VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
STUDY PROGRAM: INFORMATICS

Lithuanian Speech Synthesis Using Neural Networks

(Kalbą generuojantys neuroniniai tinklai lietuvių kalbai)

Master's Thesis

By: Arnas Radzevičius

VU email: arnas.radzevicius@mif.stud.vu.lt

Supervisor: prof. dr. Aistis Raudys

Reviewer: prof. dr. Olga Kurasova

Vilnius
2022

Summary

This master's thesis work proposes an approach to using stressed text instead of phonemes for TTS neural network inputs to solve the pronunciation problem of synthesized speech for higher-degree phonemic orthography languages. Tacotron 2 and VITS neural network architectures were used to train neural networks on multiple Lithuanian language datasets. Three single-speaker Lithuanian language speech corpora were collected to be used for the model training experiments, totaling 6, 27, and 92 hours of speech data, respectively. Finally, a survey is conducted to calculate MOS scores and evaluate each trained TTS neural network.

Furthermore, the initial experimental results of training a neural network-based accentuation model are detailed. The accentuation model is required as a pre-processing component for the TTS model to solve the synthesized speech pronunciation problem. The best-trained model achieves an accuracy (character-level) of 93%, but the model is not practical since it assigns stress marks to all the letters in the input sequence instead of assigning a single pitch accent for each word in the sequence.

The readers are provided a link to a website demonstrating the speech samples generated by the developed synthesizers. Also, the base pre-trained neural network models are provided in the links below.

Keywords: natural language processing, NLP, speech synthesis, text-to-speech, TTS, phonemic orthography, automatic text stressing, automatic accentuation, speech dataset, speech corpus, Tacotron 2, Waveglow, VITS

Santrauka

Šiame magistriniame darbe siūlomas metodas naudoti kirčiuotą tekstą vietoj fonemų TTS neuroninio tinklo įvestims, siekiant išspręsti sintezuotos kalbos tarimo problemą aukštesnio laipsnio foneminės ortografijos kalboms. Kalbos sintezės sistemoms paruošti naudojamos Tacotron 2 ir VITS neuroninių tinklų architektūros. Šie neuroninių tinklų modeliai apmokyti naudojant skirtingus vieno kalbėtojo duomenų rinkinius. Iš viso surinkti trys lietuvių kalbos duomenų rinkiniai, kuriuos atitinkamai sudaro 6, 27 ir 92 valandos vieno kalbėtojo balso įrašai. Rezultatams gauti atlikta apklausa, skirta apskaičiuoti MOS balus ir įvertinti kiekvieną apmokytą TTS neuroninį tinklą.

Darbe taip pat pateikiami pirminiai eksperimentų rezultatai apmokant neuroniniais tinklais grįstą kirčiavimo modelį. Kirčiavimo modelis reikalingas kaip pirminio teksto apdorojimo komponentas TTS modeliui, siekiant išspręsti sintezuotos kalbos tarimo problemą. Geriausiai parengto modelio tikslumas (simbolių lygmenyje) yra 93%, tačiau šis modelis nėra naudingas, nes jis priskiria kirčius visoms įvesties sekos raidėms, kai iš tiesų tik vienas kirtis turėtų būti priskirtas kiekvienam žodžiui įvesties sakinyje.

Skaitytojams pateikiama nuoroda į svetainę, kurioje demonstruojami sukurtų sintezatorių

generuojami kalbos pavyzdžiai. Be to, baziniai iš pre-treniruoti neuroninių tinklų modeliai pateikiami žemiau esančiose nuorodose.

Raktiniai žodžiai: kalbos sintezė, sintezatorius, TTS, automatinis kirčiuoklis, kirčiuoklis, kalbos duomenų rinkinys, Tacotron 2, Waveglow, VITS

Contents

Introduction

Speech synthesis is artificially generated human speech. Speech synthesis systems (also called speech synthesizers) generate speech either mechanically, or digitally (using computers). The most common speech synthesis systems are text-to-speech (TTS) systems that use computers to convert digital text into a speech waveform. TTS systems are used in virtual assistants [Hoy18], smart-home systems [NLS+11], speech synthesizers are an important tool for the disabled [IC01; SW12].

It is a complex one-to-many problem, where an input sequence (usually phonemes, less commonly - graphemes) is converted to a much larger sequence of speech features or even audio waveform samples. The main goal of speech synthesis systems is to produce an artificial speech that represents an actual human speech in terms of intelligibility, naturalness, expressiveness, etc. as much as possible. Historically, various approaches to speech synthesis systems were proposed. The very first machines emulating a human speech were produced as early as 1779 [Kra81], further inventions followed [KB91; Lem+99]. These systems could only produce a few human-like sounds resembling vowels or consonants. Currently, computer-based text-to-speech systems are capable of generating speech that is difficult to distinguish from real human speech. For a long time, concatenative and statistical parametric synthesis systems reigned as the best performing synthesizers. In recent years, neural network-based approaches to speech synthesis outperformed the traditional concatenative and statistical parametric methods. Not only that the synthesized speech sound more natural compared to traditional systems, but the training of the neural networks is also easier, eliminating the need for expert domain knowledge and extensive manual engineering during development since neural networks use data to learn to generate speech. The downside is that neural networks still require a large amount of speech data to train a robust TTS model. This is a major problem for languages that lack openly available speech data and tools for natural language processing. Such languages are commonly referred to as *low-resource* languages.

Most of the research on neural-network-based TTS systems is conducted on English language synthesizers. Because of that, a lot of natural language processing tools (text pre-processing, grapheme-to-phoneme, speech alignment, etc.), speech datasets, and other resources are publicly available for the English language. Thus, reproducing the results of academic research is less challenging for such a high-resource language. But that may not be the case for low speech-resource languages. The biggest issue is that there are no publicly available datasets that are large enough for TTS neural network training task. Besides that, tools like grapheme-to-phoneme converters that are essential for tackling the pronunciation and other synthesis problems are not publicly available for low-resource languages. Therefore it is still not trivial to build neural network-based speech synthesis systems for low-resource languages.

Because of this, other approaches may be required to train a speech synthesis neural network for low-resource languages. This work focuses on low-resource languages which are higher-degree phonemic orthographies. For phonemic orthographies, graphemes (written symbols) correspond to phonemes (spoken sounds), meaning that the graphemes are spoken pretty much the same way

they are written. This fact is leveraged to argue that phoneme inputs to TTS neural network models may be replaced by graphemes for phonemic orthographies. In reality, there are only a few actual phonemic orthographies and more slightly defective phonemic orthographies (languages with a higher degree of phonemic orthography). Most of the time, some graphemes in such languages are not fully distinguished. By introducing accents, more information like vowel length (duration) is provided. Therefore, in this work, a method is proposed to use stressed (accentuated) text instead of a raw sequence of graphemes as model inputs to solve the pronunciation problem for higher-degree of phonemic-orthography languages. Such a language is Lithuanian, which is used throughout the paper to prove the claim.

