Talgerving

Designing an (Icelandic) TTS

Using Festival and Docker

May, 2022

Name of student: Eike Blomeier

Supervisor/s: Atli Sigurgeirsson and Gunnar Örnólfsson

Inhaltsverzeichnis

Introduction and Background

Text-to-Speech Synthesis is about to make computers read textual input out loud with a subgoal of making the computers voice sound as natural as possible. Since it's difficult to create natural sounding human voice completely artificial, sentences must read by a person must be recorded and stored. While recording every word of a dictionary might be not an impossible but very time-consuming task, e.g., Icelandic contains about 43.000 words[1], English about 578.707[2], and German about 136.240 words[3] it will become a nearly impossible task if all possible alternations and pronunciations (e.g., end of sentence, end of question etc.) are considered. Additionally, if new words are added to a language, these words must be recorded and added to the system. Instead, by using phonemes, this issue can be overcome. Since phonemes, like letters, are appearing in words repeatedly. And different phonemes can be combined to different words of different length. Using this approach, a much smaller and more flexible search space must be recorded, which makes the TTS much more dynamic.

Still, some difficulties and challenges remain and must be overcome. Therefore, users of TTS created voice should carefully consider the use case (e.g., a bus stop announcing system, or a weather forecast system) and decide on a smart strategy on recording all the phonemes. While a human voice might not change a lot in a single conversation, it does on different days depending on multiple factors. This might result in a weird sounding computer generated voice. Additionally, it is probably not enough to have a single recording of each phoneme but multiple are needed since a phoneme can be pronounced slightly different depending on its appearance within a word or sentence as well as previous and subsequent phonemes.

Festival

Festival is a multi-language building speech synthesis system. It is developed and released by "The Centre for Speech Technology Research" of the University of Edinburgh under the X11-type license. It allows users to use pre-recorded as well as self-recorded audios to create and train an own TTS. Using a modular design and different waveform synthesizers, allowing the user to modify and personalize to achieve a best fitting model depending on the users usecase.

[1] Orðabók Háskólans
[2] English Wiktionary
[3] German Wiktionary

Unit Selection

Unit Selection selects a phoneme based on the previous and following phoneme. For all the possible phoneme combinations, a cost function is used to rate the combination of the phonemes. Finally, the best ranked combination of phonemes will be picked and used to create the audio. To save computations, unit selection will concatenate costs for subsets into the total costs.

Clustergen TTS

Compared to *Unit Selection*, a Clustergen TTS uses a predictive approach to pick the most fitting utterances. Therefore, the input data will vectorized every defined timespan. These vectors are used to build a tree, minimizing impurity as well as train an HMM used to predict the most fitting phonemes.

Experimental Setup

To create an own TTS using Festival, training data must be generated at first. Therefore, a set of pre-recorded Icelandic sentences of different length and content where used. Out of this set, a subset of 500 sentences was picked to train the unit selection model in Festival.

Script Design

To pick the *right* 500 sentences out of the set of pre-recorded audios, the following steps where performed:

1. Loading the dataset into two dictionaries (diphones, sentences)
2. Computing a sentence score for each sentence
3. Picking the sentence with the highest score
4. Remove the sentence from the sentences dictionary
5. Remove the diphones from the picked sentence out of the diphones dictionary
6. Remove the diphones from all sentences containing the diphones
7. Update the sentence scores
8. Repeat steps 3-7 until 500 sentences are selected

The sentence score was computed as follows:

If the sentence didn't contain any new diphones anymore, the score was set to zero. Otherwise, following formula was used

$$score = \frac{Number\ of\ diphones\ in\ sentence}{Number\ of\ unused\ diphones} + \frac{1}{Number\ of\ symbols\ in\ sentenc} + \frac{Number\ of\ diphones\ in\ sentence}{Number\ of\ symbols\ in\ sentence}$$

The Dataset

For the dataset, a given pre-recorded (male) dataset, containing 4731 recordings, was used. This dataset then was sliced down to 500 audios using the steps from explained above. Afterwards, the selected 500 audios were trimmed to remove pauses in the beginning and the end of the audio file. To archive this, the Librosa library was used. Trimming was performed manually. This means, five random audios were picked and trimmed using different frequencies and 55db were chosen to be the most suitable frequency to use for trimming. It is not cutting of valuable parts of the audio but still sensible enough to ignore disturbances in the beginning and end of the audio file. A sample can be seen in Figure 1 below.

