DT2112: Deep Learning Speech Synthesis on Podcast Data made Simple

Adam Feldstein Jacobs[1], Javier García San Vicente[1], Johannes Benedikt Wichtlhuber[1], Mikolaj Bochenski[1], and Supervisor: Éva Székely[1]

[1]*EECS, Division of Speech, Music and Hearing, KTH Royal Institute of Technology*
{*adamjac,jgsv,jbwi,mkbo*}*@kth.se*

March 2021

Abstract

When building speech and conversational systems it is desirable to achieve a natural prosody and sentence flow. This can be done by training speech synthesis on spontaneous speech, but using that data is hard due to its high variability. We trained a TTS system by building a reusable preprocessing data pipeline that segments and filters podcast data from Spotify. The training and synthesis is done with Tacotron2 on two GeForce RTX 2080 GPUs. The result is a spontaneous and natural voice, which you can listen to at [1]. The code is hosted at [2].

Acknowledgements

Thank you: Éva Székely for supervising us and giving us hope in the feasibility of the project; Jens Edlund for providing hardware resources and general enthusiasm about the project; Jussi Karlgren for introducing us to the Spotify Podcast dataset and its possibilites.

Contents

1 Introduction

Many speech synthesis systems rely on read speech data because it is historically what has been available, and because spontaneous speech data is difficult to utilize as it is highly variable and disfluent [3]. When building speech and conversational systems it is of importance to include features that humans can relate to. Otherwise it will not be considered natural. The purpose of speech is both transactional and social, and these purposes are build upon traits that comes from spontaneous speech [4] - for example mutual understanding and trustworthiness. Therefore, in order to improve the naturalness of synthesized speech, this project created a TTS system built with the recent end-to-end deep learning model Tacotron2 [5] and spontaneous speech data found in Spotify podcasts [6] - “The largest corpus of transcribed speech data“ according to Spotify, including 60,000 hours of speech. Our system relies on a specific pre-processing data pipeline which makes the process of using spontaneous speech data simpler.

1.1 Problem

There are multiple technical issues to solve in order to build an easy to reproduce and robust end-to-end speech system with spontaneous speech data. Firstly, the end-to-end deep learning model needs much text and speech data [5] to train, which is found in the Spotify dataset. However, it is of importance to have correctly worded and timed transcriptions since spontaneous speech is unstructured in its nature. Secondly, as mentioned by Szekely et al. [3], since spontaneous speech is not manually divided into distinct utterances as read speech is, speech segmentation is needed. Furthermore, the podcast data contains outliers on both episode and show level when looking at acoustical features such as pitch and speech rate [7]. Therefore automatic filtering of outlier segments is needed to avoid manual work. After the segmentation and filtering there must be enough training data left.

After the pre-processing comes the challenge of training and tuning the deep learning network. Last but not least is the software engineering challenge of making the the data pipeline clean and reproducible.

1.2 Objectives

The objective of this project is to train a text-to-speech speech synthesizer using Tacotron2 with data from a Spotify podcast show, and then use the synthesized podcast voice to read some simple utterances. The synthesized voice can also be used for reading of a podcast description for any promotion purposes. This objective contains sub-goals:

- Extract relevant podcast data
- Segment data

- Filter outlier segments by relevant acoustic features.
- Train at least one voice with Tacotron2
- Produce simple utterances
- Evaluate synthesized voice with relevant metrics

1.3 Motivation

The project is interesting because the problem of producing a natural sounding voice from spontaneous speech is not a trivial. Furthermore, the team wants to train a state-of-the-art deep learning model, which requires more hardware resources than the common student sits on.

2 Background

In this section we discuss relevant background aspects of our work, in particular Speech Synthesis and Deep Learning. We will briefly discuss traditional aproaches before we get to the state of the art solutions today.

2.1 Evolution of Speech Synthesis

We can begin addressing the evolution of speech synthesis by mentioning the formant synthesis [8], which was based on the production of sound waves from the synthesis of the main fundamental frequencies of the phonemes, called ”formants”. The underlying idea of this method is the fact that speech can be mostly understood by those basic frequencies produced at the vocal tract. Although vocals were easy to achieve, some consonants, as the burst ones, were difficult to synthesise. Besides, interpolating only from a few frequencies made the voice sound unnatural and robotic.