During this master's thesis work, three Lithuanian language speech corpora were collected, totaling 6, 27, and 92 hours of speech data, respectively. The datasets were used to train TTS neural network models based on two architectures: Tacotron 2 and VITS. As mentioned above, many academic papers claim to use phonemes as TTS model inputs to solve or at least ease the pronunciation problem common for speech synthesizers. In this work, it is argued that phoneme input representations are not necessary. A hypothesis is raised that using accentuated text as TTS model inputs improves the pronunciation of speech generated by TTS neural networks for higher-degree phonemic orthography languages, like Lithuanian. Multiple TTS neural network models are trained on stressed (accentuated) and non-stressed text instead of phonemes to prove the claim. A survey was conducted to calculate the mean opinion scores (a subjective TTS systems evaluation metric) of each trained model to evaluate the trained models. The experimental results show that stressed text inputs significantly improve the performance of neural network-based speech synthesizers and therefore prove that different approaches may be suitable to build a speech synthesis system for low-resource languages. In addition, the model that scored the highest MOS developed during this work is compared with other known Lithuanian speech synthesizers in a supplementary MOS survey. Furthermore, since it is tedious to accentuate the model input text manually, a solution to build an automatic stressing model based on neural networks is explored. Initial experiments and future work on creating such a model are detailed in this work.

1. Literature Overview

An overview of the academic literature is necessary to explore the background of the researched field and related work. Also, it may prove useful to review the possible neural network TTS architectures and consider the potential candidates for training a Lithuanian speech synthesis model. The subject area is actively developed, so it is necessary to stay updated on the latest works and achievements. The knowledge may be used to help build a state-of-the-art text-to-speech neural network in Lithuanian. So, in this section:

- The traditional and neural network-based TTS systems are reviewed
- An introduction to the concept of phonemic orthography is given

- A background on automatic text stressing systems is provided
- The Lithuanian language synthesizers are overviewed
- Open-source text-to-speech datasets are described
- Methods for evaluating TTS systems are discussed

1.1. Text-to-speech Systems

Text-to-speech systems (TTS), also called speech synthesis systems, generate artificial human speech given a text sequence input. It is a large-scale inverse problem: a highly compressed source (text) is decompressed into audio waveform samples [WSS+17]. It is also a one-to-many problem: the same text can correspond to a different pitch, energy, phoneme duration, and other speech properties, i.e. multiple speech variations correspond to the same text [RHQ+20]. The task is actively researched for many years and has many practical uses: in virtual assistants, smart homes, voicing audio books, movies, games, auxiliary devices for the disabled, etc. The most popular traditional approaches to solving the speech synthesis task are concatenative synthesis and statistical parametric synthesis. In recent years, advances in the field of deep neural networks gave rise to state-of-the-art neural network-based TTS systems, outperforming the traditional methods both in naturalness, intelligibility, expressiveness, and other speech features. In addition, the process of developing a TTS system for a new language has become easier. But, even though neural networks improved the performance of speech synthesis systems in many areas, there are still a lot of challenges to be tackled.

First of all, a lot of audio and text pair data is needed to train a TTS neural network. Also, a large amount of computational resources is required to train neural network models using these large datasets. Furthermore, the TTS systems still depend on various third-party tools and algorithms and may use them as architectural components, preventing the systems from being fully end-to-end neural networks, and introducing dependency problems. Often, the aforementioned tools are required for input text pre-processing.

Having said all that, research on speech synthesis is done mainly for the English language, which is considered a high-resource language. Such high-resource languages have a number of tools and huge amounts of speech data that are publicly available. But that is not the case for low-resource languages like Lithuanian. So, different neural network training approaches may be needed to build neural network TTS systems for low-resource languages.

1.1.1. Traditional TTS Systems

Historically, the first attempts at speech synthesis date back to the late 18th century. The very first speech synthesis systems were mechanical ones, meaning the sound was generated by using physical machines [KB91; Kra81; Lem+99]. Unsurprisingly, such systems could only synthesize

very basic sounds like phonemes or, at best, individual syllables, words, or a specific sentence. The introduction of computers led to many new approaches to speech synthesis. The most notable traditional TTS systems are concatenative and statistical parametric synthesis systems. Before the introduction of neural networks into the field, for decades the best-performing TTS systems were developed using these two methods. In the following subsections, the concatenative and statistical parametric methods are reviewed.

1.1.1.1. Concatenative synthesis

The concatenative approach to speech synthesis has been popular for many years [HB96; SBV+06; ZLD21] before being outperformed by neural network-based approaches. In essence, the concatenative synthesis systems workflow is the following: given a text sequence input, short audio segments corresponding to the units of the input text are selected from a big database of speech samples and are concatenated together (hence the name concatenative synthesis) to form a final synthesized speech output [HB96]. To be able to select the appropriate audio segments given a text, a concatenative synthesis system may execute a linguistic, morphological, or other text analysis to extract linguistic features, by which an appropriate audio segment can be assigned to a particular grapheme, phoneme, syllable, word, etc. State-of-the-art concatenative systems produce high-quality synthesized speech really fast, hence they are suitable for production use. But this approach has its downsides [KKW18]. As may be guessed, the concatenation of audio segments extracted from different sentences that may have a different pitch, duration, energy, and other speech properties produces notable speech artifacts. This is especially true when observing the concatenation points of the concatenated speech waveform. This problem is partly solved by increasing the size of the speech database, using speech segments that have speech properties as monotonic (similar) as possible, and using different techniques for cost functions and weights. Furthermore, the development of a concatenative speech synthesis system requires a lot of manual engineering and expertise in the field of natural language which is highly expensive.

1.1.1.2. Statistical parametric synthesis

Another approach to building speech synthesis systems is using the statistical parametric synthesis methods. Simply put, such methods work by averaging some sets of similarly sounding speech segments. First, the parametric representations of speech are extracted from a speech database and then modeled by using a set of generative models like hidden Markov models (HMMs) [TNT+13]. A maximum-likelihood criterion is usually used to estimate the model parameters. The parametric representations of speech are later used to reconstruct a speech waveform. Building the HMM-based statistical parametric synthesis system consists of training and inference (synthesis) stages [ZTB09]. During the training stage, the maximum likelihood is estimated, and the synthesis is performed by generating speech parameters and maximizing the output probabilities.

The main advantage of statistical parametric synthesis over the concatenative approaches is its

flexibility in changing its voice characteristics, speaking styles, and emotions. Four main techniques are used to achieve this: adaptation (obtain speaker-specific speech features with a small amount of speech data [MTK+97]), interpolation (mixing voices - enables synthesis with voice characteristics that the model was not trained on [IS95], eigenvoice (producing voices), and multiple regressions (controlling voices)

When comparing the best examples of both traditional TTS systems, concatenative synthesis systems outperform their parametric statistical counterparts in terms of speech quality and naturalness. On the other hand, parametric approaches offer other previously mentioned benefits, like controllability.