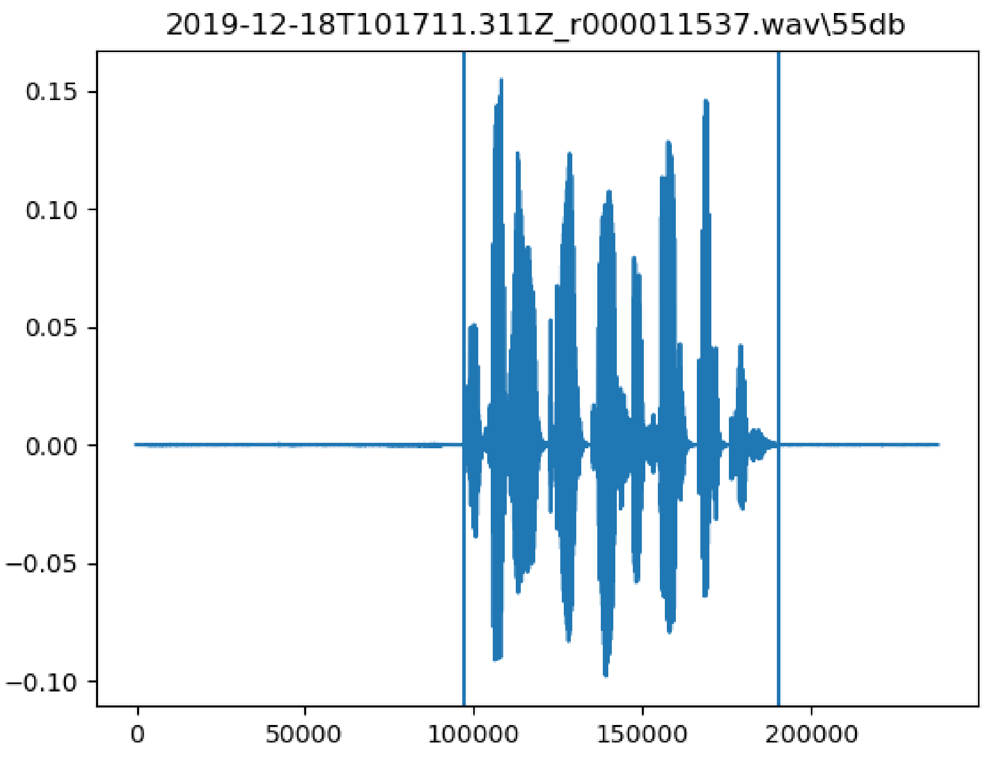


Figure 1 - Trimming using 55db (sample)

Unit Selection in Festival

The following steps are performed, using Unit Selection in Festival:

1. Load the prepared training data into the environment
2. Download the pre-build model into the environment
3. Copy *build-unit-selection.sh* into the environment
4. Change the VOX parameter to use own data
5. Run *build-unit-selection.sh*
6. Create text-file containing example sentences to build

7. Create example outputs by running *add_to_lexicon.sh* and *synth_file.sh*

Clustergen in Festival

The following steps are performed, using Clustergen in Festival:

1. Make a copy of *build-voice.sh* (*custom-build-voice.sh*)
2. Delete unnecessary parts of the script
3. Change the VOX value to use own data
4. Change the number of values to train on to dynamically use all the audios
5. Run *custom-build-voice.sh*
6. Create text-file containing example sentences to build
7. Create example outputs by running *add_to_lexicon.sh* and *synth_files.sh*

Results

Script design

The main goal of the script I designed is to achieve a high coverage while keeping the sentences to read of a minimum length. This is done, to exculpate the reader from the difficulty of reading and recording long sentences with a minimum of variations in the reader's voice. Following this idea, a total coverage score of 0.8239 is accomplished. The following figures a showing the sentence length compared to the covered (new) diphones (Figure 2) and the score values for the sentences (Figure 3).

