**Prepared by:** Matija Šantek, Vjeko Glibota, Andre Garašić, Ivana Galić

**Mentors:** Snježana Draženović Đoja, Mateo Majstorović

**Table of contents**

[Table of contents 1](#_Toc83120176)

[1 Summary 2](#_Toc83120177)

[2 Introduction 3](#_Toc83120178)

[3 Solution description 4](#_Toc83120179)

[3.1 Architecture 4](#_Toc83120180)

[3.2 Functional specification 5](#_Toc83120181)

[3.3 Information flows 5](#_Toc83120182)

[4 Source code organization 6](#_Toc83120183)

[5 Future work 7](#_Toc83120184)

[6 Tips and tricks 8](#_Toc83120185)

[7 Installation and run instructions 9](#_Toc83120186)

[References 10](#_Toc83120187)

[Appendix A: List of acronyms and abbreviations 11](#_Toc83120188)

# Summary

For the Croatian-speaking area, there are several solutions for text-to-speech, all of which are owned by foreign companies. The need for such a tool has long been recognized, and the benefits of the TTS application are diverse and applicable in a wide variety of fields. This is a strong incentive for the development of a quality tool for the Croatian language in the domestic environment. The goal is to develop a desktop application (depending on the competencies and preferences of team members) that would convert text in Croatian into speech.

# Introduction

Aware of the complexity and longevity of TTS tool development, we want to scratch beneath the surface and direct talented individuals towards creating a TTS prototype with basic capabilities.

The goal is to develop a desktop application (depending on the competencies and preferences of team members) that would convert text in Croatian into speech. They would work with a smaller database that the neural network would use to convert text into speech.

The idea is to create train and test scripts and develop an application in the Python programming language because of its accessibility and available support.

The goals we met:

* Data created in Croatian for neural network training.
* Trained and tested neural networks based on the initial version of the created data.
* Completed desktop application.

# Solution description

## Architecture

In project architecture (Figure 1) we have an application that sends text either to a trained model or to a script that generates a robotic voice. After that, that model or script (with the help of wav recordings of letters) generates an audio file which is then played.

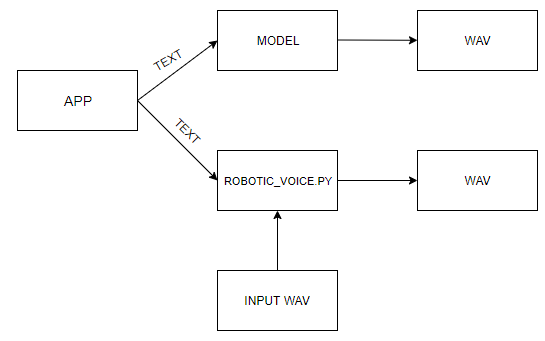


Figure . Project architecture

In the model architecture (Figure 2) we have audio files, a list of sentences and config files that are used to generate the model through the train.py script. Also, the train.py script generates self-feedback.

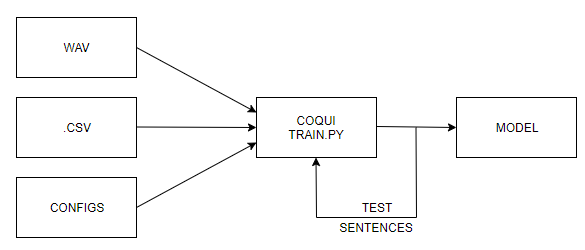


Figure . Model architecture

## Functional specification

The functions of the ultimate desktop application are:

* text input,
* speech conversion,
* robot voice playback (male / female),
* audio playback,
* text clear,
* text deletion,
* advanced settings,
* about us.

## Information flows

The end user enters the sentence and turns it into a robotic voice or a trained TTS model. The app allows him to select a female or male robotic voice and allows to run a pre-trained TTS model. User can insert him own custom model. In the end, the user has two options - play and delete.

