VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

DATA SCIENCE STUDY PROGRAMME

Master's thesis

Data Selection Strategies for Multi-Speaker Text-to-Speech Synthesis in Lithuanian

Work Title in Lithuanian

Aleksandr Jan Smoliakov

Supervisor	:	Gerda Ana Melnik-Leroy
Scientific advisor	:	pedagogical/scientific title Name Surname
Reviewer	:	pedagogical/scientific title Name Surname

Vilnius
2025

List of Figures

List of Tables

Contents

Introduction

The goal of creating machines that can speak like humans has captivated researchers for centuries. One of the earliest known attempts dates back to the 18th century, with Wolfgang von Kempelen's mechanical speech machine that utilized a bellows-driven lung and physical models of the tongue and lips.

Over the centuries, advancements in technology and understanding of human speech have driven significant progress in this field. Today's state-of-the-art systems, dominated by end-to-end (E2E) neural models, have achieved highly naturalistic speech with unprecedented acoustic quality. Notably, these end-to-end systems have unified the entire synthesis process into a single neural network, eliminating the need for complex multi-stage pipelines.

0.1 Background and motivation

0.2 Problem statement

Training high-quality TTS models typically requires large amounts of annotated speech data. The common recommendation is to use at least 10 hours of recorded speech from a single speaker to achieve good results.

Liepa 2 is a recently released Lithuanian speech corpus that contains 1000 hours of annotated speech; however, this data is distributed across more than 2600 speakers, with most speakers contributing only a few minutes of speech. The top speaker has around 2.5 hours of recorded speech.

Training a high-quality single-speaker TTS model on such limited data poses a challenge. Multi-speaker TTS models can utilize data from multiple speakers to improve performance. However, training on all available data is a time-consuming and computationally expensive process, especially in the context of a master's thesis.

Therefore, it makes sense to explore strategies for selecting smaller subsets of the available data for training. The question that arises is, what is the best way to sample multi-speaker data for training TTS models?

0.3 Research questions

This thesis aims to answer the following research questions:

- TODO

0.4 Objectives

0.5 Scope of the study

Scope: This study is exclusively focused on the Lithuanian language and the Liepa 2 speech corpus. It investigates a fixed total training data size of 30 hours. The models are limited to Tacotron 2 with DDC and FastPitch architectures within the Coqui TTS framework, using a pre-trained WaveGlow vocoder for waveform generation.

Limitations: The findings may not generalize to other languages, datasets with different characteristics, or other TTS architectures. The 30-hour training data size is a practical constraint and may not reflect performance at larger scales.

0.6 Thesis structure

1 Literature review

Before diving into the specifics of TTS systems and data selection strategies, it is important to establish the broader context of audio processing, text-to-speech synthesis, and its evolution over time.

1.1 Digital signal processing fundamentals

1.1.1 Physical properties of sound

In the physical world, sound is a continuous pressure wave that propagates through a medium, such as air. Key properties of sound waves include frequency (pitch), amplitude (loudness), and phase. Speech is also a complex sound signal composed of various frequencies and amplitudes that change over time.

The human audible range is typically considered to be from 20 Hz to 20 kHz, with most speech information concentrated between 100 Hz and 8 kHz.[13]

1.1.2 Digital representation of audio

As the real-world sound wave is continuous, it must be converted into a digital format for processing by computers. This involves two main steps, namely sampling and quantization.

Sampling is the process of measuring the amplitude of the sound wave at regular time intervals.

The rate at which these samples are taken is called the sampling rate (or sample frequency), measured in Hertz (Hz). According to the Nyquist-Shannon [14] sampling theorem, accurate reconstruction of a continuous signal requires a sampling rate that is at least twice the highest frequency present in the signal. In text-to-speech applications, common sampling rates for audio are 22.05 kHz and 24 kHz, which can capture frequencies up to approx. 11 kHz and 12 kHz, respectively.