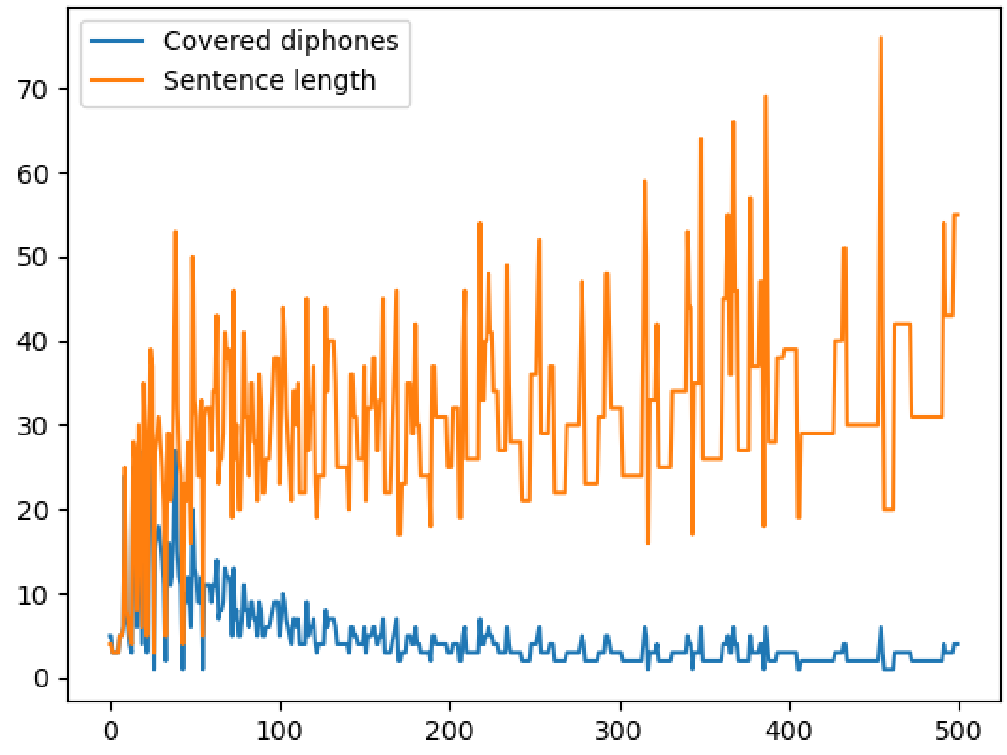


Figure 2 Covered diphones vs. Sentence length (taken from Assignment 4)

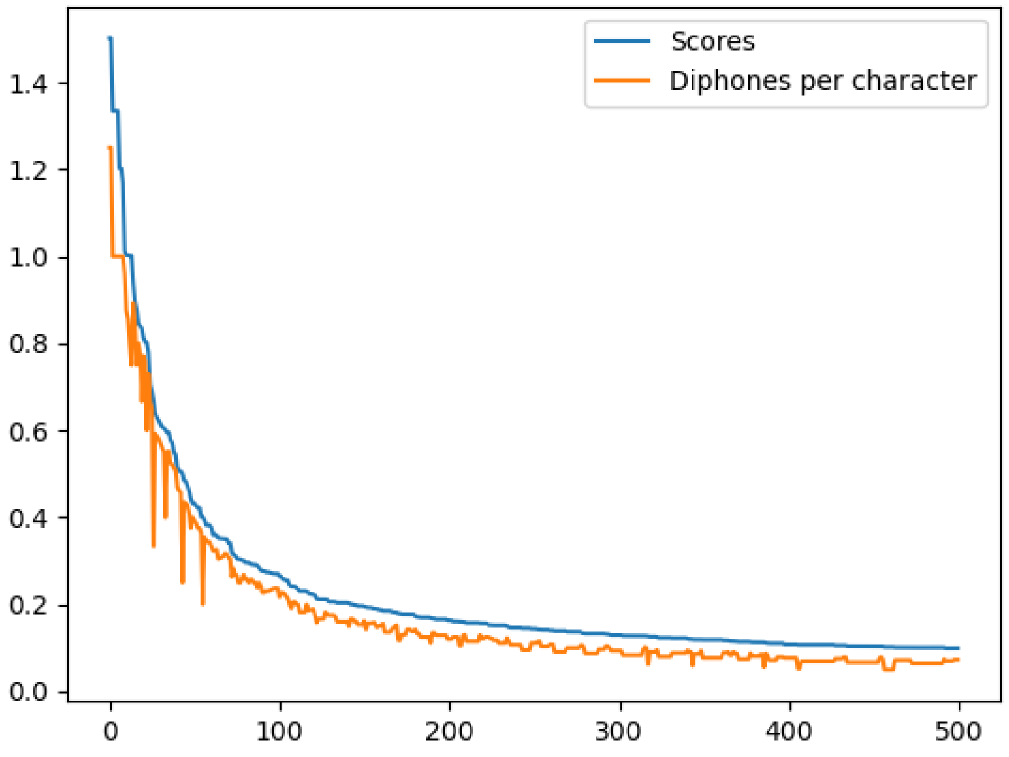


Figure 3 Scores and diphones per character (taken from Assignment 4)

The dataset

To create the dataset which will be used in the upcoming TTS creation, a subset of 500 sentences were taken from a dataset of almost 5000 pre-recorded sentences. Figure 4 shows the total amount of tokens compared to the recording duration of each sentence.

Since the recording at some silent parts in the beginning and the end, each recording must have been trimmed to remove the silent parts. This was done in a manual way with a frequency value of 55 db. Further explanation on this decision is explained in Experimental Setup. Table 1 is showing the trimming results on example recordings.

