. There are multiple ways to do TTS, but this project will focus on neural networks and one formant synthesis solution. The best neural nets on the market sound like real humans, but they can still be distinguished by their staggered manner of speaking. Examples can be found on Bloomington's public transportation when arriving at stops or in your phone via voice assistants. A few famous examples of TTS are Microsoft Sam, a very simple program included in early versions of Windows, NOAA Severe Weather Alerts, and the late Stephen Hawking's voice synthesizer.

2 Text-to-Speech Systems

Neural net systems are among the best in speech synthesis, but are they necessary to achieve fluid, intelligible speech? An overview of the models used in this project can be seen in Table 1.

The criteria used for ranking the systems will be described in further detail later in this report. The two main factors I used when comparing them are said to be a speech synthesis system's most impor-

System	Basis	Rank
Ossian & Merlin	DNN	*1*
Mozilla LPCNet	RNN	*3*
eSpeakNG	Formants	*2*

Table 1: Speech synthesis systems.

tant: *naturalness* and *intelligibility* (Taylor, 2009).

2.1 Ossian & Merlin

Ossian is a front-end for speech synthesis development by the Centre for Speech Technology Research (CSTR) at The University of Edinburgh. It serves as the text processor and front-end for the Merlin library, which is a neural-net based speech synthesis system developed by the same team. Merlin uses a **DNN** for training acoustic models and could be considered the "brains" of the dichotomy. The libraries make use of both `.txt` and `.wav` files for training. (CSTR-Edinburgh, 2018).

2.2 eSpeakNG

eSpeakNG (Next Generation) is the only non-neural-net system I used for this project. It derives the "NG" moniker from its status as a fork of the original eSpeak project by Johnathan Duddington. It takes a unique approach to speech synthesis by instead using *formant synthesis* (Duddington, 2013) on written rules for the language. Once you provide morphological and phonological rules, it can piece individual sounds together to form a cohesive audio sample. I referred to documentation for both eSpeak and eSpeakNG when conducting this project, but I chose to use the latter because it is more up-to-date than its predecessor.

I tested eSpeakNG with short English and Spanish phrases and achieved impressive results. Both samples were intelligible, and in the case of Spanish, especially fluid.

2.3 Mozilla LPCNet

Mozilla's LPCNet uses a recurrent neural net (**RNN**) for training and aims to make speech synthesis painless on regular consumer computers (Mozilla, 2019). Though a high-end GPU is recommended, training can be done on a midrange CPU. This system requires special 16-bit, 16 kHz PCM machine-endian audio files for training. I was unable to automate the conversion process and had to abandon LPCNet in the interest of time.

2.4 Data & Preprocessing

I used Francis Tyers' `Turkic_TTS` repository of Chuvash-language news clips to train my model. Located in the `./corpus/chv/speakers/chuvash_news/` folder is a README detailing how to extract training data from the repository. This grew the size of my project folder considerably, but it gave me usable data. No further preprocessing was required for Ossian.

LPCNet requires a very specific encoding for audio input. Preprocessing was necessary but not feasible given my unfamiliarity with endianness as it pertains to audio. Attempting to convert audio by hand resulted in painfully loud audio samples that resembled screeching. I believe this has more to do with endinanness than the other requirements (frequency, bits, and waveform).

2.4.1 Corpora Repositories

I made use of the following repositories for training and testing models:

1. Turkic_TTS by Francis M. Tyers (*ftyers*)
2. apertium-chv (GPL-3.0) by *Apertium*

3 Configuration & Training

All training was done in an Ubuntu 18.04 virtual machine with 8 cores and 8GB of RAM. Due to the virtualization, the OS was unable to access my graphics card for quick training. The first system I tried to train, Mozilla TTS, was dependent on a GPU. LPCNet, also written by Mozilla, requires the same. My virtual machine setup was a limiting factor in which systems I was ultimately able to pursue.

Josh Meyer's TTS Workshop guide (Meyer, 2016b) proved invaluable in setting up Ossian & Merlin. It listed dependencies, Hidden Markov Model Toolkit (HTK, a dependency) installation instructions, and more. The biggest factors in the success of this project were Meyer's guide and the `Turkic_TTS` repository of well-organized audio data.

3.1 Training Ossian

Training was, as expected, a lengthy process. It utilized all of my CPU power and rendered my computer — both the host and the virtual Linux machine by extension — near-useless on occasion. Training lasted roughly 4 hours. The Ossian model was trained on April 9, 2019.