1.1.2. Neural network-based TTS systems

There were early attempts to introduce neural networks to the field of speech synthesis [FM06; Vai+01], but the speech quality produced by such systems did not manage to outperform the speech generated by traditional approaches at the time, so the methods did not grow popular. But, in the mid-2010s, advances in the field of neural networks and easier access to more computational resources (especially the widespread use of GPUs for neural network training) led to the development of multiple TTS neural network architectures which surpassed the traditional approaches. In this section, the literature on the most notable neural network-based TTS systems is reviewed. But before moving to the said systems, a brief overview of relevant concepts typical for neural speech synthesis systems architectures will be reviewed, concretely - acoustic and vocoder models, autoregressive and non-autoregressive generation. Other neural network concepts like seq2seq, attention, Transformers, etc. are not overviewed to avoid an over-detailed literature review. Also, note that only the papers that have an open-source implementation available online are reviewed in this section. Other papers, in the author's opinion, are not worth investigating for practical reasons - the implementation of the systems would take too much time.

1.1.2.1. Acoustic and vocoder models

For traditional speech synthesis systems, the usual approach to synthesize speech is to first develop a so-called front-end that can extract various linguistic features from the text, provide a duration model, an acoustic feature prediction model, and a complex signal-processing-based vocoder [Ito17]. By introducing neural networks into the field, it became possible to replace some of these components using a neural approach. Nevertheless, a two-stage flow is still widely used in neural network-based TTS systems. Specifically, such a system is composed of acoustic and vocoder models. An acoustic model takes text as input and generates acoustic features as outputs. The most popular representation of such features is spectrograms and Mel-spectrograms. So, an acoustic model in neural network TTS systems typically performs a text-to-spectrogram conversion. Next, the outputs of the acoustic model (acoustic features) are fed to a vocoder as input which then converts the features to a speech waveform. In other words, the vocoder performs a spectrogram-to-waveform

conversion.

Until quite recently such a two-step approach produced the best quality speech synthesizers. In spite of that, it has its flaws [KKS21]:

- Training and inference pipelines can get complicated
- Both models need to be trained separately, so training time is increased
- Inference takes a longer time
- Such a system is not a fully end-to-end neural network, as it is composed of several components

Recently, fully end-to-end neural TTS architectures were proposed. They are described in further sections.

1.1.2.2. Autoregressive and non-autoregressive networks

Text-to-speech is a one-to-many problem, where a highly-compressed data representation such as characters, graphemes, or phonemes is used to generate a Mel-spectrogram (speech features) or an audio waveform (sequence of audio samples). One second of audio can contain 22 thousand samples and more, thus a lot of computational resources are required for a neural network model to predict such a sequence sample-by-sample. Neural network models that are conditioned on previously generated outputs in a sequence are called autoregressive models [Aka98]. It is hard to parallelize the computations for such models since the previously generated outputs are needed at each time step. On the other hand, non-autoregressive models are conditioned on the whole input sequence [RLT+20]. Therefore, it is easier to parallelize such systems. As a consequence, the training and inference times can be greatly reduced. Also, since the generation is conditioned on the whole input, a larger receptive field and context is available for the model [KKS21], thus allowing to make use of more information on pitch, energy, and other features and generate a more expressive, variable, and tunable speech.

1.1.2.3. Neural network architectures

Systems like DeepSpeech [HCC+14] imitate the flow of traditional TTS approaches, where multiple components are combined to work as a front-end by extracting various linguistic features, providing a duration model, an acoustic feature prediction model, and a complex signal-processing-based vocoder [Ito17]. DeepSpeech simply replaces the rule-based traditional TTS components with neural network models. This approach provides fair results, but the training and inference flow is too complicated because each system component needs to be trained separately. Also, due to the fact that the system consists of multiple independent components, errors easily propagate during inference.

The introduction of Tacotron [Ito17] had a huge impact on the field of speech synthesis. The system pioneered a fully end-to-end neural approach to TTS systems. Tacotron is an autoregressive end-to-end generative TTS model based on recurrent neural networks (specifically - gated recurrent units, or GRU [CVG+14]) and a modified sequence-to-sequence (seq2seq) architecture [SVL14] with attention [BCB14] paradigm. Tacotron can learn to synthesize speech at a character level, meaning it does not require tools for converting text sequences to phonemes or similar representations. This relaxed the pronunciation problem for the network. Trained on a dataset of text and audio pairs (speech corpus), during inference Tacotron can output a raw spectrogram given a character sequence as input. Still, even though at the time it outperformed the state-of-the-art production parametric systems, it performed slightly worse than concatenative synthesis systems. This is mainly due to the use of the Griffin-Lim [GL84] algorithm as a vocoder, that is known to synthesize poor quality speech from spectrograms. Also, since not a lot of text data is provided during training, the network cannot learn all the exceptions in pronunciation and suffers from occasional mispronunciation errors. Besides, the model occasionally produces errors in prosody such as word-skipping, generating silence or unintelligible speech, etc.

At the time the Tacotron paper was released, the authors noted that the use of the Griffin-Lim algorithm as a vocoder was only a placeholder for the future development of a fast and high-quality trainable spectrogram-to-waveform inverter. In the same year, the authors of Tacotron released a sequel to the article - Tacotron 2 [SPW+18], that solves the aforementioned problem. The authors use a similar feature-prediction architecture for generating spectrograms from text, instead this time they chose to predict Mel-spectrograms that represent the human receptive field better and are proven to provide better synthesis results. For the vocoder part, Tacotron 2 uses a modified WaveNet [ODZ+16]. The WaveNet vocoder is modified to use Mel-spectrograms as inputs instead of conditioning it on various extracted linguistic features to generate a speech waveform. This modification resulted in a state-of-the-art end-to-end TTS neural network system that outperformed the best concatenative and parametric synthesis systems of the time. Because of how easy it is to train Tacotron 2 and its resulting synthesis quality, the system is still popular and widely used even today. Overall, Tacotron 2 is a high-performance speech synthesis system, but it has its flaws. The most notable ones are that it suffers from occasional mispronunciation and word skipping or repeating. Also, because of its two-component pipeline - acoustic and vocoder models - it requires a lot of time to train while the inference speed is slow. Besides, compared to traditional TTS systems, it offers a relatively slow inference speed because the WaveNet vocoder predicts each sample one by one. NVIDIA introduced its own implementation of Tacotron 2 [1] by replacing the vocoder in the original paper with WaveGlow, but the system is still relatively slow and especially struggles with longer sequences since inference time increases linearly as the input sequence increases.

After the introduction of Transformer networks [VSP+17], a first attempt was made to incorporate the method in TTS systems by Microsoft Research [LLL+19]. Authors replaced the RRNs used in Tacotron architecture to tackle the vanilla system's problems of low efficiency during training

[1] https://github.com/NVIDIA/tacotron2

and inference, and modeling long dependencies. The resulting system can be trained faster, but just slightly improves on inference speed, while retaining a similar synthesized speech quality. Besides, the authors use phoneme sequences as inputs to the model to solve pronunciation problem, but the grapheme-to-phoneme conversion tools are not available for low-resource languages like Lithuanian, so the system needs to be adapted to work for such languages.