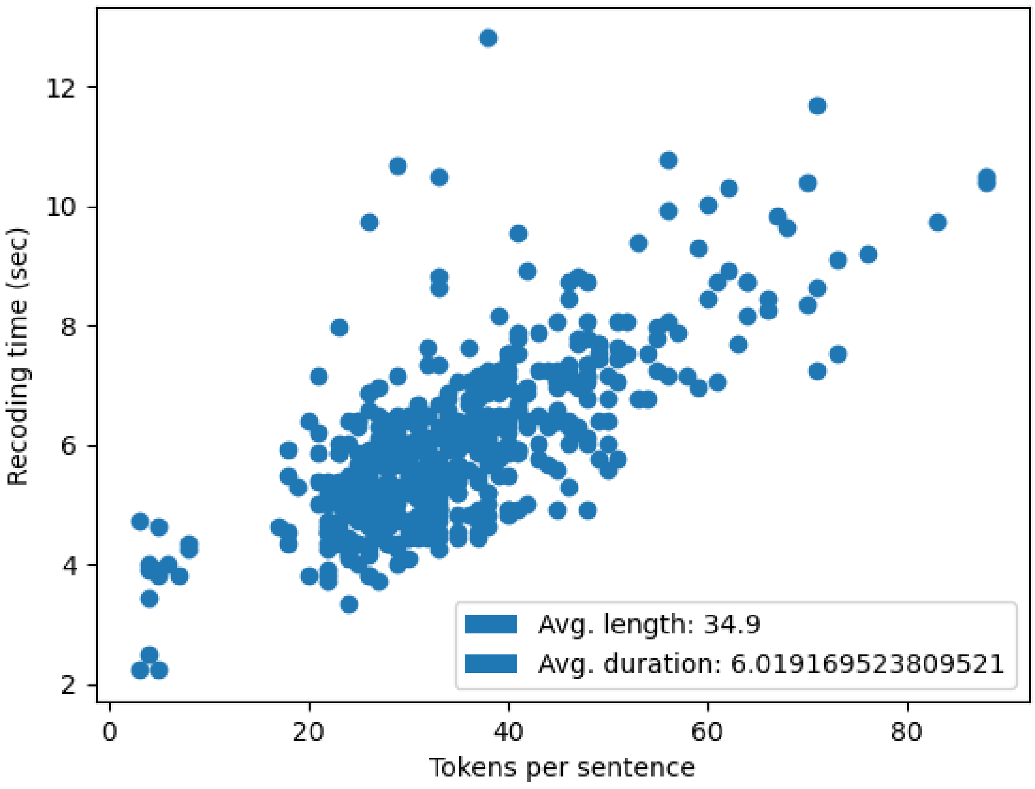
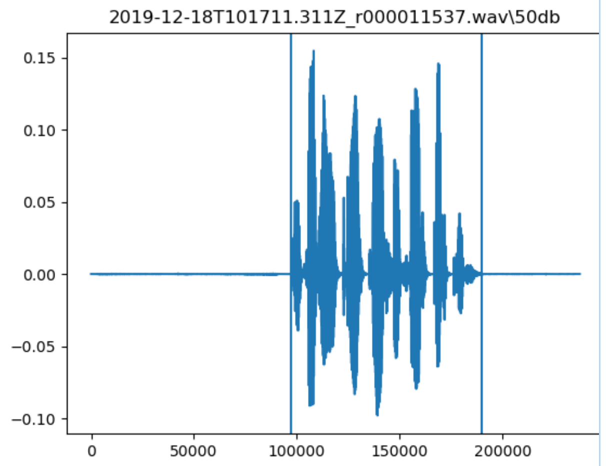
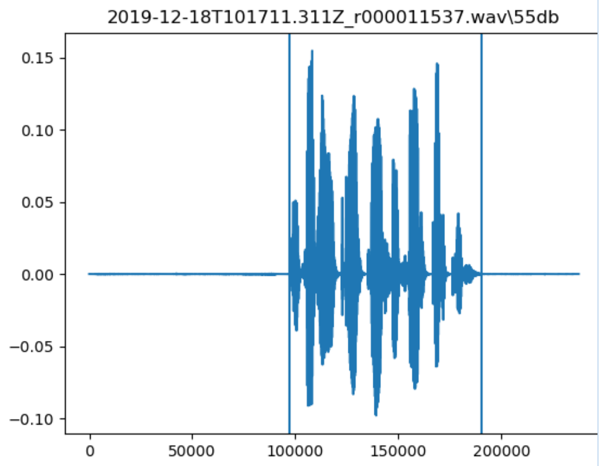
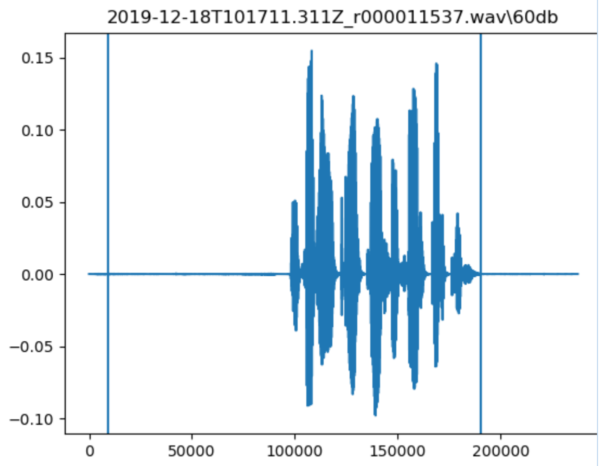
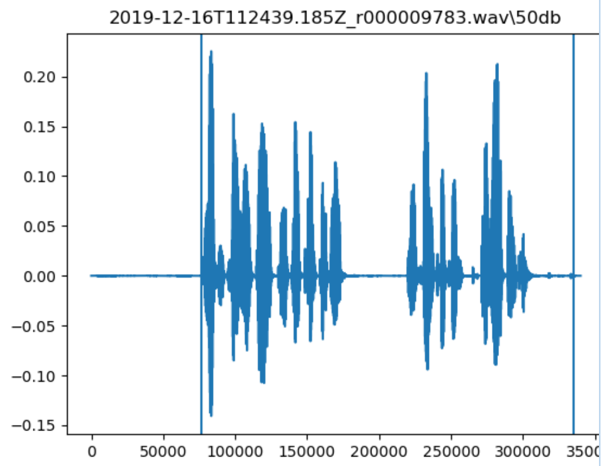