The next approach developed was unit selection synthesis [9]. It is based on the concatenation of what is called ”units”, consisting of sound segments of phones or diphones (a combination of phones) between 10 milliseconds and 1 second. Once all the units are recorded, the synthesis process consists of identifying the most feasible units for each word and concatenating them to form the wave of sound. Nevertheless, apart from being computationally heavy, as the system had to search among all the possible units, the union of different units sounded abrupt and discontinuous.

That is why the next revolution in terms of speech synthesis was brought by HMM synthesis [10]. It produces sound sequences according to an automatically trained Hidden Markov Model. This not only avoids the computation about all the possible unit combinations but also generates an automatic probability model for transitions, based on their hidden states. This produces more natural transitions between different sounds.

Nevertheless, the next revolution - and current state-of-the-art - is deep neural networks, which are addressed just below.

2.2 Speech Synthesis and Deep Learning

There are different representations of speech, usually in forms of different spectograms, such as zcr-grams and mel-spectograms. We will briefly discuss Mel-spectograms, since they play a major role in our pipeline. Spectograms are basic tools in speech analysis, which are sometimes defined as intensity plots operating on different scales such as the log scale, basically visualizing the energy at different frequencies. **Mel-spectograms** however are conditioned on the mel-scale, a human perceived scale of pitches. The authors of Tacotron2 retrieve mel-spectograms by applying a nonlinear transform on a linear frequency spectogram. Furthermore they argue that mel-spectograms capture the frequency content of a spectogram with fewer dimensions, consequently making it a "simpler, lowerlevel acoustic representation of audio signals" [5] [11].

Tacotron2 is an end to end Text to Speech(TTS) synthesiser, mostly made of stacked neural network layers. Since Tacotron2 is used for training with the data we want to introduce the most imminent building blocks of the architecture in the following segments [5].

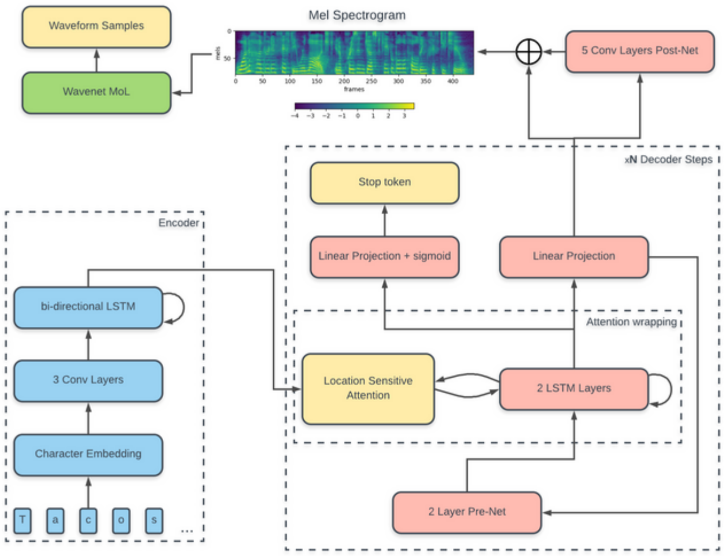
In Tacotron2 one would find many elements from modern Deep Learning frameworks. Starting with **character embeddings**, which convert characters into high dimensional vector spaces with desirably semantic information. The model uses many different neural network architectures, such as **Convolutional Neural Networks(CNN)**. CNNs are fully connected deep neural networks that use a mathematical operation called convolution, a special linear operation, in at least one of their layers instead of a general matrix multiplication. CNNs are often used in visual computing because they can process various data lengths and result in reasonable representations in their layers for the invariances and variability in image data [12].