Quantization (also known as bit depth) is the mapping of continuous amplitude values to discrete levels for digital representation. Typical bit depths for audio are 16-bit and 24-bit formats.

Pre-emphasis is a high-frequency filtering technique applied to audio signals before further processing. Natural speech signals tend to have more energy in the lower frequencies, with a gradual drop-off towards higher frequencies (typically around -6 dB per octave). Pre-emphasis compensates for this spectral tilt by boosting high frequencies using a first-order high-pass filter, which is defined as:

$$y[n] = x[n] - \alpha x[n-1] \tag{1}$$

where $y[n]$ is the pre-emphasized signal, $x[n]$ is the original signal, and α is the pre-emphasis coefficient (typically between 0.9 and 1.0).

This transformation balances the frequency spectrum, improving the signal-to-noise ratio for higher frequencies and preventing the model from optimizing only for low-frequency components.

1.2 Time-Frequency Analysis

1.2.1 Fourier Transform

Fourier Transform (FT) is a mathematical technique that transforms a time-domain signal (such as an audio waveform) into its frequency-domain representation. The signal is decomposed into a sum of sine and cosine waves at various frequencies, each with a specific amplitude and phase. This allows us to analyze the frequency content of the signal.

Short-Time Fourier Transform [12] (STFT) extends the FT by applying it to short, overlapping segments (frames) of the signal. This transformation provides a time-frequency representation, showing how the frequency content of the signal changes over time.

In TTS applications, the STFT is computed by dividing the audio signal into short frames (e.g., 25 ms) with a certain overlap (e.g., 10 ms) between frames.

1.2.2 Spectrogram

The spectrogram is a visual representation of the STFT, displaying frequency on the vertical axis, time on the horizontal axis, and amplitude represented by the color intensity.

1 figure. *Comparison of a linear spectrogram vs. a Mel-spectrogram.*

1.2.3 Mel-spectrograms

The human ear does not perceive frequencies linearly - it is more sensitive to lower frequencies than higher ones. To mimic this perceptual characteristic, the Mel scale [17] maps linear frequency f (in Hz) to a perceptual scale m (in Mels) using the following formula:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{2}$$

Mel-spectrograms are computed by applying a Mel filterbank of overlapping triangular filters (or kernels) to the magnitude spectrogram obtained from the STFT. This results in a compressed representation of the audio signal that aligns more closely with human auditory perception. Such Mel-spectrograms are commonly used as input features for modern TTS systems.

1.3 Text-to-speech synthesis

Text-to-Speech (TTS) synthesis, also known as speech synthesis, is the process of converting written text into human-like spoken words. Nowadays TTS is a key technology in numerous applications, including virtual assistants, accessibility tools, and language learning platforms.

1.3.1 Traditional TTS approaches

The first known attempts to create artificial speech date back to the late 18th century. These early devices were purely mechanical, relying on physical models to mimic the human vocal tract. One notable example is Wolfgang von Kempelen’s invention called the “Acoustic-Mechanical Speech Machine”, which utilized a bellows-driven lung and physical models of the tongue and lips to articulate both vowels and consonants. Such early efforts had limited success, being able to produce only individual phonemes or simple syllables.

The advancement of electricity and electronics in the late 19th and early 20th centuries opened new possibilities for speech synthesis. Homer Dudley’s Voder [8] (Voice Operation Demonstrator), unveiled in 1937, represented the first successful electronic recreation of human speech. This invention allowed a trained human operator to synthesize intelligible speech by manipulating a complex arrangement of keyboards and pedals, each controlling different acoustic parameters of speech production. The Voder marked a shift from mechanical to electronic synthesis methods.

In the decades that followed, two main approaches for speech synthesis emerged: concatenative synthesis and parametric synthesis.

1.3.2 Concatenative synthesis

The concatenative synthesis approach [5] synthesizes speech by piecing together pre-recorded segments of human speech. This method involves several steps. First, it requires pre-recording a large database of speech segments spoken by a human voice actor in pristine, highly controlled studio conditions to ensure consistent audio quality and minimize background noise. Each segment is labeled and indexed based on its phonetic and prosodic properties.