Filepath	Action
./Makefile.am	*edit*
./phsource/phonemes	*edit*
./phsource/ph_chuvash	*create*
./dictsource/cv_extra	*create*
./dictsource/cv_list	*create*
./dictsource/cv_rules	*create*
./espeak-data/voices/family/trk	*create*

Table 2: eSpeakNG Modifications. (Meyer, 2016a)

3.2 Configuring eSpeakNG

The eSpeakNG library does not require training in the same sense as other neural-net-based systems. Instead, the target language's phonological rules are written in text for the library to "learn".

Initially, I installed the version of eSpeakNG from the Ubuntu package manager to get a feel for it and test it out. This can be done with the following code:

```
> sudo apt-get install espeak-ng
```

In order to use custom languages, one must clone the eSpeakNG repository, build it using the provided scripts, '`make`' it, then install. First, I installed dependencies (a few of which were already present on my machine):

```
> apt-get install make autoconf \
automake libtool pkg-config
```

Once the dependencies were installed and/or updated, I cloned Harry Zhang's fork (Zhang, 2019) of the official `espeak-ng` repository. The following code clones the repo, enters the new directory, then tells Git to focus on the desired repository branch.

```
> git clone https://github.com/ \
Contextualist/espeak-ng \
espeak-ng-chv
> cd espeak-ng-chv
> git checkout cv
```

Finally, use the following code to generate the files needed to build eSpeakNG, build it, then install it to the `/usr` directory.

```
> ./autogen.sh
> ./configure --prefix=/usr
> make
> sudo make install
```

Phn	Manner & Place of Articulation	IPA
@	Close-mid central unrounded vowel	[ɘ]
r	Alveolar trill	[r]
S	Voiceless alveolopalatal fricative	[ɕ]
Z	Voiced alveolopalatal fricative	[ʑ]
tS	Voiceless alveolopalatal affricate	[tɕ]
dZ	Voiced alveolopalatal affricate	[dʑ]

Table 3: Sample of Chuvash phonemes.

3.3 Adding Chuvash to eSpeakNG

Josh Meyers' guide (Meyer, 2016a) outlined how to add any language to eSpeakNG. I followed the first few steps before electing to use Harry's library with the necessary files already added in the interest of time. I double-checked Harry's work against the guide, and it appears that he followed it faithfully. The changes necessary to implement Chuvash are outlined in Table 2.

3.3.1 Voice File

The voice file is a simple declaration of the name of the language you wish to add, and a two-letter code for it.

The file must be placed in the correct directory for eSpeakNG to recognize it. Since Chuvash is part of the Turkic family of languages, I put the voice file in the `./espeak-ng-data/voices/trk`

The file must be placed in the correct directory. Since Chuvash is part of the Turkic family of languages, its voice file should go in the `./espeak-ng-data/voices/trk` directory. The following code was used to navigate here, create the `trk` folder, enter it, then create a file named `cv`:

```
> cd espeak-ng-data/voices
> mkdir trk
> cd trk
> touch cv
```

I used **nano**, my favorite command-line text editor, to edit and save `cv`. The contents of the file are transcribed below using the `cat` command.

```
> cat cv

  name chuvash
  language cv
```

3.3.2 Phoneme Definition File

This file is used to define the sounds of the target language. The following code was used to create the *phoneme definition file* for Chuvash.

```
> cd ./phsource/
> touch ph_chuvash
```

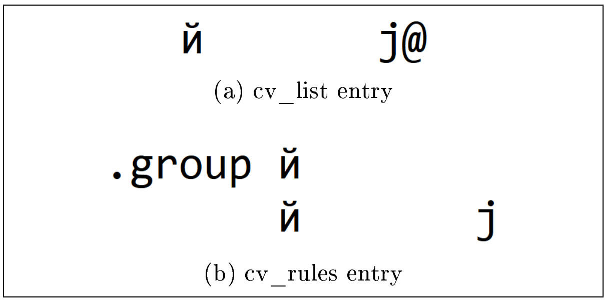
Due to the time constraint, only the vowels in Table 3 were implemented. Figure 1 below is the entry for the *voiced alveolopalatal fricative*.

```
phoneme Z.
vcd alp sib frc
//voicingswitch S.
ipa [ʑ]
endphoneme
```

Figure 1: Voiced alveolopalatal fricative.