Recurrent Neural Networks(RNN) are specialized neural networks with a focus on sequential data. There are many designs one can imagine when designing RNNs. In general most of them have recurrent connections between hidden units and result in one output at each time step. More sophisticated models implement recurrent connections from an earlier timestep to the next time step. A special case of Recurrent Neural Networks are gated RNNs. A frequently used gated unit layer is the **Long Short-term memory(LSTM)** layer. LSTM networks adress a common problem in Deep Neural Networks, vanishing gradients, by circumventing recurrencies with gates("Input gate"/"Forget gate"/"Output Gate"). These gates enable the network to establish a long short-term memory that traditional RNNs seem to miss [13].

The final building block of Tacotron2 is the Vocoder. The authors use a modified version of the **WaveNet architecture** to transform Mel-Spectograms into waveform samples. In a stand alone application, WaveNet is a generative model, thus it takes in suitable parameters - characteristic to the target voice - and generates speech. But these parameters require a lot of human expertise. The authors circumvented this problem by allowing the modified version of WaveNet to input Mel-Spectograms [14] [15].

By now, we have introduced the important building blocks of the Tacotron2 architecture, thus we can discuss now the overall architecture of our model, a classic **decoder-encoder architecture**. Tacotron2 decodes text into character embeddings, which are further processed by convolutional layers and LSTMs. The decoder part of the networks processes this representation with local attention and recurrent connections. Local attention is a concept of machine learning models, that allows our model to focus on the important bits of our data. The end results of the decoder is a Mel-Spectogram. The spectogram serves as the input of our modified WaveNet, which produces our final output, our waveform samples. One can revisit our architecture in figure 1.

Figure 1: The decoder-encoder architecture of Tacotron2. The encoder produces a suitable presentation for the text, the decoder decodes this information to a Mel-Spectogram. The modifed WaveNet architecture produces our audio samples [16].



2.3 Podcast Data

This section describes the origin and characteristics of the data used for our purpose.

The Spotify Podcasts Dataset [6] collects data from around 100.000 episodes of different podcasts shows uploaded to Spotify. The data for each episode consists of an *.ogg* audio file, a text file containing the transcript of the episode, and other metadata. All the audio files together sum up to around 50.000 hours of data, and the transcripts contain more than 600 million words.

The main language of all the collected audio is English. Nevertheless, the dataset contains podcasts about any topic, and thus some of them contain, apart from natural speech, voice reading, music, or even speech in other languages.

As stated in the paper of the dataset, Google's cloud speech-to-text API2 is used to provide word level transcripts and speaker diarization. The transcripts are structured in a way that the start and end times of each word are stated. Some words also contain another attribute corresponding to the speaker tag, which identifies the speaker that is saying that word. However, these tags are only consistent at episode level, not between different episodes. This means that speaker number 1 could be a different person in episode 1 and episode 2.

2.4 Related work

This project takes much inspiration from a paper by Székely et al. [3], and tries to extend and automate some parts, and put the steps into the data pipeline (see Methodology). In that paper, a TTS system is trained with Tacotron2 on spontaneous speech data from the ThinkComputers podcast. The data is segmented into smaller utterances via breath groups using a speaker dependent CNN-LSTM breath detector. Thereafter useful segments are picked out manually. These segments are used to in training with Tacotron2, which is then used to generate mel-spectrograms. Griffin-lim is used to generate the audio data from mel-spectrograms.

As mentioned, one part of our pipeline deals with breathing detection. We used a pretrained model from Székely et al., which enabled us to segment our data reasonably [17]. The model was trained in a semi-supervised fashion on a CNN like architecture. We discuss the performance of the model later on.

Since we use a specific dataset [6] we looked at a paper that did data exploration on the same dataset [7]. The point of this related study was to investigate what stylistic characteristics can be used to do search for podcasts, from both a user and computational perspective. In the data exploration part, statistics on acoustic features were collected on episode, show, and genre level. As you can see in the Methodology, we will utilize statistics on episode and show level.

The exploration study shows that each acoustic feature has a clear distribution on both episode and show level. Examples of some distributions can be seen in figures 2a, 2b, 2c below. More details can be found in the paper doing data exploration.

Tacotron is the predecessor of Tacotron2. Unlike its successor, Tacotron is not a fully neural network approach. It still depends on a Griffin-Lim vocoder, which works with a short-time Fourier transformation. The authors motivate this change with the better audio quality produced by the WaveNet vocoder [5] [18].