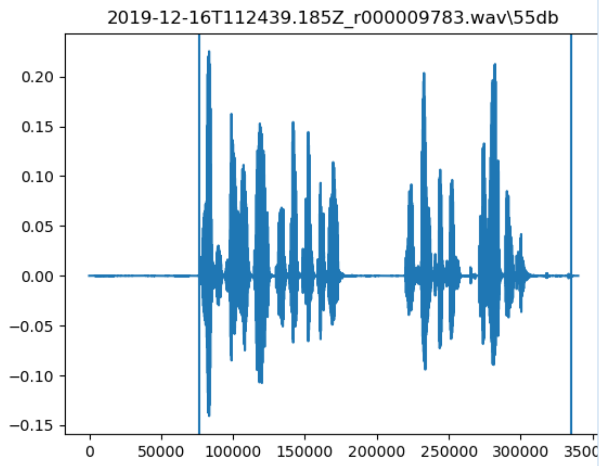
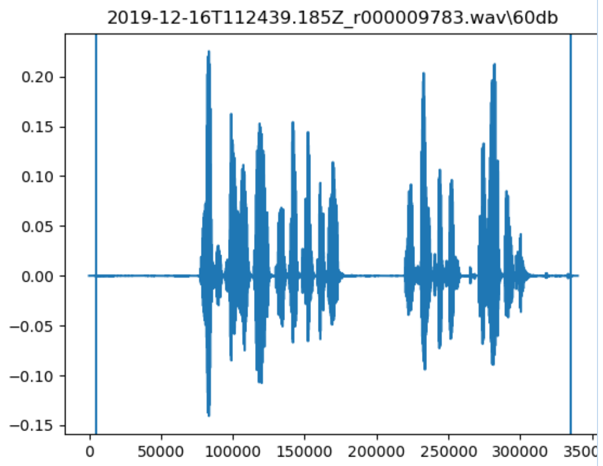
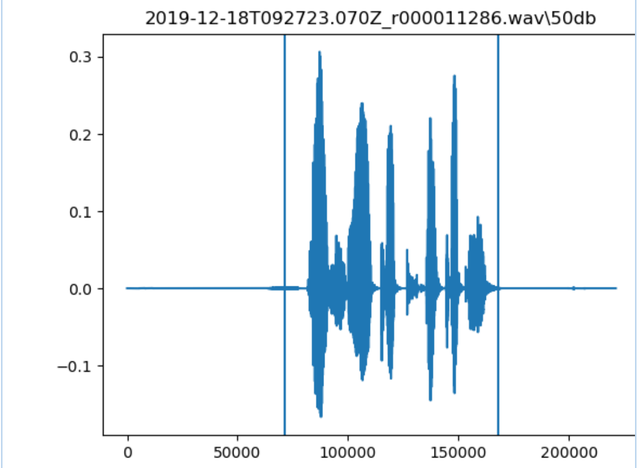
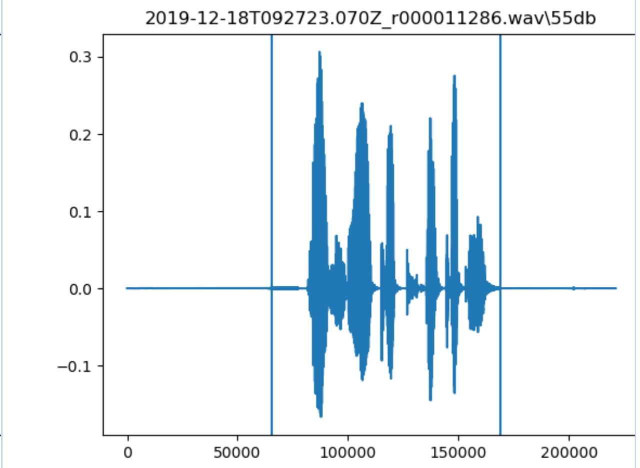
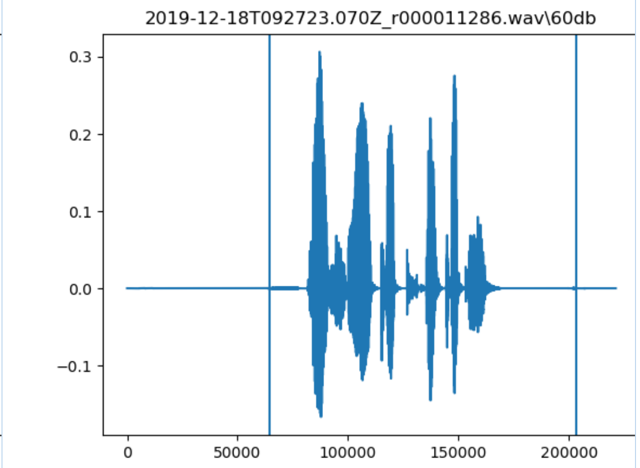