During synthesis, the system breaks down the input text into short linguistic units (such as phonemes or syllables) using a text analysis module. Then, it queries the speech database to find the best-matching segments for each unit using selection cost functions [2]. The retrieved segments are blended and concatenated to form a continuous speech waveform. The system then uses signal processing techniques to smooth the transitions between segments and adjust pitch and duration to match the desired output characteristics.

Concatenative synthesis can produce natural-sounding individual speech units, but the final audio often has noticeable glitches and breaks at the points where segments are joined together [2]. The segments may not blend smoothly due to differences in pitch, duration, and timbre. The prosody also tends to sound “choppy” and unnatural, since stringing disjointed segments together does not capture the natural rhythm and intonation patterns of connected speech.

Finally, concatenative synthesis requires language-specific expertise to design and maintain the underlying speech database and selection algorithms. This need for extensive data can make it challenging to develop concatenative TTS systems for low-resource languages or dialects.

1.3.3 Parametric synthesis

In contrast, statistical parametric speech synthesis [21] (SPSS) uses statistical models, typically Hidden Markov Models (HMMs) [18], to generate the parameters that control a speech waveform.

This method involves training a statistical model on a large corpus of recorded speech. The model learns the relationship between linguistic features (like phonemes and prosody) and the acoustic features of the speech signal, such as spectral envelope and fundamental frequency. During synthesis, the system takes text as input, converts it to a sequence of linguistic features, and then uses the trained model to generate a corresponding sequence of acoustic parameters.

Compared to concatenative synthesis, the statistical approach allows for more flexibility and control over the speech synthesis process, enabling the generation of a wider variety of voices and speaking styles. However, HMM-based synthesis [18] had a persistent problem: the statistical averaging built into the models tended to over-smooth the acoustic features, creating a characteristic "buzzy" or "muffled" sound that lacked the sharpness and detail of natural human speech.

1.4 Linguistic Representation (Text Processing)

1.4.1 Text normalization

Text normalization [16] is the process of converting raw text into a standardized format suitable for TTS synthesis. Typical steps include:

- Lowercasing: Converting all text to lowercase to reduce variability.
- Punctuation removal: Stripping out unnecessary punctuation marks.
- Expanding abbreviations: Converting abbreviations to their full forms (e.g., "Dr." -> "Doctor").
- Number normalization: Converting numbers into their spoken equivalents (e.g., "123" -> "one hundred twenty-three").
- Tokenization: Splitting text into words or subword units.

1.4.2 Graphemes vs. Phonemes

A key decision in TTS systems is the choice of input representation for text.

There are two main approaches, namely grapheme-based and phoneme-based. The grapheme-based approach uses the raw characters of the written text as input to the TTS model. This method is straightforward and does not require any additional pre-processing steps. However, it can struggle with languages that have complex orthographies or inconsistent spelling-to-sound correspondences.

In contrast, the phoneme-based approach uses a phonetic transcription of the text, typically in the International Phonetic Alphabet (IPA) format. The underlying assumption is that phonemes possess a more direct and consistent mapping to the acoustic features of speech, making it easier for the TTS model to learn the relationship between text and audio. However, this approach requires an additional grapheme-to-phoneme (G2P) conversion step [1], which requires a complex pre-processing pipeline or a separate G2P model.

There is yet another approach that combines both grapheme and phoneme inputs, known as accented grapheme-based synthesis. This method augments the raw text with accentuation markers (typically tilde, acute, grave accents) to indicate the stressed syllables in words. This additional information provides cues about the prosodic features of speech, which can help improve the pronunciation of speech.

1.5 Embeddings and Representation Learning

1.5.1 The Concept of Embeddings

1.5.2 Text Embeddings

1.5.3 Speaker Embeddings

In order to enable multi-speaker synthesis, TTS models need to receive a representation of the speaker's identity.

