LING-L 445 Course Project RFP 2e

Dante Razo[†]

Abstract
Chuvash is a minority language spoken by roughly one million people in European Russia.[1] This project aims to train speech synthesizers with neural networks to reproduce intelligible Chuvash from text. The synthesizers used are Ossian, Mozilla TTS, and Mozilla LPCNet. At the minimum, the performance of the three will be compared. The Expected Product involves cleaning the Chuvash data to produce better results with one of the three synthesizers. If time permits, the best-performing synthesizer will be tweaked to produce even more accurate results. Data will come from reliable sources such as the Apertium project. The end goal is to train a neural network that mimics Chuvash speech to some degree, then improve on it as much as time allows.

Keywords
chuvash — deep learning — linguistics — speech synthesis — text-to-speech — transfer learning

[†] *Computer Science; School of Informatics, Computing and Engineering, Indiana University, Bloomington, IN, USA*

Contents

1. Introduction

The Chuvash language is spoken by roughly one million people in European Russia.[1] Despite the large number of speakers, it is considered a minority language. This project aims to train popular speech synthesizers to produce intelligible Chuvash from written samples.

Speech synthesis is the production of artificial human speech.[2] Text-to-speech (TTS) is a subset of the field which focuses on taking text as input, and returning audio "speech" as output. There are multiple ways to do TTS, but this project will focus on Deep Neural Networks (DNNs). The best DNNs on the market sound like real humans, but they can still be distinguished by their staggered manner of speaking.

Text-to-speech works by accepting text, conducting linguistic analysis on the input, then producing audio waveforms. Data preprocessing techniques include normalization and tokenization,[3] though the latter is meant for text corpora used as input. Audio could be "preprocessed" by normalizing volume and editing silence out from samples.

The DNNs used in this project will be trained on audio samples mainly taken from Chuvash-language news programs. Due to the small number of samples ($\sim$ 546) for the task at hand, I'll likely need to do transfer learning to get the best results. This technique is explained in further detail in §2.3.

2. Proposed Goals

2.1 Minimum Viable Product (MVP)

My MVP is to compare the performance of existing speech synthesizers with the Chuvash language. I plan on using the following solutions:

1. Ossian
2. Mozilla TTS
3. Mozilla LPCNet

The Ossian library will use deep neural networks (DNNs) trained by the Merlin toolkit as models. Mozilla TTS is a deep learning speech synthesis engine that accepts preprocessed datasets.[4] Mozilla's LPCNet requires at least an hour of speech data,[5] so I'll be using Francis M. Tyers' `Turkic_TTS`

repository to train it. It remains to be seen whether this will be enough data to properly train LPCNet; transfer learning may be required (see §2.3).

In order to understand the subject matter, I will work on memorizing the Cyrillic script and Chuvash's additional four letters. This will help me sound out text and isolated symbols (disregarding irregular phonological features such as palatalization). Almost a third of the characters used in Chuvash are reserved for Russian loanwords,[6] but they are worth learning in case they appear in the data.

2.2 Expected Product (EP)

I will explore new methods of preprocessing the data in hopes of improving model accuracy. Due to the fact that data has to be processed differently to work for each synthesizer, I will likely choose one to focus on for the duration of the project. Ideas include changing audio bitrate, removing unintelligible/messy audio samples, ReplayGain volume normalization, removing silence from audio samples, and k-folds cross-validation (separating the data into separate groups for testing and training).

2.3 High-achievement Product (HAP)

Comparing the different speech synthesizers will give me insight into their strengths and weaknesses. With this knowledge, I will tweak them (or my preferred synthesizer) in an effort to improve their effectiveness. If I see significant improvement in accuracy, I will consider reaching out to the original developers and discussing how to incorporate my changes with their source code.

If there is ample time to do so, I'll use transfer learning to make up for the lack of high-quality labeled Chuvash audio data. This would entail training a DNN on the English language (for which audio samples are plentiful) and applying insights (i.e. neuron weights) to a Chuvash-trained DNN.[7] As a native speaker, I'll be able to evaluate English audio output by myself. The problem is the same, just with different languages, so speech synthesis for English can still be used to enhance a Chuvash DNN.

2.4 Final Evaluation

For evaluation, I will try to get in touch with Chuvash speakers and have them give feedback on the audio output. I plan on using social media to gather some volunteers. This goal is contingent on whether I can find people fluent in both English and Chuvash who are willing to donate some of their time and stay in contact.