The problems that the Tacotron model face can be observed in most autoregressive TTS systems: slow inference, non-robust synthesis (word skipping and repeating), and lack of controllability. Further works like Parallel WaveNet [OLB+18], ClariNet [PPC18], and WaveGlow [PVC19] solves these issues, but they still need to be conditioned on Mel-spectrogram features that are generated autoregressively. In other words, they act only as vocoders. Microsoft introduced one of the first non-autoregressive TTS systems - FastSpeech [RRT+19]. The authors of the paper claim to solve the aforementioned problems by generating Mel-spectrograms from phonemes in parallel by utilizing a feed-forward Transformer network, phoneme duration predictor, and length regulator. At a similar time, another TTS system that generates Mel-spectrograms in a non-autoregressive manner was introduced [PPS+19], but it is based on an encoder-decoder framework with attention mechanism, therefore requires more parameters and time for inference. Furthermore, it does not solve the problem of word skipping and repeating.

Even though FastSpeech was a big improvement to the speech synthesis systems of the time since it operates in a non-autoregressive manner, it can only generate acoustic features non-autoregressively and needs an external vocoder to synthesize a speech waveform (the authors of the original paper use WaveGlow), thus the system is not fully end-to-end. Further work from the same authors produced FastSpeech 2s [RHQ+20] - the first attempt at a fully end-to-end phoneme-to-waveform generating system. The main challenges of text-to-waveform generation specified in the paper are a large information gap between input and output, and difficulties in training due to long waveform samples and limited GPU memory. The authors solve these problems by employing adversarial training, a Mel-spectrogram decoder based on WaveNet and Parallel WaveGAN [YSK20], and special waveform decoder losses: multi-resolution STFT loss and LSGAN discriminator loss. Again, FastSpeech 2 requires third-party tools to produce phoneme-to-audio alignments, thus is too complex to implement for low-resource languages.

Glow-TTS [KKK+20] eliminates the need for an external phoneme-audio aligner. Inspired by the fact that a human reads out text in order, without skipping any words, it is a flow-based non-autoregressive generative model for parallel TTS that can internally learn its own alignment by utilizing an efficient monotonic alignment search implemented with a dynamic programming algorithm. Its architecture is composed of an encoder that follows Transformer TTS [LLL+19] structure, a flow-based decoder that is partly based on WaveGlow [PVC19], and a duration predictor similar to FastSpeech [RRT+19]. Glow-TTS synthesis is fast, high-quality, and controllable, but acts only as an acoustic model to synthesize Mel-spectrograms given an input phoneme sequence, thus it requires an external vocoder to generate audio speech (authors use WaveGlow).

Most of the aforementioned systems are two-stage acoustic-vocoder model systems. Other

attempts at a fully end-to-end text-to-waveform neural speech synthesis system (e.g. FastSpeech 2s [RHQ+20], EATS [DDB+20], and Wave Tacotron [WSB+21]) reduce the training complexity and significantly decrease the synthesis time, but lags behind the two-stage systems in terms of speech quality. To solve this issue, the creators of Glow-TTS introduced VITS (Variational Inference with adversarial learning for end-to-end Text-to-Speech) [KKS21] - a parallel fully end-to-end TTS method that generates more natural-sounding audio than the two-stage models of the time. The authors of the paper achieve this by utilizing the variational auto-encoder (VAE), monotonic-alignment search (MAS), and adversarial training. VITS architecture consists of a prior encoder, posterior encoder, decoder, discriminator, and stochastic duration predictor. Originally, the authors of the paper used phoneme sequences as inputs to the model, but since the phoneme-audio alignment is learned internally and does not require external tools, any text representation can be used as input to the model.

1.2. Phonemic Orthography

A *phonemic orthography* is an orthography (system for writing a language) in which the graphemes (written symbols) correspond to the phonemes (spoken sounds) [Sga87]. A true phonemic orthography is one where each phoneme would invariably be represented by its corresponding grapheme. In such a case, the spelling of a word would unambiguously and transparently indicate its pronunciation. On the other hand, by knowing a pronunciation of a word its spelling can be inferred confidently. In reality, there are few natural languages that are true phonemic orthographies and relatively more defective or slightly defective orthographies. A defective orthography is one that is not capable of representing all the phonemes or phonemic distinctions through graphemes. An example of such language is English, e.g. a grapheme *th* can be pronounced different in each of the following words: *this*, *thin*, *goatherd*. A language may also be a slightly defective orthography (otherwise called a higher-degree phonemic orthography) if its vowels, tone and vowel length, vowel phonation, etc. are not fully distinguished. An example of such a language is Lithuanian. It is a slightly defective orthography as it does not fully distinguish tone and vowel length and is classified as a pitch-accent language.

Previous sections show that most of the research in the field of text-to-speech neural network systems is conducted using English as the target language. Since English is highly non-phonemic, the systems are often built to generate speech from phonemes and not from graphemes. This way, it is easier for the neural network to learn to pronounce the words correctly, instead of relying on the network to learn all the pronunciation exceptions of the graphemes. Third-party tools or pronunciation dictionaries are usually used to convert the text to phonemes that are fed into a network. There were attempts to use raw letters as inputs to the TTS neural network [Ito17], but the authors of the work note that pronunciation problems arise since usually there is not enough data for the model to learn all the pronunciation exceptions in a language. Anyway, grapheme-to-phoneme tools and pronunciation dictionaries are not available for most low-resource languages (including Lithua-

nian). Thus, different pronunciation problem solutions may be required for such languages to build a quality speech synthesizer. One approach to solving the problem is to use stressed text as model inputs instead of phonemes as it would help the model distinguish between the length and tone of the phonemes for the corresponding graphemes. It proved to be a good solution for languages with a higher degree of phonemic orthography [RRK21]. In the latter paper, the TTS neural networks trained on Lithuanian language datasets with and without stressed text labels are compared. The MOS survey showed that using pitch accents to train TTS neural networks for languages with higher-degree of phonemic orthography improves the naturalness of the system considerably. On the other hand, the results of the paper may seem unconvincing as only one neural network architecture was trained on a single dataset. In this work, the problem is explored in more detail.