Lookup Tables (LUT): Early approaches used simple learnable embeddings where each speaker ID is mapped to a unique vector. The vectors are learned joinly with the TTS model during training. While efficient, this approach cannot generalize to speakers not seen during training.

Reference Encoders and Speaker Embeddings:

d-vectors: d-vectors [19] are fixed-length speaker embeddings derived from a separate speaker verification model. A reference encoder network takes a reference audio recording of arbitrary length and compresses it into a fixed-length vector known as a d-vector, that summarizes the speaker's timbral and prosodic characteristics. These d-vectors are then provided as additional input to the TTS model, and are kept fixed during TTS training.

x-vectors: An evolution of d-vectors, x-vectors [15] use a more Time Delay Neural Network (TDNN) architecture to capture the temporal context more effectively. These embeddings have shown an improved ability in zero-shot TTS scenarios.

One limitation of d-vectors and x-vectors is that if the reference audio is of poor quality or contains background noise, the resulting speaker embedding may not accurately represent the speaker's identity, leading to degraded synthesis quality.

1.6 Deep learning for TTS

The limitations of these complex, multi-stage pipelines opened the door for a new approach, the end-to-end (E2E) model. E2E systems learn the entire speech synthesis process–from input text directly to acoustic output–using a single neural network. This approach promised to eliminate the need for hand-crafted pipelines that were difficult to design, required extensive expertise, and suffered from errors that accumulated across multiple components. By learning directly from text-audio pairs, E2E models showed they could produce speech with higher naturalness and expressiveness than previous methods, representing a significant leap in TTS technology.

However, although deep learning TTS models are more robust to variations in data quality compared to concatenative approaches, they are essentially "data-hungry" beasts that require large amounts of training data to achieve optimal performance. This relationship between model performance and training data size follows well-established scaling laws for neural models [7]. This characteristic makes data selection and quality control critical factors in developing effective neural TTS systems.

1.6.1 Sequence-to-sequence models

[20]

The main architecture for modern E2E TTS is the sequence-to-sequence (seq2seq) framework, known through e.g. Google's Tacotron model. This framework has three main parts: an encoder, an attention mechanism, and a decoder. The encoder–usually a recurrent neural network (RNN) or a more complex module like the CBHG (Convolution Bank + Highway network + Gated Recurrent Unit)–processes the input text and converts it into a high-level representation. The attention mechanism acts as a bridge, learning to align the text representation with the output audio frames. At each step, it decides which part of the input text the model should focus on to generate the next piece of audio. The decoder, also typically an RNN, takes the encoder output and uses the attention context to generate an acoustic representation of the speech, one frame at a time.

1.6.2 Non-autoregressive (parallel) models

FastPitch [11] is a non-autoregressive TTS model.

1.6.3 Neural vocoders

The Inversion Problem: Standard TTS models like Tacotron or FastPitch generate intermediate acoustic features (typically mel-spectrograms), but do not directly produce raw audio waveforms. However, mel-spectrograms are lossy representations that only capture the magnitude of the frequency bands, discarding phase information. This information is discarded during the STFT process. Thus, converting a spectrogram into audio is a non-trivial task, as the phase information must be estimated. This challenge is known as the inversion problem.

The tool used to convert spectrograms back into waveforms is called a vocoder.

Traditionally, the Griffin-Lim algorithm [4] has been used to iteratively estimate and reconstruct the phase information from the magnitude spectrogram. However, this method often produces audio with noticeable artifacts and lower quality compared to natural speech.

GAN-based Vocoders: Neural GAN (Generative Adversarial Network) based vocoders, such as HiFi-GAN [9], have become the dominant approach for vocoding in modern TTS systems. These

GAN-based vocoder models consist of two main components: a Generator, which reconstructs mel-spectrograms into raw waveforms, and a Discriminator, which distinguishes between real and synthesized audio. This adversarial training forces the Generator to produce high-fidelity audio that closely resembles natural speech.