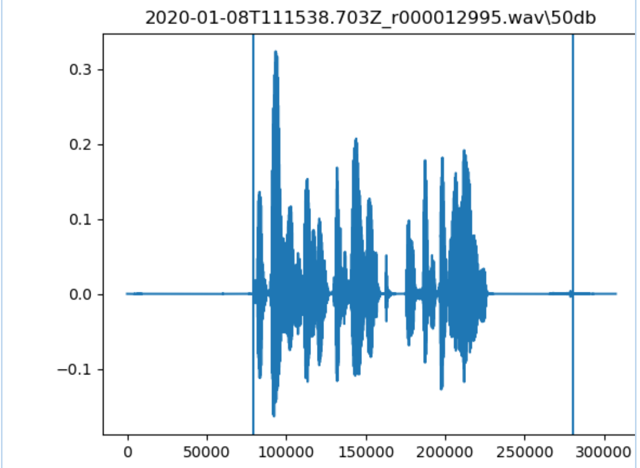
Figure 4 Token-duration scatterplot

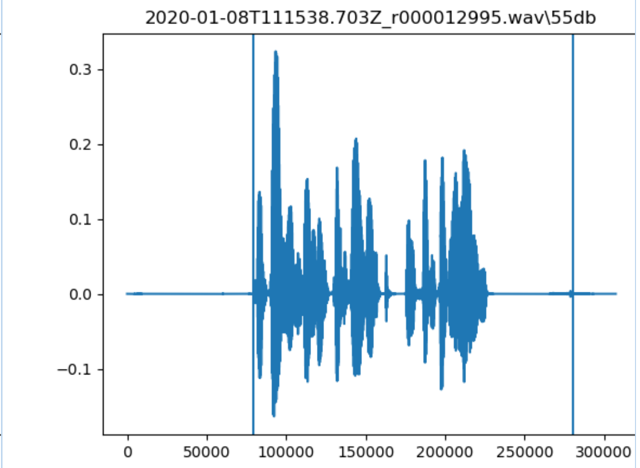
db 50	db 55	db 60
		
		