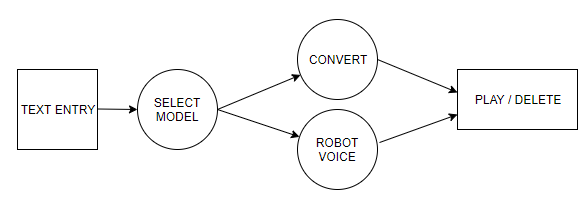
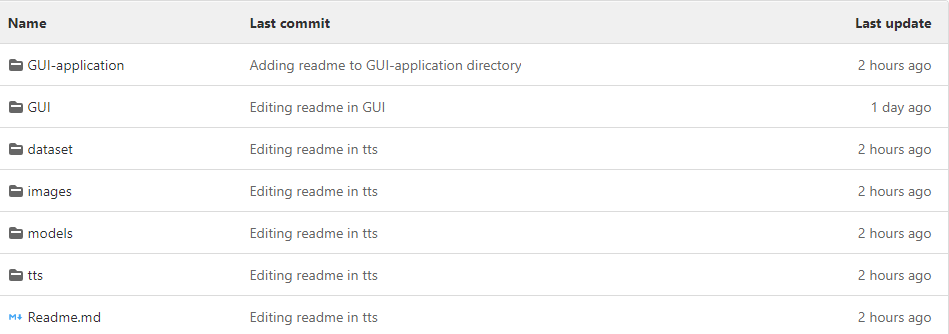
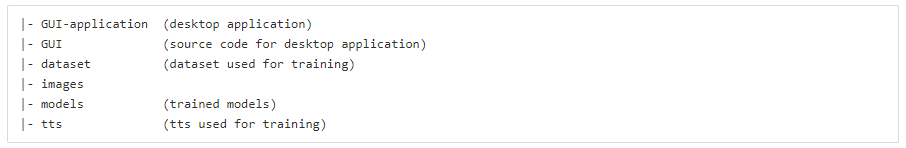


Figure . Information flows

# Source code organization





# Future work

Some of the things that could happen in the future:

* Develop a mobile and web application that would turn text in Croatian into speech.
* Increase the dataset that the neural network would use to convert text to speech.
* Better adjust training parameters to better suit the new dataset and to get better results.
* Integration into other applications as additional functionality.

# Tips and tricks

* When recording an audio files, we must be careful to record in one channel (mono channel), 16 000 hz num\_freq.
* During base recording, background noise elimination is necessary.
* We need to edit the cleaners in the configuration file. These are the functions that edit the initial sentences (trim, lowercase, ...).
* When the opening sentence is added, it is first converted into phonemes and only after that it goes into text-to-speech. Phonemes are generated based on the Croatian language.
* Use ISO 8859-1 or ISO 8859-2 depending on the application, because UTF-8 does not work.
* Use this version of TTS because most of the documentation is written for it: https://github.com/coqui-ai/TTS/tree/1c41c64a594c16e765d31cecd3d9f0485a1f1b2e.
* Train with a graphics card because it speeds up the coaching process.
* Generate scale\_stats.npy because it significantly improves training results.

# Installation and run instructions

How to use TTS app:

* This is text to speech GUI application that enables user to enter sentence and convert it with robotic voice or a trained TTS model. The app allows you to choose whether you want female or male robotic voice and allows you to run a pretrained TTS model or you can insert your own custom model.

Running the robotic voice:

* input a sentence you wish to play,
* select whether you want a female or male voice (female is default) in checkbox,
* click play.

Running the trained model

* go to ./models and choose the model you wish to use (e.g. model 700, 16kHz),
* go to config file of your selected model (./model 700, 16kHz/config.json) and change audio.stats\_path to match path to scale\_stats.npy file of your selected model (.../model 700, 16kHz/scale\_stats.npy),
* open TTS.exe,
* in Advanced settings enter your model name and press insert,
* enter sentence you would like to convert and press button convert.

Running app with your custom model:

* insert your model with scale\_stats.npy, config.json and you\_model.pth.tar into ./models - follow the steps above

# References

1. Erissson TTS: <http://192.168.209.11/sc21_etts/erissson-tts> (accessed 09.09.2021.)

# Appendix A: List of acronyms and abbreviations

|  |  |
| --- | --- |
| TTS | Text-to-Speech |
| ISO | International Organization for Standardization |
| UTF | Unicode Transformation Format |
| GUI | Graphical user interface |