1.6.4 Notable End-to-End TTS models

1.7 Multi-speaker TTS

Multi-speaker TTS models are designed to synthesize speech in the voices of multiple speakers. As the name suggests, these models are trained on data from many different speakers, allowing them to learn the characteristics of each voice and synthesize speech that sounds like a specific individual, while still being able to generalize the shared linguistic and acoustic patterns across speakers.

1.7.1 Advantages

1.7.2 Techniques

Techniques for multi-speaker TTS often involve conditioning the model on speaker identity. This can be done by providing a speaker embedding vector as an additional input to the model. Early successful implementations of this approach include Deep Voice 2 [3], which demonstrated effective multi-speaker synthesis by learning speaker-specific embeddings.

The speaker embedding can be learned jointly with the TTS model during training, or it can be pre-trained using a separate speaker verification model.

1.7.3 Challenges

One key challenge in multi-speaker TTS is ensuring that the model can effectively capture the unique characteristics of each speaker's voice while still producing natural-sounding speech.

The distribution of training data across speakers can also influence the model's performance and its ability to mimic all voices equally well. The training datasets are often strongly imbalanced, with some speakers having hours of annotated speech, while others may have only a few minutes. This imbalance can cause the model to overfit to the speakers with more data, resulting in lower synthesis quality for underrepresented speakers.

Thus, data selection and sampling strategies are critical considerations when training multi-speaker TTS models.

1.8 Data selection in TTS

As TTS models require large amounts of annotated speech data for training, training on all available data can be computationally prohibitive and time-consuming. Therefore, data selection aims to identify optimal subsets of the available data that can yield high-quality synthesis while minimizing training time and resource requirements.

1.8.1 Data quantity

1.9 Evaluation methods for TTS

1.9.1 Subjective evaluation

TTS systems are often evaluated using subjective listening tests, where human listeners rate the naturalness and intelligibility of synthesized speech samples.

A common subjective evaluation method is the Mean Opinion Score [6] (MOS) test. In a MOS test, listeners are presented with a set of synthesized speech samples and are asked to rate each sample on a scale from 1 to 5. Here, the scale typically represents the following levels of quality:

- 1 - Bad: The speech is completely unnatural and unintelligible.
- 2 - Poor: The speech is mostly unnatural and difficult to understand.
- 3 - Fair: The speech is somewhat natural but has noticeable artifacts or issues.
- 4 - Good: The speech is mostly natural with minor artifacts that do not significantly affect intelligibility.
- 5 - Excellent: The speech is indistinguishable from natural human speech.

1.9.2 Objective evaluation

Objective evaluation methods use computational metrics to assess the quality of synthesized speech.

Common objective metrics include:

- Mel-Cepstral Distortion [10] (MCD): Measures the spectral distance between synthesized and natural speech.
- Fundamental Frequency Root Mean Square Error (F0 RMSE): Assesses the accuracy of pitch contours.

1.10 Summary and research gap

2 Methodology

2.1 Research design

The independent variables in this study are:

- Data selection strategy: Different methods for selecting subsets of the training data.
- TTS model architecture: Comparing different TTS architectures.

The dependent variables in this study are:

- TTS model performance: Measured using objective metrics (MCD, F0 RMSE) and subjective evaluations (MOS scores).

The controlled variables in this study are:

- Dataset: The Liepa 2 Lithuanian speech corpus.
- Training data size: The same total amount of training data used across all experiments.
- Training procedure: The same training hyperparameters and protocols applied across all experiments.
- Evaluation metrics: The same objective and subjective evaluation methods applied across all experiments.

2.2 Data and preprocessing

2.2.1 Liepa 2 dataset

The primary dataset for this research is the Liepa 2 corpus, a comprehensive collection of Lithuanian speech data. The corpus encompasses 1000 hours of short voice recordings from 2621 speakers, along with corresponding text transcriptions. The recordings span various speech styles and contexts, including read speech (audiobooks, news), TV and radio broadcasts, spontaneous speech.