There are other Deep Learning aproaches for speech synthesizing however. Flowtron is another state of the art Deep Learning solution. It is yet another generative speech synthesizer with an interesting architecture, an autoregressive flow-based generative network. It also has parameters that gives more control over the voice. One example is controlling the prosody. FlowTron does this by learning a latent space over non-textual features which can be looked at and manipulated to give more control the generative output.

3 Methodology

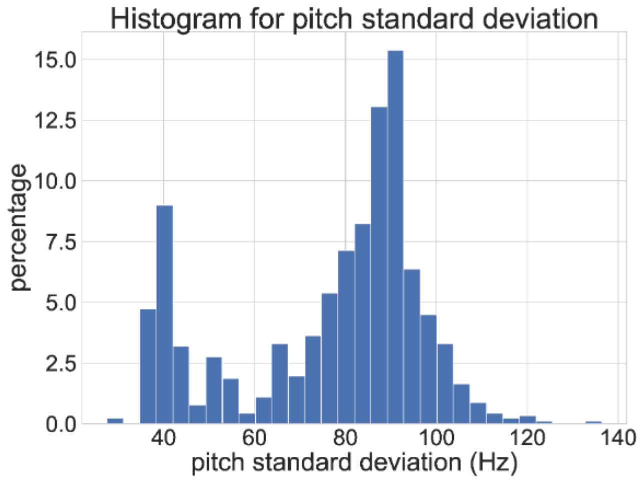
We put extra engineering effort into making the data pipeline easily reproducible for future use. We use Poetry [19] for dependency management, specifying the exact versions of packages that our code works with. A newcomer only has to issue a single command for Poetry to create an isolated virtual environment with the correct versions of all our dependencies.

Perhaps more interestingly, we use Guild AI [20] to make the data process itself as reproducible and clear as possible. Our code is structured as a series of operations which feed into one another. At each step, the user might make a choice, eg. of which podcast to detect breath in, or how sensitive outlier filtering should be. Guild AI makes it easy to run operations multiple times, making different choices each time, and keeping the output of each run in isolation for use in operations further down the line. For example, one could pick one podcast, segment it and then try many different filtering operations with just one command. A visualization of the pipeline can be seen in figure 3 below.

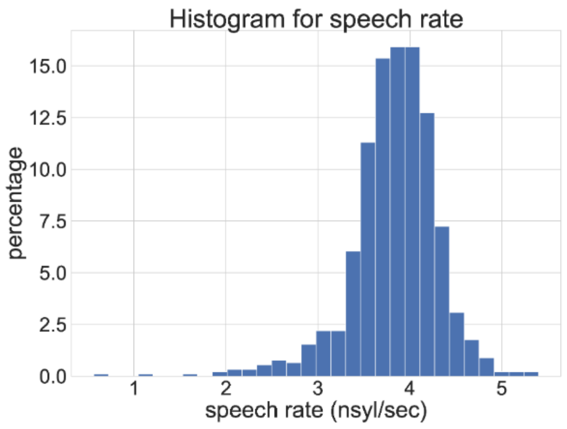
3.1 Data Extraction

The podcast data is downloaded using rclone [21] via a remote server containing the data. This is encapsulated in a guild command, but can be replaced with any data downloading commands or scripts for any other data sources. Also, since the audio data is only available in .ogg format all relevant podcast data was converted to .wav using the FFmpeg library [22].