In the event that I can't find bilingual speakers who are willing to help, I'll use k-folds cross validation to hold out a subset of the data. I can compare this collection of natural Chuvash with synthesized samples to determine a speech synthesis system's most important attributes: naturalness and intelligibility.[8] I've devised a tentative scoring system (subject to future improvements) to keep evaluations subjective and allow for comparisons between the models:

1. The audio can be transcribed to some degree (resembles speech) [+3 points]
2. Quality of output:
   (a) The transcription matches the input text exactly [+5]
   (b) The transcription's consonants match the input text, but some vowels differ [+4]
   (c) The transcription has some overlap with the input text, but a few vowels and consonants differ [+3]
   (d) The audio can be transcribed (meaning it produces intelligible phonemes), but it doesn't match the transcription [+1]
3. Intonation sounds natural [+1]
4. Cadence sounds natural [+1]

This system rewards successfully producing natural-sounding waveforms, even if they aren't what we want. A score ≥ 10 indicates an intelligible model. Models that score above 8 are considered satisfactory, but may sound "cold" and "robotic". Intonation and cadence will be subjective to non-native speakers of the language, hence their low weight on the final score. If I can find Chuvash speakers, I will ask them to rate intonation and cadence on a scale of 1-6 for a total of 20 points that could be earned. I choose 6 so they can't pick 3 as a neutral/indifferent response; instead, every answer will be weighted towards either bad (1-3) or good (4-6).

3. Requirements

3.1 MVP Goals

3.1.1 Subgoal 1

Configure all three speech synthesizers as described in their tutorials, and confirm that they work.

3.1.2 Subgoal 2

Find suitable Chuvash data for training and process it for use with all synthesizers

3.1.3 Subgoal 3

Train synthesizers on Chuvash data and report results. Evaluate the models with single words.

3.2 EP Goals

3.2.1 Subgoal 1

Employ different preprocessing techniques until model accuracy is significantly higher than it was before.

3.2.2 Subgoal 2

Use tokenization and/or segmentation on a Chuvash text corpus. Evaluate the models with whole sentences.

3.2.3 Subgoal 3

Get in contact with bilingual Chuvash and English speakers for model output evaluation.

3.3 HAP Goals

3.3.1 Subgoal 1

Understand the inner workings and parameters of the chosen speech synthesizer.

3.3.2 Subgoal 2

Tweak the synthesizer to achieve higher accuracy with Chuvash data.

3.3.3 Subgoal 3 (Stretch Goal)

Coordinate with developers to incorporate my changes into the official release/repository of the synthesizers.

4. Timelines

4.1 MVP Timeline

1. Subgoal 1: *March 12, 2019*
2. Subgoal 2: *March 15, 2019*
3. Subgoal 3: *March 17, 2019*

4.2 EP Timeline

1. Subgoal 1: *March 24, 2019*
2. Subgoal 2: *March 31, 2019*

4.3 HAP Timeline

1. Subgoal 1: *April 7, 2019*
2. Subgoal 2: *April 13, 2019*
3. Subgoal 3: *N/A (Stretch Goal)*

4.4 Project Timeline

1. Paper: by *April 12, 2019*
2. Project: by *April 15, 2019*

5. Data Policy

Despite the scarcity of readily-available Chuvash datasets, I aim to only use reliable data. Currently, I'm looking to use Francis Tyers' `Turkic_TTS` repository for training. I will study Tyers' `apertium-chv` transducer as well for insights into Chuvash's lexical features. This will be useful for quickly evaluating the quality/fidelity of text samples and voice output.

The synthesizers I'll be using can be found on their respective GitHub repositories (listed below). I will follow typical software development procedures such as making regular commits, and committing every change to make backtracking easy. I plan to utilize branches on Git to allow for backtracking if needed.

My final report will include easily-reproducible instructions so that other scientists will be able to achieve the same (or similar) results. In the event that my adjustments to any synthesizer improves performance, I will get in touch with the original developer to discuss how to merge our code. This might be as simple as making a pull request.

My work will use a GPL-3.0 license (subject to change).

5.1 Data Sources

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

5.2 Repositories

1. Mozilla TTS (MPL-2.0) by Mozilla
2. Ossian (Apache-2.0) by CSTR-Edinburgh
3. Mozilla LPCNet (BSD-3-Clause) by Mozilla

References

[1] Russian Bureau of Statistics: Владение Языками Населением Российской Федерации. (Russian)
[*Population of the Russian Federation by Languages*]
http://www.gks.ru/free_doc/new_site/perepis2010/croc/Documents/Vol4/pub-04-05.pdf

[2] Wikipedia: Speech Synthesis
https://en.wikipedia.org/wiki/Speech_synthesis

[3] Wikipedia: Lexical Analysis
https://en.wikipedia.org/wiki/Lexical_analysis

[4] Mozilla: TTS README.md
https://github.com/mozilla/TTS

[5] Mozilla: LPCNet README.md
https://github.com/mozilla/LPCNet

[6] Wikipedia: Chuvash Language
https://en.wikipedia.org/wiki/Chuvash_language

[7] Niklas Donges
Towards Data Science: Transfer Learning
https://towardsdatascience.com/transfer-learning-946518f95666

[8] Paul Taylor
Text-to-Speech Synthesis.
Cambridge University Press, Cambridge, United Kingdom, 2009.

[9] Josh Meyer
Create New Voice with Ossian & Merlin
http://jrmeyer.github.io/tts/2017/09/15/Ossian-Merlin-demo.html

[10] Josh Meyer
Let's make a Chuvash voice!
http://jrmeyer.github.io/tts/2016/12/09/tts-workshop.html

[11] Jean-Marc Valin
Mozilla LPCNet Demo
https://people.xiph.org/ jm/demo/lpcnet/