LING-L 445 Course Project RFP

Dante Razo[1]

Abstract
Chuvash is a minority language spoken by roughly one million people in European Russia. 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 neural networks — linguistics — speech synthesis — tts

[1] *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. 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.

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. Mozilla's LPCNet requires at least an hour of speech data; 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.

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, 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 synthesizer 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.

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.

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; report results.

3.2 EP Goals

3.2.1 Subgoal 1

Clean and preprocess Chuvash data for cleaner input.

3.2.2 Subgoal 2

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

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 Tyer's `Turkic_TTS` repository, as well as the `apertium-chv` transducer for Chuvash. It's unclear whether a transducer is necessary for training the synthesizers.

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'm admittedly unfamiliar with how to create branches in Git, so I'll be committing to `master`.

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.

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. Create New Voice with Ossian & Merlin by Josh Meyer
2. Let's make a Chuvash voice! by Josh Meyer
3. Mozilla LPCNet Demo by Jean-Marc Valin