(a) Pitch distribution on episode level



(b) Speech rate distribution on episode level



(c) Pitch distributions on show level for different podcast shows

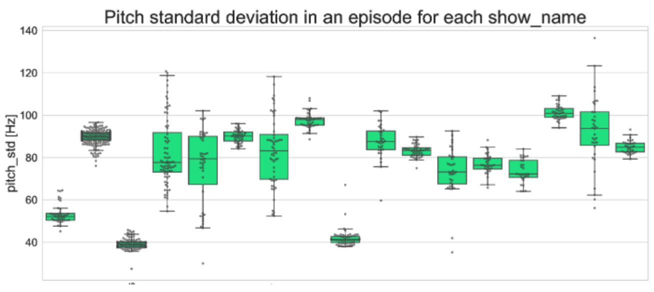
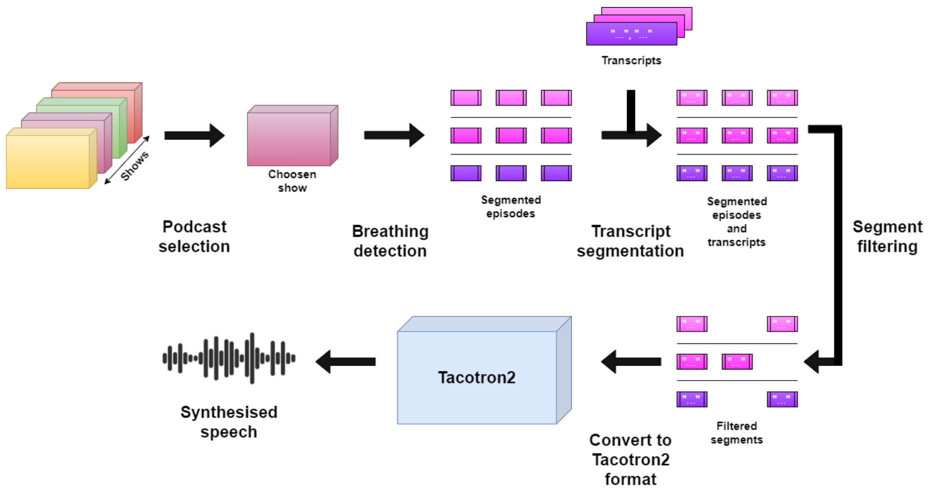


Figure 3: Data pipeline.



3.2 Podcast selection

The goal of the podcast selection was to find a podcast primarily consisting of monologue and with at least 20 hours of data. The voice should be female since the vocoder in Tacotron2 is pre-trained on female speakers.

In order to find primarily monologue podcasts a filtering was made by selecting podcast shows where most episodes only contained one or no speaker tags. By further investigation of the transcripts the speaker tagging was often incorrect, and it often classified the same speaker as two different speaker tags. However, looking at shows with mostly one speaker according to the transcripts, and with enough episode hours, served as a heuristic for picking out a list of potential candidates. However, to select a suitable podcast a few hours of manual work listening to the podcasts was required. This manual work can be reduced or even automated by a better speaker separation system, which was not included in this project. Also, selecting podcast author by gender was done manually from the select of podcast candidates, but this could be automated with gender classification for each speaker tag.

3.3 Breathing detection

We knew from previous research that Tacotron2 is very sensitive to breath and silence, we therefore aimed to segment our data accordingly. We used a pretrained model that was trained on a female voice from Székely et al. [17] within a podcast dialogue, which also motivated us to look for a female podcast to train on. Since the model was pretrained with one channel audio data, we had to convert our data from two channel (stereo) data to one channel (mono) data. The model classifies utterances in breath, clean data and noise. Note that

”noise” is interpreted by the model as another person speaking, since the model was trained on dialogue data and trained to detect one specific voice. After running the model for the first time we noticed that the utterances are mostly unreasonably short. We consequently included the ”noise” in our utterances, which turned out to be mostly clean, although some utterances included indeed noise and silence. Since we had a solid amount of data, we were quite confident that the segment filtering part of our pipeline would help us filtering these undesirable utterances.

3.4 Transcript segmentation

Once the original audio source is segmented according to breaths, the next step is segmenting also the transcripts according to the divisions made in the audio. This means that the output of this phase of the data pipeline corresponds to each segment previously produced, but now with its correspondent transcript, which is a list of words.

This transcript segmentation can be easily performed by looking at the start and end time tags of each word in each transcript file and including it in the transcript of the relevant segment for that episode. We include a word if its end or start time overlaps overlap with the timestamp of the speech data of the segment.