1.3. Automatic Text Stressing Systems

Automatic text stressing (also commonly referred to as *automatic accentuation*, *automatic stress assignment*, *stress prediction*, *word accentuation prediction*) task is responsible for assigning a correct pitch accent for each word in a given input sentence. Some early works on such systems note that an automatic text stressing system would be a valuable component to a TTS system as a text pre-processor [Wil87]. Automatic accentuation systems were developed using multiple approaches. Algorithmic-approach systems use morphological and phonological analysis to stress words in a sentence [Chu86; Wil87]. Multiple machine learning algorithms were used to solve the problem: decision trees [Anb10; Šef06], support vector machines (SVMs) [DBJ+09], maximum entropy ranking [HS13]. Notably, not much research was done on automatic accentuation using neural networks. One early work uses 3 layers of feed-forward neural networks to classify whether a word is accented or not [MPvS07], but it achieves a poor accuracy of 84% (only slightly better than chance as measured by ROC analysis). Another work uses GRUs to predict word accents and achieves a 94.4% accuracy [Mos20]. For now, machine learning approaches seemed to offer the best results for text stressing with up to 98% accuracy [DBJ+09].

There were multiple attempts at an automatic Lithuanian language text stressing system. A work by Pijus Kasparaitis proposed a rule-based approach to stress words using a dictionary [Kas00; Kas01b]. Another work utilized decision trees to achieve 95.5% accuracy for the task [Anb10]. There is also an online tool developed by VDU researchers [2], but no paper providing details of the work is published.

1.4. Lithuanian Language Synthesizers

This work describes the development of a Lithuanian language speech synthesizer, so it is relevant to review the previous work done for such systems. While English language synthesizers were developed for decades [HB96; TYM+00], a major breakthrough in Lithuanian synthesizers occurred

[2] http://donelaitis.vdu.lt/main.php?id=4&nr=9_1

in 2013-2015. A production-level synthesizer followed in 2021. These synthesizers are briefly reviewed in the following paragraphs.

There were early works done by foreign companies (in some cases - in collaboration with foreign companies). The very first Lithuanian language speech synthesizer - Apollo - was introduced in 1994 by UK company *Dolphin Systems Inc* [Kas+16]. It was based on a formant synthesis method [Kla80]. Aistis synthesizer followed in 1996 [Kas01a]. This time, the system was based on concatenative synthesis and utilized the accentuation and transcribing modules. The last two synthesizers are no longer used.

In 2003, a Chech company RosaSOFT developed a multispeaker synthesizer WinTalker based on the concatenative synthesis, which included 3 Lithuanian voices: Gintaras, Aistis, and Aistis 2 [Kas+16]. Some of the algorithmic solutions used to develop the system are not open to the public. In the case of synthesizer Aistis 2, different synthesizers were developed using different approaches to determine the sound duration and tone modifications using the method employed by RasaSOFT in WinTalker, MBROLA algorithm [DPP+96], and TD-PSOLA algorithm [MC90].

In 2008, the Egidius synthesizer was developed by the Lithuanian and Belarussian companies UAB Etalinkas and Sakrament respectively. Again, it is based on the concatenative synthesis, but here it uses different approaches to text stressing [Kas+16].

Synthesizer SINT.AS was developed by dr. prof. Pijus Kasparaitis and UAB ”Algoritmų sistemos”, and deployed in production in 2013 [Kas+16]. It introduced two artificial voices: Marijus (male) and Laima (female). Compared to previous Lithuanian synthesizers, SINT.AS is distinct in that the narrators were specially selected to make the speech corpus, and that the unit-selection method was used for synthesis. Neither the details about the work nor the synthesizer itself is available publicly.

The project LIEPA [3] introduced a state-of-the-art Lithuanian speech synthesizer of the time [Kas+16]. Developed in 2013-2015 by Vilnius University and its partners, it introduced 4 new voices: Aistė, Edvardas, Regina, and Vladas. It uses the unit selection method and is publicly available.

In 2021, VDU scientists collaborating with the LRT team developed a production-quality Lithuanian speech synthesizer [Bla]. It is the first Lithuanian speech synthesizer deployed in production that is based on neural networks. The synthesizer is deployed on the LRT website [4] to narrate the published articles. The synthesized speech is very natural and comprehensible. Neither the systems nor the details on how it was developed are open publicly, besides the fact that the TTS system is based on neural networks [5].

Lastly, Microsoft’s Azure provides a service of Lithuanian speech synthesis in two voices - Leonas and One (male and female, respectively) footnote https://azure.microsoft.com/en-us/services/cognitive-services/text- to-speech. The synthesizer is also based on neural networks

[3] https://raštija.lt/liepa/paslaugos-vartotojams/sintezatorius-akliesiems/

[4] https://www.lrt.lt/

[5] https://www.lrt.lt/naujienos/tavo-lrt/15/1340747/nuo-siol-portale-lrt-lt-visus-straipsnius-galima-ne-tik-skaityti-bet-ir-klausyti

and is built in a low-resource setting using LR-UNI-TTS - Azure's neural TTS production pipeline footnote https://techcommunity.microsoft.com/t5/ai-cognitive-services-blog / neural-text-to-speech-previews-five-new-languages-with / ba-p / 1907604. The speech produced by the synthesizer

The reviewed literature suggests that until quite recently, most Lithuanian language synthesizers were based on concatenative methods. As observed in the previously reviewed papers, currently these methods are outperformed by neural network-based approaches to speech synthesis. The recently created Lithuanian TTS systems based on neural networks - VDU and Microsoft's synthesizers - perform rather well, but no academic article was published on how the systems were built and therefore these details are not publicly known.

1.5. Text-to-speech Datasets

The development of any TTS system - whether it is concatenative, parametric, or neural network-based - usually requires a collection of a speech corpus. While the data may be more complex for the concatenative or statistical parametric approaches as it may require more complex annotations, TTS neural network systems can usually learn from text and the corresponding speech pairs data. In the context of speech synthesis neural network models, the text is an input to the model, and the audio waveform is the target. Datasets for text-to-speech neural network training may contain audio samples voiced by a single or multiple speakers. Such datasets are henceforth called single-speaker and multi-speaker datasets, respectively. For different neural network architectures, different amounts of speech data may be required to achieve satisfactory results. But it is common to see the use of the same publicly available speech datasets in different articles. Usually, those are English language speech corpora. This is done because it is much easier to compare the performance of the developed methods if different networks are trained on the same data. One such popular open-source single-speaker dataset for the English language is LJSpeech [6]. It contains nearly 25 hours of speech data and is used in many papers where state-of-the-art neural network architectures are introduced [Ito17; KKS21; RRT+19].

There are a few popular multi-speaker datasets for TTS systems. These datasets can differ in duration, original sample rate, and number of speakers. One such popular multi-speaker dataset is VCTK [7]. It contains 44 hours of speech data spoken by 109 speakers. A much larger multi-speaker dataset is Common Voice [8] [ABD+19], containing 1118 hours of audio recordings, narrated by 51072 speakers. It also contains data for multiple languages. Both these datasets are available at a 48 kHz sample rate, so it is possible to downsample the recordings to suit specific use cases. Other notable datasets are LibriTTS [9] [ZDC+19] (586 hours, 2456 speakers, 24 kHz sample rate), and

[6] https://keithito.com/LJ-Speech-Dataset/
[7] https://datashare.ed.ac.uk/handle/10283/3443
[8] https://commonvoice.mozilla.org/en/datasets
[9] http://www.openslr.org/60/

LibriSpeech [10] [PCP+15] (982 hours, 2966 speakers, 16 kHz sample rate).