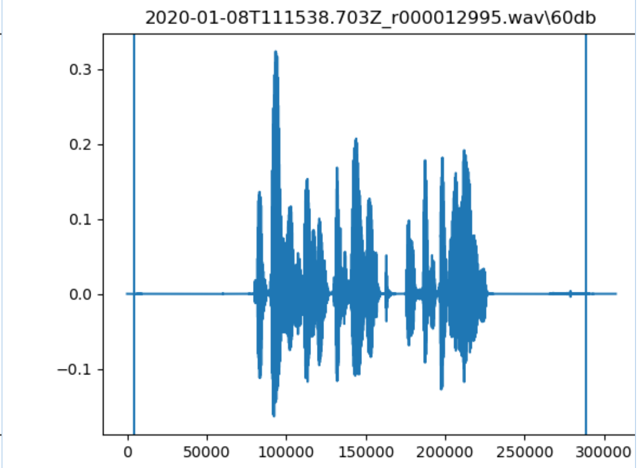

2019-12-18T092723.070Z_r000011286.wav\50db


2019-12-18T092723.070Z_r000011286.wav\55db


2019-12-18T092723.070Z_r000011286.wav\60db


2020-01-08T111538.703Z_r000012995.wav\50db


2020-01-08T111538.703Z_r000012995.wav\55db


2020-01-08T111538.703Z_r000012995.wav\60db

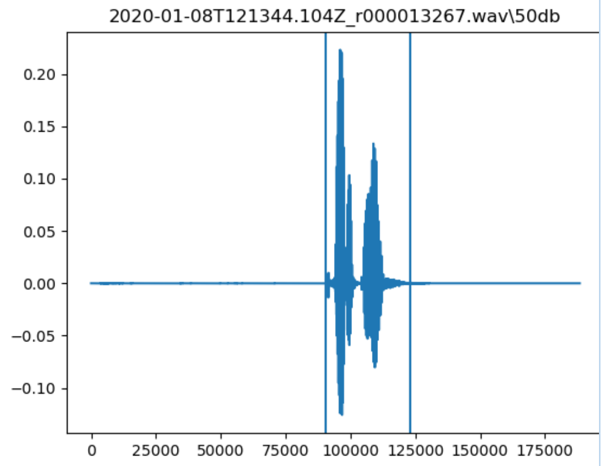
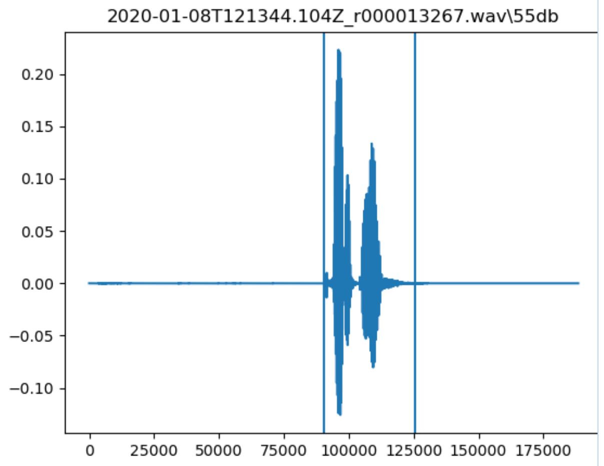
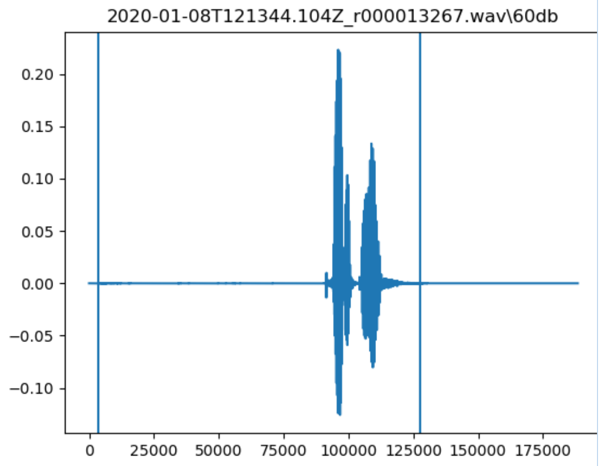
		

Table 1 DB values for trimming (comparison, taken from assignment 5)

MOS

The following figures are showing some descriptive statistics on the output of the Festival TTS. Figure 5 is showing the average MOS for each sentence. Figure 6 displays the total number of responses for each possible score value [1-5], and Figure 7 is representing the score for each sentence by every participant of the survey. It is quite interesting to see that listener number 4 looks like an outlier. This is something what must be considered on the evaluation of the MOS, since the total number of participants is low (4) and possible 25% of the total responses are corrupted and not usable.