More specifically, the dataset contains:

- Total duration: 1000 hours of speech data.
- Number of speakers: 2621 unique speakers.
- Number of utterances: 1,874,648 recorded utterances.

2.2.2 Acoustic preprocessing

The raw audio recordings from the Liepa 2 dataset are sampled at 16 kHz. To prepare the audio for TTS model training, the audio waveforms are resampled to 22,050 Hz. While resampling to a higher frequency does not add new information, it will be compatible with pre-trained vocoders that expect 22,050 Hz input.

Additional preprocessing steps, such as silence trimming and normalization, are performed by the Coqui TTS framework during training.

The acoustic processing that transforms audio waveforms into mel-spectrograms is performed on-the-fly during model training by the Coqui TTS framework. The mel-spectrogram parameters used are as follows:

- fft_size: 1024
- win_length: 1024
- hop_length: 256
- frame_length_ms: null
- frame_shift_ms: null
- stft_pad_mode: ”reflect”
- sample_rate: 22050
- resample: false
- preemphasis: 0.98
- ref_level_db: 20
- do_sound_norm: false
- log_func: ”np.log10”
- do_trim_silence: true
- trim_db: 60
- do_rms_norm: false
- db_level: null
- power: 1.5
- griffin_lim_iters: 60
- num_mels: 80
- mel_fmin: 0.0
- mel_fmax: 8000.0
- spec_gain: 20
- do_amp_to_db_linear: true
- do_amp_to_db_mel: true
- pitch_fmax: 640.0
- pitch_fmin: 1.0
- signal_norm: true

- min_level_db: -100
- symmetric_norm: true
- max_norm: 4.0
- clip_norm: true
- stats_path: null

2.2.3 Text normalization

Before feeding text data into TTS models, the text undergoes normalization to convert it into a more consistent and model-friendly format. The Liepa 2 text already includes a significant level of normalization - for instance, the following elements are written exactly as they were spoken: dates, times, acronyms, abbreviations, numbers.

However, some additional normalization steps are applied to further standardize the text:

- Punctuation standardization: Replacing uncommon punctuation marks with more common equivalents (e.g., replacing em dashes with hyphens, and semicolons with commas).
- Removal of extraneous characters: Eliminating any remaining characters that are not letters, digits, whitespace, or basic punctuation (.,-?!).
- Whitespace normalization: Collapsing multiple consecutive whitespace characters into a single space and trimming leading/trailing whitespace.
- Accent addition: Adding accent marks (tilde, acute, grave) to words using Kirčiuoklis tool (where Kirčiuoklis suggests multiple options, accent is not added).
- Lowercasing: Converting all text to lowercase to reduce vocabulary size.
- Letter replacements: Substituting non-Lithuanian letters with their Lithuanian equivalents (‘w’ with ‘v’, ‘q’ with ‘kv’, and ‘x’ with ‘ks’).

In order to add the accents to the text, the online tool Kirčiuoklis was queried with all unique words from the dataset. The tool returned the accented versions of the words, which were then used to replace the unaccented words in the text. In cases where Kirčiuoklis provided multiple accentuation options for a word, no accent was added to avoid introducing errors.

As a result of these normalization steps, the vocabulary size is reduced from 140 characters to 41 characters, simplifying the learning task for TTS models.

2.3 Data subset creation

To investigate the impact of data selection on TTS performance, several strategies for selecting subsets of the Liepa 2 dataset were employed:

- Random sampling: Randomly selecting a subset of the training data.
- Speaker diversity maximization: Selecting samples to maximize the number of unique speakers in the subset.

2.4 Model training

2.4.1 Tacotron 2 with DDC

The Tacotron 2 model was configured with the following hyperparameters:

2.4.2 FastPitch

The FastPitch model was configured with the following hyperparameters:

2.4.3 Computational hardware

The experiments were conducted on a personal high-performance computing setup with the following specifications:

- CPU: AMD Epyc 7642 48-Core, 96 thread processor
- RAM: 256 GB DDR4 3200 MHz
- GPU: NVIDIA GeForce RTX 3090 with 24 GB VRAM
- Storage: 2 TB NVMe SSD

2.5 Evaluation protocol

2.5.1 Test set

Evaluation was performed on a held-out test set of 100 phrases from the Liepa 2 dataset.