All the mentioned datasets are in the English language. There are not many open-source data available for the Lithuanian language. Although not entirely. Common Voice dataset offers 16 hours of validated labeled Lithuanian speech data. But the speech data is distributed between 232 speakers, therefore it is not sufficient for a single-speaker TTS system training. Project LIEPA [LTK+18] offers 100 hours of speech data designed for speech recognition systems, and 13 hours of speech data distributed between 4 speakers for text-to-speech systems. There are more papers describing the publicly available Lithuanian speech corpora [DK09; RR02], but at the time this paper is written the links to the published datasets do not seem to work and may be outdated.

While the aforementioned datasets contain unprocessed text labels for the corresponding audio samples, it is common to pre-process the text to make it easier for the neural network to distinguish between all the pronunciation intricacies. For neural network-based systems, it is common to use the phoneme representation for the model so there is no ambiguity for the model as to how to pronounce each input unit [KKK+20; KKS21; RRT+19]. Third-party grapheme-to-phoneme tools are used to convert the grapheme (letter) sequence to a phoneme sequence. It is also not uncommon for some speech synthesis systems to make use of third-party tools to extract the duration information from the phoneme and audio pairs [RRT+19]. Other systems do that internally [KKK+20; KKS21].

1.6. Evaluation of TTS Systems

There are multiple objective and subjective metrics aimed at evaluating TTS systems. Objective metrics are based on various algorithms to automatically evaluate a speech synthesis system and provide a numeric value of its quality in terms of different aspects. Such a metric would be a preferred choice when building TTS systems because the quality could be evaluated objectively and quickly. However, in practice, such metrics are rarely applied. There are multiple reasons for that. First, most of the objective evaluation metrics focus only on evaluating specific features of synthesized speech, like intelligibility [URB+15; WWT+12]. Such evaluation would not represent real-world scenarios where people may prefer one speech over others depending on multiple aspects of speech like voice tone, expressiveness, and most importantly - naturalness, i.e., how much the speech resembles an actual human speech. Secondly, most objective evaluation systems are not publicly available or are hard to adjust for a specific use case. That may be true for TTS metrics developed quite recently that is based on neural networks [CLS+20; JYB+22]. These networks would need to be adjusted for a specific environment, for example, when evaluating the speech of an unsupported language.

In practice, a subjective TTS evaluation metric - mean opinion score - is the most popular. It is used in most of the scientific literature where speech synthesis systems need to be evaluated [HCC+14; Ito17; KKK+20; KKS21; LLL+19; ODZ+16; OLB+18; RHQ+20; RRT+19; SPW+18]. Because of the reasons mentioned above, the mean opinion score subjective evaluation metric was

[10] https://www.openslr.org/12/

Table 1. Mean opinion score table [RFZ+11]

Rating	Quality	Naturalness	Distortion
5	Excellent	Completely natural	Imperceptible
4	Good	Mostly natural	Just perceptible, but not annoying
3	Fair	Equally natural and unnatural	Perceptible and slightly annoying
2	Poor	Mostly unnatural	Annoying, but not objectionable
1	Bad	Completely unnatural	Very annoying and objectionable

used to evaluate the TTS models created in this work. This metric is described below.

1.6.1. Mean Opinion Score

The mean opinion score (MOS) [Rec94] is a subjective evaluation metric calculated by collecting the results of a subjective listening test. Various systems are evaluated using MOS, including text-to-speech systems. Subjective listening tests are generally regarded as the most reliable and definitive way of assessing speech quality and naturalness [RFZ+11]. In general, subjective quality measures require that [RRK21]:

- There are enough listening subjects of sufficient diversity to produce statistically significant results;
- Experiments are conducted in a controlled environment with specific acoustic characteristics and equipment;
- Every subject receives the same instructions and stimuli.

In a MOS test, the listeners evaluate randomly selected samples (also called signals, or utterances) from a pool of speech samples. In the case of TTS MOS, the pool of speech samples usually contain audio samples generated by multiple TTS systems and ground truth (real human speech) samples. The MOS score is calculated for each of the sources. Also, the limitation is that the same listening test should never contain two samples created from the same utterance by the same source.

At the beginning of a test, respondents are asked to use headphones to achieve better results because people have a smaller discrimination capacity by using loudspeakers. They are also provided with a MOS score table, similar to the one presented in table 1.

2. Lithuanian Speech Dataset

A speech corpus (dataset) meant to train a TTS neural network should contain pairs of short text sentences and corresponding single speaker speech recordings. The quality of these text-audio pairs should be as high as possible as experimental results provided in the results section of this work

show that a neural network's ability to learn to synthesize intelligible, comprehensible, and high-quality speech, depends very much on the quality of the dataset on which it is trained. There are multiple requirements for a good quality dataset. Some of them are noted below:

- The text labels should match the spoken sounds of the corresponding recording as much as possible.
- The quality of the speech recordings produced by a TTS neural network corresponds to the quality of the speech recordings that the network learned from. Therefore, The quality of the speech recordings should be as high as possible.
- A speech corpus should be large enough for a neural network to learn various speech features of a language. For now, most of the papers that introduce state-of-the-art speech synthesis neural network models use speech datasets that contain as much as dozens of hours of speech data [Ito17; KKS21; RRT+19]. Research is conducted on models that require less data (up to a few minutes) to train a speech synthesizer, but such models do not compare to the state-of-the-art models in terms of quality.
- The speech recordings should contain background noise, breathing sounds, and other sound artifacts as little as possible. These artifacts are usually not preferred in a good-quality speech synthesizer. The reasons are analogous to the previous point: the neural network will learn to synthesize speech similar to the recordings it learned from.

One way to assemble such a dataset would be to hire a professional speaker, record well-prepared text utterances in a recording studio, and post-process and validate the recordings to create a dataset. This way, a high-quality dataset is guaranteed, but such a workflow is very expensive. Since the open-source Lithuanian language datasets are too small to train a TTS neural network of reasonable quality, for the purpose of this work, a solution was chosen to use pairs of large-sized audiobooks narrated by a single speaker and respective digital books. Professional speakers usually narrate audiobooks in a recording studio with high-quality equipment, so the quality of such audio clips is sufficient for neural network training. Also, there are many audiobook narrators in the Lithuanian language, so there are multiple options for a preferred synthesizer voice. This work collected three single-speaker speech datasets of different sizes and qualities. The speakers for each dataset are Vytautas Radzevičius, Aurimas Nausėdas, and Giedrius Arbačiauskas. Some high-level information about the datasets and the books they are comprised of are presented in table 2:

Table 2. A list of the books used in creation of the speech datasets.