3.3.3 Dictionary Files

In Table 2, there were three files created in the `./dictsource` directory. These are the *dictionary files*. Their purpose is to match text to sounds for eSpeakNG. Figures 2a and 2b were taken from `cv_list` and `cv_rules` respectively and represent the letter й.



```
й          j@
```

(a) cv_list entry

```
.group й
       й        j
```

(b) cv_rules entry

Figure 2: Chuvash dictionary entries for й

3.3.4 Master Phoneme File

Next, the `phonemes` file located in `./phsource` needed to be edited to include references to the new Chuvash files. The following two lines were added to the final section of the file titled "*ADDITIONAL PHONEME TABLES*":

```
phonemetable cv base2
include ph_chuvash
```

3.3.5 Makefile

In order to compile new eSpeakNG languages when running `make`, the `Makefile.am` must be modified to point towards the *master phoneme file* created in the previous step. The following line was added under the *dictionaries* header to point `make` to the new Chuvash dictionary file:

```
dictionaries: \
    ...
    espeak-ng-data/cv_dict \
    ...
```

Below the *dictionaries* section but not within it, the following references to `cv_dict`, `cv_list`, `cv_rules` were added:

```
cv: espeak-ng-data/cv_dict
espeak-ng-data/cv_dict:
    dictsource/cv_list dictsource
    /cv_rules
```

3.3.6 Compiling Changes

The following one-liner will `make` and update your installation of eSpeakNG. I ran it every time I made a change to the library's files.

```
> make && sudo make install
```

4 Results & Evaluation

4.1 Scoring System

In the request-for-proposal (RFP), I outlined an objective scoring system that I could use despite not being fluent in Chuvash. Due to the small set of systems I used, this system is no longer necessary. I have ranked the systems below:

1. Ossian & Merlin
2. eSpeakNG
3. Mozilla LPCNet

The rankings for each system are explained below in their respective section.

4.2 Ossian & Merlin

Ossian was the only system that I was able to get working with Chuvash. The audio sounds good, and will likely be intelligible to speakers of the language. It received the first-place distinction mostly due to the fact that it worked and produced results for Chuvash.

4.3 eSpeakNG

eSpeakNG, when installed from the Ubuntu package manager, worked with all preconfigured languages. I tested *English* and *Spanish* primarily and got good results. eSpeakNG placed second because it works with other languages but not Chuvash (yet).

4.4 Mozilla LPCNet

Mozilla's **LPCNet** showed promise, but it gets third place because I was unable to configure and train it before the deadline. I believe it would still get third place even if it had worked. The demo audio clips from the documentation (Valin, 2018) are intelligible (for English, at least), but they have a lower bitrate than eSpeakNG's synthesized samples.

4.5 Additional Systems

I previously attempted to configure Mozilla's "**TTS**" DNN-based system, but I was unable to set it up as well. It required **Docker**, a virtualization software that I was unfamiliar with. Using Docker renders my VirtualBox Ubuntu installation useless because they can't work concurrently. Without Linux, the rest of the project would be impossible, so Mozilla TTS was not an option.

5 Future

I originally planned on training more than one neural-net based speech synthesis system as well as eSpeakNG. It was an ambitious goal, but time was the biggest limiting factor. Having more than one system would give me multiple points of comparison (one DNN-based, one RNN-based, and one that utilizes *formant synthesis*).

There are some data preprocessing techniques I wanted to try in an effort to get better results with Ossian. Ideas include removing silence from the **end** of audio files since the noise floor would have already been established by then, and regulating volume throughout the entire dataset with ReplayGain. Transfer learning, though not necessarily a data preprocessing technique, is guaranteed to improve model performance. Implementing it would be a whole project in itself, however.

If I could get in contact with a native speaker of Chuvash, I would have them score audio samples using the objective scoring system developed for my RFP.

Finally, if I made any breakthroughs in implementing Chuvash into eSpeakNG, I would make a pull request for the developers to integrate my code into the repository's main branch. One of my personal goals is to contribute to an open-source software project in the near future.

Acknowledgments

I'd like to acknowledge Francis M. Tyers' work on the `Turkic_TTS` corpus. Compiling the data was easy using the provided scripts and instructions. I'd like to thank Professor Tyers again for his website, reading, and tutorial recommendations which improved my understanding of the Chuvash language and the tools used in this project. Thanks to the original authors of the ACL 2018 format as well for this template. Thanks to Josh Meyer for his extremely helpful **Ossian** and **eSpeakNG** tutorials. Finally, thanks to Harry Zhang for his advice on audio preprocessing and preliminary work on implementing Chuvash into the eSpeakNG library.

License

References

CSTR-Edinburgh. 2018. Github repository: Ossian.

Jonathan Duddington. 2013. espeak text to speech.

Josh Meyer. 2016a. How to add a language to espeak ng.

Josh Meyer. 2016b. Let's make a chuvash voice!

Mozilla. 2019. Github repository: lpcnet.

Russian Bureau of Statistics RBS. 2012. Population of the russian federation by languages (in russian).

Paul Taylor. 2009. *Text-to-Speech Synthesis*. Cambridge University Press.

Jean-Marc Valin. 2018. Lpcnet: Dsp-boosted neural speech synthesis.

Wikipedia. 2019a. Lexical analysis.

Wikipedia. 2019b. Speech synthesis.

Harry Zhang. 2019. Github repository: espeakng.