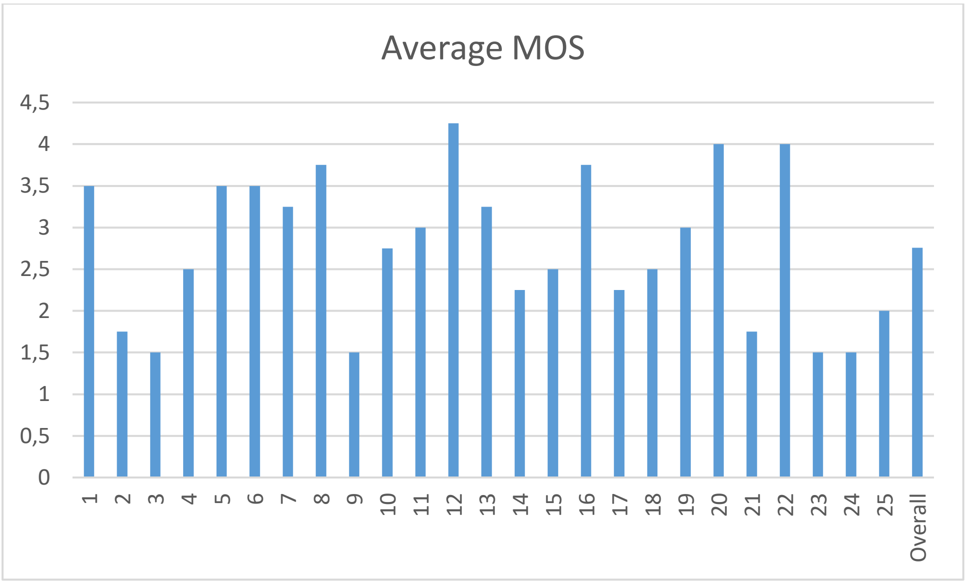


Figure 5 Average MOS per sentence and overall

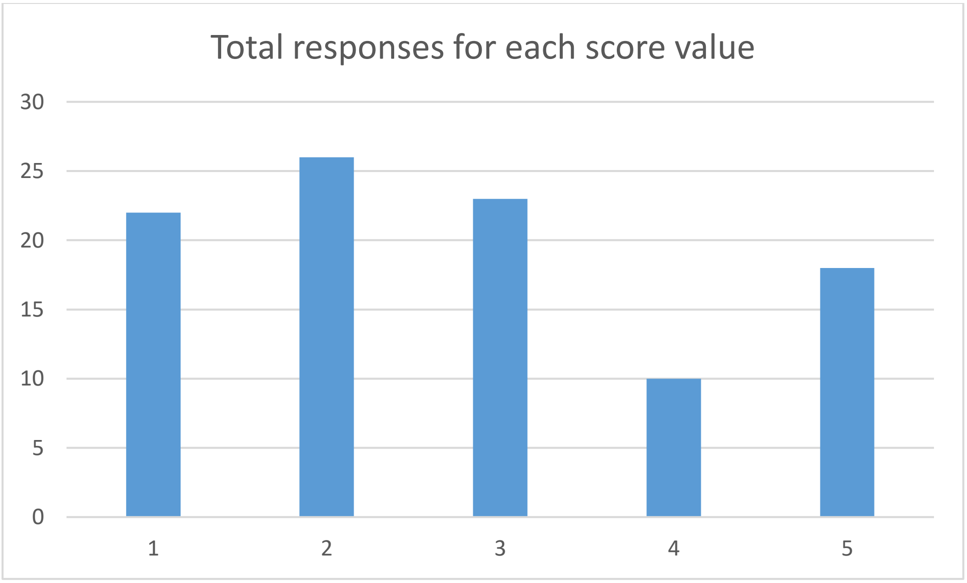


Figure 6 Total responses for each score value

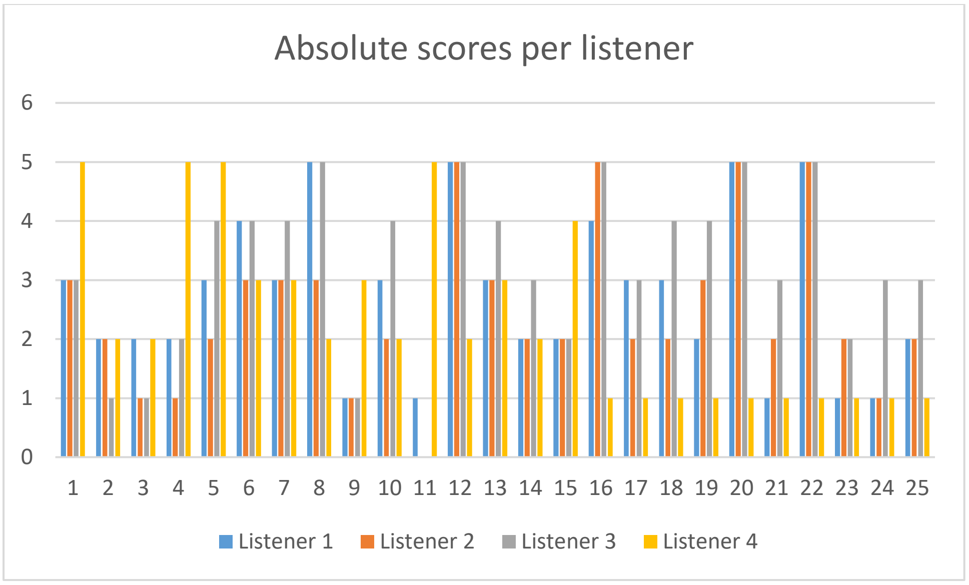


Figure 7 Absolute scores per listener and sentence

Finally, the two different Festival TTS models are performing very different regarding their naturality. As the author doesn't speak Icelandic he can only rely on the sound and not what is said. The main difference here is that the Unit Selection Voice has a more *natural* sounding, while the clustergen voice sounds more *electronical* and *disturbed*.

Conclusions

In the authors opinion, the unit selection models sounds more *natural* compared to the more *electronic* and *disturbed* voice of the clustergen model. Both models were trained on 500 sentences, but the clustergen voice used a given model to train with the 500 sentences. This is a fairly low amount of data, compared to all the possible sentences and diphone combinations. Regarding to that, models could be improved by increasing the amount of training data on the one hand. On the other hand, it could be more useful to think of the use case for the TTS. E.g., a TTS what is designed to call out train connections in train stations probably won't need to call out weather forecasts and vice versa. This could improve the use-case result significantly. Additionally, this TTS was tested under the best possible conditions (headphones, no noise from the outside) while in reality, these conditions are rarely given and the TTS should better be tested in places where it will be used in the future.