Speaker	Title	Author	Year Recorded	Duration
Vytautas Radzevičius	Detektyvas	Arthur Hailey	2003	18 val. 20 min.
Vytautas Radzevičius	Ir padegė šiuos namus	William Styron	2004	22 val. 58 min.
Vytautas Radzevičius	Vakarykštis pasaulis	Stefan Zweig	2005	16 val. 31 min.
Vytautas Radzevičius	Meteoritas	Dan Brown	2006	16 val. 43 min.
Vytautas Radzevičius	Fenikso brolija	J. K. Rowling	2006	25 val. 38 min.
Vytautas Radzevičius	Hobitas arba ten ir atgal	John R. R. Tolkien	2007	9 val. 51 min.
Aurimas Nausėdas	Vakarų fronte nieko naujo	Erich Maria Remarque	2019	7 val. 8 min.
Giedrius Arbačiauskas	Altorių šešėly	Vincas Mykolaitis Putinas	2020	32 val. 10 min.

Note that after processing the speech data, the resulting total duration of speech recordings in each dataset is lower than the duration of the audiobooks. The following section describes the steps required to process audiobook and digital book pairs into a speech corpus suitable for text-to-speech neural network training.

2.1. Creating the Lithuanian Speech Dataset

As mentioned in the previous sections, audiobook and digital book pairs were used to create three Lithuanian language speech datasets (corpora). A metadata validation was carried out to ensure that the editions of each audiobook and digital book pair are equal. This way, there is less discrepancy between the speech recordings and the text. Nevertheless, the beginnings and the ends of audiobooks and digital books usually differ because of the introductory and concluding content. These parts were cut out manually so that both the audiobook and the digital book start and end in the same sentences.

In theory, text-to-speech neural networks can train on samples (audio-text pairs) of any audio duration and text length. However, in practice, shorter samples are used for training because of the lack of computational resources. Usually, the speech datasets contain audio samples of duration ranging from 0.3 to 16 seconds. In this work, the samples in each dataset are collected in the range of 0.3 to 15 seconds. For the experiments, shorter or longer samples are filtered out depending on the neural network architecture. The sample filtering is done because some neural network architectures perform worse when trained on too short or too long samples. Also, the max length of the samples was set to 15 seconds to save computational resources (because the longer the samples, the more VRAM is required by neural networks during training) and be able to set a larger batch size during training.

2.1.1. Text pre-processing

As mentioned before, the digital book text needs to correspond to the audiobook recordings as much as possible. The digital book needs multiple steps of processing to achieve this. Most digital books are stored in EPUB, MOBI, or PDF formats. So, to begin with, they must be converted to

a TXT format. This conversion usually introduces various artifacts in text. That is especially true when converting a PDF book that does not contain text data (it contains only images) and needs to be processed using optical character recognition (OCR) algorithms. Some of the most commonly found artifacts are listed below:

- *Misspelled letters*. Text processing and conversion algorithms are most effective for the English language. Therefore it recognizes the English language alphabet the best. When converting other language text, the letters from the corresponding alphabet are sometimes not recognized or misinterpreted. For example, the Lithuanian language letter *ą* (*a* with an ogonek) is sometimes interpreted differently: a letter *a* with a comma (*a,*), letter *q*, or letter *a*. A letter *š* (*s* with a caron) may be interpreted as simply a letter *a*, or even as a letter *s* with an additional letter *v* placed one line above the letter. Many other letter misinterpretation artifacts are found after converting a digital book to a TXT format.

- *Page headers and footers*. The artifacts caused by headers and footers are especially notable when executing an OCR over a PDF file. The text repeated on every page in a digital book is also repeated in the converted TXT file. That includes page/chapter numbering, chapter/book/author names, and other symbols like horizontal lines and star symbols *. Since a correspondence between text and audio recordings is desired, this repeated, not narrated text needs to be removed.

- *Out-of-vocabulary symbols*. In the context of text-to-speech systems, a set of symbols supported by the synthesizer is called a *vocabulary*. More details on this will be presented in later chapters, but for now, it is worth noting that book conversion to a TXT format often produces many out-of-vocabulary (OOV) symbols. Some of these symbols may be replaced automatically, but others need a manual examination to ensure that important text information is not lost. A list of replaceable out-of-vocabulary symbols that were encountered the most often during the collection of the datasets described in this work is shown in table 3.

Table 3. Out-of-vocabulary symbols found after digital book conversion to a TXT format. The table contains the symbols that the OOV symbols were automatically converted to. <*blank*> and <*whitespace*> denote an empty and a whitespace characters respectively. Note that this is not a full list of possible OOV symbols, but only those that were encountered the most often.

OOV symbol	Replacement
_	<*whitespace*>
≪	<*blank*>
≫	<*blank*>
\	<*blank*>
®	<*blank*>
·	<*blank*>
∘	<*blank*>
○	<*blank*>
2	<*blank*>

OOV symbol	Replacement
—	-
–	-
―	-
‒	-
’	’
‘	’
‚	’
“	”
”	”
„	”
’’	”

It is possible to fix a part of the artifacts automatically (regular expressions help a lot), but others need to be found and fixed manually, which is time-consuming.

Audiobooks are usually split into multiple audio files (chapters) of varied duration. On the other hand, digital books are provided as a single file. The monolithic text file needs to be split into chapters corresponding to the audiobook to be able to align text and audio later. It is preferred to split the text into chapters rather than concatenate the audio files to save computational resources during future processing steps.

After fixing the text conversion errors and splitting the text into respective audiobook chapters, the text needs further processing to ensure that the speech recordings correspond to each symbol as much as possible. This process involves a considerable amount of manual work, detailed in the following sections. For now, an overview of the points that need to be fixed in the text is presented below:

- *The beginning and the end of a digital book.* Both the audiobooks and the digital books usually have information segments at the start, end, or both. In the case of an audiobook, that may include the publisher, speaker, year narrated, etc. For a digital book - the edition, author, publisher, year published, etc. These segments need to be removed to match the start and end of both the audio and the text. For the digital book, simply delete the text that is not narrated at the start and end of the audiobook. Similarly, in the case of the audiobook, audio processing software may be used to cut the irrelevant segments.

- *Footnotes*. Books often include a number of footnotes. These may describe an event mentioned in a text, define terminology, translate foreign text, etc. These parts are sometimes narrated, rephrased, or inaudible in a corresponding audiobook. Therefore, manually listening to these segments in the audiobook and accordingly editing the text is necessary. Most

often, the footnotes in a text can be navigated by an asterisk (*) or, in other cases - by a respective numbering.

- *Digits*. Digits are usually not included in the vocabulary of TTS neural networks but are narrated in audiobooks. Therefore it is necessary to find digits in text and convert them to corresponding word representations. This conversion is not a trivial task for some languages because of multiple possible inflections of number words depending on the context (other words in the sentence). To illustrate, the digit 7 in Lithuanian language sentences *"7 dienų neužtenka"* and *"savaitėje yra 7 dienos"* expands respectively: *"septynių dienų neužtenka"* and *"savaitėje yra septynios dienos"*. The task of automatically expanding the digits to words is not researched in this work, and open-source tools solving this problem for the Lithuanian language were not found. Therefore the digits in the text were converted to words manually.