2.5.2 Objective evaluation

2.5.3 Subjective evaluation

Subjective evaluation was conducted using Mean Opinion Score (MOS) tests.

3 Results and analysis

3.1 Tacotron 2 with DDC

3.1.1 Objective results

3.1.2 Subjective results

3.1.3 Qualitative analysis

3.2 FastPitch

3.2.1 Objective results

3.2.2 Subjective results

3.2.3 Qualitative analysis

4 Conclusion

4.1 Summary of findings

4.2 Contributions

4.3 Limitations of the study

4.4 Future work

5 References

[1] M. Bisani, H. Ney. “Joint-sequence models for grapheme-to-phoneme conversion.” In: *Speech Communication* 50.5 (2008), pages 434–451.

[2] A. W. Black, N. Campbell. “Optimising selection of units from speech databases for concatenative synthesis.” In: *Eurospeech*. 1995.

[3] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, Y. Zhou. “Deep Voice 2: Multi-Speaker Neural Text-to-Speech.” In: *arXiv preprint arXiv:1705.08947* (2017).

[4] D. Griffin, J. Lim. “Signal estimation from modified short-time Fourier transform.” In: *IEEE Transactions on acoustics, speech, and signal processing* 32.2 (1984), pages 236–243.

[5] A. J. Hunt, 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*. Volume 1. IEEE. 1996, pages 373–376.

[6] ITU-T. *Methods for subjective determination of transmission quality*. Recommendation P.800. International Telecommunication Union, 1996.

[7] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, et al. “Scaling Laws for Neural Language Models.” In: *arXiv preprint arXiv:2001.08361* (2020).

[8] D. H. Klatt. “Review of text-to-speech conversion for English.” In: *The Journal of the Acoustical Society of America* 82.3 (1987), pages 737–793.

[9] J. Kong, J. Kim, J. Bae. “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis.” In: *NeurIPS*. 2020.

[10] R. Kubichek. “Mel-cepstral distance measure for objective speech quality assessment.” In: *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*. Volume 1. IEEE. 1993, pages 125–128.

[11] A. Łańcucki. “FastPitch: Parallel Text-to-speech with Pitch Prediction.” In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pages 6588–6592.

[12] A. V. Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[13] L. R. Rabiner, R. W. Schafer. *Theory and applications of digital speech processing*. Pearson, 2010.

[14] C. E. Shannon. “Communication in the presence of noise.” In: *Proceedings of the IRE* 37.1 (1949), pages 10–21.

[15] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, S. Khudanpur. “X-vectors: Robust DNN Embeddings for Speaker Recognition.” In: *ICASSP*. 2018.

[16] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, C. Richards. “Normalization of non-standard words.” In: *Computer Speech & Language* 15.3 (2001), pages 287–333.

[17] S. S. Stevens, J. Volkmann, E. B. Newman. “A scale for the measurement of the psychological magnitude pitch.” In: *The Journal of the Acoustical Society of America* 8.3 (1937), pages 185–190.

[18] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, K. Oura. “Speech synthesis based on hidden Markov models.” In: *Proceedings of the IEEE* 101.5 (2013), pages 1234–1252.

[19] E. Variani, X. Lei, E. McDermott, I. L. Moreno, J. Gonzalez-Dominguez. “Deep neural networks for small footprint text-dependent speaker verification.” In: *ICASSP*. 2014.

[20] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, et al. “Tacotron: Towards End-to-End Speech Synthesis.” In: *Interspeech*. 2017, pages 4006–4010.

[21] H. Zen, K. Tokuda, A. W. Black. “Statistical parametric speech synthesis.” In: *Speech communication* 51.11 (2009), pages 1039–1064.