This may seem at the beginning a naive approach, as the timestamps of the words do not necessarily match the segment. Nevertheless, as the segments are made from breath to breath, it seems reasonable to assume that a breath never occurs in-between a word. Therefore, if the timestamps of both the segments and the words do not match, we can assume that it is mostly due to a lack of precision in the transcription, and thus, the word should be included in the segment transcript.

However, there are techniques that are less naive like using a forced aligner. This was not included in the project but there are many options like for example [23].

3.5 Segment filtering

Due to the natural differences of tones, intensities and speed during natural speech, all these segments present really different characteristics, which may hinder the training process. Instead of normalizing, which could lead to an artificial modification of the speech that could also disrupt the final objective of synthesising realistic voice, a filtering is performed over all the segments.

First of all, an early condition is imposed to keep only the segments of length between 3 and 10 seconds. While the lower bound tries to ensure that a minimum amount of relevant information is contained in the segment, the upper bound is set in order to avoid the model, Tacotron 2, running out of memory during its training process.

After that, as investigated in the data exploration in [7], we define four different features of interest of each audio segment: The average pitch, intensity and energy, and the speech rate. The speech rate was calculated by the number of phonemes divided by the time length in seconds, using a grapheme to phoneme conversion library [24] since text is available for each segment.
For each of the features of interest, the filtering is performed at both episode level and show level. When filtering at episode level, each segment is only compared with the segments of its same episode, avoiding taking into account the differences between the audio of different episodes.

The filtering and comparison process are performed by two different methods. Firs of them consists of setting an specific interval around the mean of each feature in terms of standard deviations. Therefore, for each feature of interest, we define that an specific segment of episode e is valid if it is inside the interval defined by $\mu_e \pm \alpha \cdot \sigma_e$. The second one consists of simply removing the outliers, by keeping an specific percentage of the central data.

With this configurations, we develop three different training sets. The summary of them appears in Table 1. For the first two, we set an α value of 1 and 2 respectively, following the assumption that the features of interest could follow a normal distribution. In that case those intervals would include around the 68% and 95% of the data approximately. The last training set simply takes the 90% of the central segments, discarding 5% of outliers in each side. All three filtering processes where performed at episode level. Those datasets can be directly converted to the input format of Tacotron 2, beginning the training process.

3.6 Training

We start from a model trained by NVIDIA [25] on the the LJSpeech dataset [26]. We then proceed to retrain the model using the data from the previous steps in the pipeline. Two GeForce RTX 2080 GPUs was used for training.

Tacotron 2 did not fit well into our general data pipeline - it is a separate repository, and it requires Docker for easy setup. We use Tacotron in accordance with the instructions provided in its repository [27]. However, we found that the following steps are necessary to train the model without running into software issues:

- Use the Docker contaner named tacotr:ver1
- Do not attempt to install any dependencies - they are already in the docker image

Number of batches	**Filtering method**	**Segment length**	**Number of segments**
16000	mean $\pm 1\sigma$	3-10 seconds	4635
36000	mean $\pm 2\sigma$	3-10 seconds	9698
22000	central 90%	3-8 seconds	6132

Table 1: Parameters used for different runs.

- Instead of using the WaveGlow model linked in that repository, use the one from PyTorch Hub [28]

We use the default values for hyperparameters, provided in the file hparams.py. The only settings that we adjust are much more technical - we set distributed_run and fp16_run to True.

4 Experiments and results

We conduct multiple experiments to try and find the optimal parameters for the data pipeline and training. Table 1 documents our attempts. None of the training runs produced results which were much better to our ear than the others, so we settled for the final one to do our evaluation on.
We compare our model with the pretrained one we started from. We input four categories of sentences into the models:

- ”Chit-chat”, ie. transcriptions of spontaneous speech form other podcasts, containing filler words such as ”like” and repetitions
- Parts of books, of the same kind the pretrained model was trained on
- Difficult sentences and words, tongue-twisters
- Unnatural/malformed sentences, eg. ones containing only punctuation

Due to lack of time, we were unable to conduct a full-scale evaluation involving external participants. We resort to a qualitative evaluation performed by ourselves. However, the reader is welcome to listen to the output of the model at [1].

In general, our model produced speech which sounded much more like it was improvised in contrast to any read speech. In fact, in some cases, the speech seemed perhaps *too* improvised, introducing significant amounts of hesitation and sometimes ”messing up” words. Overall, the audio quality is certainly lower than from the model trained on LJSpeech. Part of this is most likely due to the microphone that was used to record the podcast we trained on.

The spontaneity is, unsurprisingly, a welcome improvement for spontaneous speech - especially the filler words sound much more natural. For reading books, however, the model trained on LJSpeech is clearly superior - it maintains the intonation of a reader, whereas our model often stutters and fails to clearly articulate sentence structure. The same holds for tongue-twisters - the significant effort that goes into reading books in a very clear manner definitely influenced this result. Notably, our model mispronounces some words (eg. ”evidence”) - we speculate that they were not present, or at least not as common, in the podcast we trained on. Finally, unnatural sentences are hard to evaluate on in the first place (it’s unclear what constitutes ”better” performance) - however, our model does seem more robust to this weirdness: for example, the LJSpeech model produces a strange ringing sound when tasked with pronouncing a sentence consisting of only random punctuation marks, while our model produces breathing sounds.

5 Discussion

5.1 Limitations of the Pipeline

The pipeline is automatic in all steps except in the podcast selection. To improve that, better filtering techniques would be needed, together with a higher-quality speaker diarization of the data inside the transcripts, i.e the speaker tags. If a classifier can separate different voices and filter out other elements like music, then it will more fully automatic and fully end-to-end. Furthermore, other elements to sort out the selection data could be used, like gender classification.

Another thing is that the text was put together with the audio segments in a naive way. As we mentioned earlier, it could be improved with a forced aligner. However, we are unsure how much this would improve the quality of the voice since the breathing does quite a good segmentation. The breathing even helps to synthesize breaths for the nonsensical input, while using only the LJ speech gives weird noises.

Furthermore, we noticed that most of our work was about data pre-processing. To our surprise, we did not have to tune any hyperparameters of the Tacotron2 model, which is usually something you need to do in machine learning training. It seems that models in deep learning speech synthesis are quite reproducible parameter wise, and the most important aspect is having quality data. The preprocessing steps in our data pipeline could be improved further easily by adding more guild operations. Our filtering was for example quite general since we only look at statistical outliers but did not specifically look to filter things like music. Since the filtered segments were not investigated we are unsure if it managed to filter out music. However, it does not seem to be reflected in the voice. Another aspect here is most likely that the breathing detection will segment the utterances into parts where there is speech in between breaths, and not music.

5.2 Future Work

One application that can be built on top of our work is to generate podcasts. This will of course need to be combined with content generation. It will be quite feasible to do a monologue podcast with the model we trained with Tacotron2 since it gave a very natural and spontaneous voice according to our qualitative evaluation. However, with more state of the art models, such as flowtron, there can be more control of the voice in terms of for example prosody. A risk in Tacotron2 is that the data can have a similar style or be monotonous and therefore controlling things such as the prosody gives an advantage if you want to make the voice more dynamic.

Another application is to use our data pipeline but instead with dialogue and conversational data. In order to do this the podcast selection would need to be improved, as mentioned above. Also, the filtering would need to be extended since it only looks at one average per acoustic feature, which will be different for multiple speakers.

Furthermore, we feel it is important to mention a bit about the dangers of this technology. Our voice is natural and could most likely trick some humans. The models for speech synthesis are getting better, which might mean that there will be some point where it will be fully indistinguishable. If there is no safety mechanisms involved, it could be used to trick people.

Lastly, as a result of our voice sounding very natural, it will be used in a large scale study at TMH at KTH [29] where a method is evaluated for seeing how hard it is to distinguish human and synthetic voices. We are very proud of that and look forward to see the results.

5.3 Conclusion

In conclusion, we think we successfully trained and synthesized a natural and spontaneous voice from podcast speech, which can be heard on [1]. And together with that built a re-usable framework for the data pipeline, where the data and synthesis model can easily be replaced. The pre-processing could be improved upon to be more dynamic with the data selection and filtering. However, it be easily improved upon by programming new independent guild operations without touching our code. See the repo [2] for more info.

References

[1] Samples from our results online. [Online]. Available: https://www.notion.so/malyvsen/4967e4e8627b48f998b9524933764a21?v=645207dbb991489da6e881f06a7af622

[2] Speech synthesis spotify repository on github. [Online]. Available: https://github.com/worldyn/speech-synthesis-spotify/

[3] É. Székely, G. E. Henter, J. Beskow, and J. Gustafson, "Spontaneous conversational speech synthesis from found data," in *Proc. Interspeech*, 2019, pp. 4435–4439.

[4] L. Clark, N. Pantidi, O. Cooney, P. Doyle, D. Garaialde, J. Edwards, B. Spillane, E. Gilmartin, C. Murad, C. Munteanu, and et al., "What makes a good conversation?" *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, May 2019. [Online]. Available: http://dx.doi.org/10.1145/3290605.3300705

[5] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[6] A. Clifton, S. Reddy, Y. Yu, A. Pappu, R. Rezapour, H. Bonab, M. Eskevich, G. Jones, J. Karlgren, B. Carterette, and R. Jones, "100,000 podcasts: A spoken English document corpus," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 5903–5917. [Online]. Available: https://www.aclweb.org/anthology/2020.coling-main.519

[7] K. MARTIKAINEN, "Audio-based stylisticcharacteristics of podcastsfor search andrecommendation: A userand computational analysis." University of Twente, EIT Digital Master School, 2020.

[8] S. Lukose and S. S. Upadhya, "Text to speech synthesizer-formant synthesis," in *2017 International Conference on Nascent Technologies in Engineering (ICNTE)*, 2017, pp. 1–4.

[9] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, 1996, pp. 373–376 vol. 1.

[10] K. Tokuda, H. Zen, and A. W. Black, "An hmm-based speech synthesis system applied to english," in *IEEE Speech Synthesis Workshop*, 2002, pp. 227–230.

[11] "Spectrograms." [Online]. Available: https://ccrma.stanford.edu/~jos/mdft/Spectrograms.html

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning, chapter 9*. MIT Press, 2016, http://www.deeplearningbook.org.

[13] ——, *Deep Learning, chapter 10.* MIT Press, 2016, http://www.deeplearningbook.org.

[14] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2018, pp. 4779–4783.

[15] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[16] R. Mama, "Rayhane-mamah/tacotron-2."

[17] É. Székely, G. E. Henter, and J. Gustafson, "Casting to corpus: Segmenting and selecting spontaneous dialogue for TTS with a CNN-LSTM speaker-dependent breath detector," in *Proc. ICASSP*, 2019, pp. 6925–6929.

[18] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[19] Poetry - python packaging and dependency management made easy. [Online]. Available: https://python-poetry.org/

[20] Guild ai - experiment tracking, ml developer tools. [Online]. Available: https://guild.ai/

[21] Rclone - rclone syncs your files to cloud storage. [Online]. Available: https://rclone.org/

[22] Ffmpeg: A complete, cross-platform solution to record, convert and stream audio and video. [Online]. Available: https://www.ffmpeg.org/

[23] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi," 08 2017, pp. 498–502.

[24] J. Park, Kyubyong Kim, "g2pe," https://github.com/Kyubyong/g2p, 2019.

[25] Tacotron 2 - pretrained model. [Online]. Available: https://drive.google.com/file/d/1c5ZTuT7J08wLUoVZ2KkUs_VdZuJ86ZqA/view?usp=sharing

[26] K. Ito and L. Johnson, "The lj speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[27] Tacotron 2 - repository. [Online]. Available: https://github.com/NVIDIA/tacotron2/tree/185cd24e046cc1304b4f8e564734d2498c6e2e6f

[28] Waveglow on pytorch hub. [Online]. Available: https://pytorch.org/hub/nvidia_deeplearningexamples_waveglow/

[29] Avdelningen för tal, musik och hörsel. [Online]. Available: https://www.kth.se/sv/is/tmh/division-of-speech-music-and-hearing-1.780110