- *Narrated abbreviations*. Most of the abbreviations found in the text are expanded and narrated by speakers in a corresponding audiobook. Usually, it is necessary to listen to the recording of the corresponding text segment containing an abbreviation to know its full-word representation in confidence. However, sometimes it is sufficient to assume the expanded word depending on the surrounding words in a sentence. Some of the most common abbreviations and their corresponding expanded forms: *kun. - kunigas*, *kan. - kanauninkas*, *prof. - profesorius*, *a. - amžius*, *m. - metai*, *d. - diena*, *dr. - daktaras*, *Šv. - šventas/švenčiausias*, *pvz. - pavyzdžiui*, *t.t. - taip toliau.*

- *Inaudible abbreviations*. There are also abbreviations that the speaker does not narrate in an audiobook. It is necessary to listen to these segments in an audiobook manually to know for certain. If the abbreviation is not narrated - it must be removed from the text. Most often, these are name abbreviations, e.g., *J. K. Rowling - Rowling*, *George R. R. Martin - Martin*, etc., but other cases may occur, too.

- *Foreign words*. It is not uncommon for foreign words to be present in a book. That is especially true for translated books. These foreign words are usually (but not always) narrated by following the phonology of the respective language. Therefore it is necessary to convert the text segments where these foreign words are present to match the target language phonology. Another option is to filter out such segments to save work time. That may be a preferred option if the foreign words appear in an audiobook frequently since editing all the occurrences may prove too time-consuming. It is also worth noting that sometimes the foreign words are not narrated by the speaker. Examples of foreign words converted to a form respecting Lithuanian phonology: *Stefan Zweig - Štefan Cvaig*, *Voltaire - Volteras*, *examen conscientiae - egzamen konšiantiae*, *Browday - Brodvėjus*, etc.

2.1.2. Audio and text alignment

After performing the text processing described above for each chapter text-audio pair, the text and

the respective speech in the audiobook need to be aligned to get the start and end timestamps for each text segment. As mentioned before, short samples are preferred to build a speech corpus. For this reason, the text needs to be split into shorter segments. In this work, the text was split into sentences. Next, sentences that take longer than 15 seconds for the speaker to voice were split again at the time the narrator makes a pause. It is possible to split longer sentences in the way mentioned above because most of the time, the speaker must take a breath sometime during 15 seconds of non-stop talking. The sentences were split according to the official Lithuanian language sentence punctuation directives [11].

Next, the prepared text and the respective audiobooks need to be aligned to produce the start and end timestamps of each sentence in the audiobook. This process is called *forced alignment* [MJV+98]. For this task, a third-party tool Aeneas [12] was used. The inputs to the tool are an audio file (audiobook chapter) and a text file containing sentences corresponding to the audio recording. Note that the order of the sentences must match the utterances in the audio file. The sentences in the text file must be split by double new lines for Aeneas to consider it a separate entry. Aeneas forced alignment is configurable. It is possible to set different output formats, like JSON or SRT (subtitle file format). It is also possible to set the alignment boundary percent value so that the boundary between two aligned segments would be closer to either the end of the first segment (boundary < 50%) or closer to the start of the second segment (boundary > 50%). During the collection of the datasets, it was noted that it is convenient to verify the audio-text pair samples using subtitle processing tools, so the output format for the Aeneas forced alignment was set to SRT with a text type - *subtitle*. The alignment boundary was set to 80% to allow more silence only on one side of a boundary rather than on both sides. That was done for easier editing later. The Aeneas forced alignment tool was used to align all the audio-text file pairs to output corresponding SRT files, containing the start and end timestamps for each sentence in the audio files.

Usually, audiobook narrators make pauses between each sentence of the book. Because of that, the aligned audio segments of sentences contain regions of silence at the beginning and endings. These silent regions can sometimes get relatively long (more than 1 second). A neural network that would learn from such a set of samples would also learn to synthesize silent regions at the beginnings and endings of the synthesized utterances. This claim is verified by the TTS neural network training experimental results discussed in later sections of this work. Redundant synthesis of silence segments is undesirable in a TTS system. For this reason, a script was written to further post-process the SRT files and automatically strip the silent region timestamps from each entry in SRT files so that each dataset audio sample would contain a maximum of 10 milliseconds of silence at the start and end of the sample. This way, the neural network would not learn to synthesize unnecessary silence.

After performing the said steps, a set of audio and subtitle file pairs corresponding to book chapters was produced. From here, it is possible to validate each dataset sample conveniently. For

[11] http://www.vlkk.lt/vlkk-nutarimai/nutarimai/nutarimo-del-lietuviu-kalbos-taisykles
[12] https://github.com/readbeyond/aeneas

this purpose, the author of this work used a free subtitle editing software *Aegisub*. Each sentence sample of each book chapter was validated using this tool for correct start-end time boundaries and valid text-spoken sound correspondence. After validating the samples, it is finally possible to use the timestamps in the SRT files to cut the long-duration audio files into a dataset of short audio-text pairs, filtering out the samples that are too short (< 0.3 seconds) or too long (> 15 seconds). By repeating the process for each audiobook, a collection of speech and text pairs was built.

The speech and text pairs were further processed into a zipped dataset format. That was done by compressing the samples in batches of 1000 using a ZIP compression method resulting in multiple zipped dataset files. Each zipped dataset file contains short audio segments and a manifest file that provides the metadata on each audio segment in the ZIP file (*text*, *sample_rate*, *id*, *group*, *audio_format*, etc.). Such a format was chosen for the following reasons:

- The zipped data is structured and is easier to analyze programmatically.
- Reusability: the dataset can be reused for multiple TTS system codebases, as the code for loading the zipped data can be reused, and the manifest file contains all the potentially required metadata about the audio files so that it can be used according to the use case
- The datasets are stored in Google Storage (Google Cloud Platform), where the pricing depends on the storage used, so by compressing the datasets, less data is stored in the Cloud, thus reducing costs.
- It takes less time to transfer the zipped datasets over the network since fewer network requests are required to send the batched ZIP files than thousands of files in an uncompressed format. The file transfer speed is essential when working with multiple cloud virtual machines.

2.2. Stressing text labels

The reviewed literature shows that most of the models use phoneme representation as inputs to the TTS neural network [KKK+20; KKS21; LLL+19; RRT+19]. An exception to this was the Tacotron model [SPW+18; WSS+17] that generates Mel-spectrograms directly from text (grapheme sequence). Regardless, the authors report that the model suffers from occasional mispronunciation because of this reason. They argue that there may never be enough data for a neural network to learn all the pronunciation nuances occurring in natural language. While grapheme-to-phoneme and phoneme alignment tools are openly available for high-resource languages such as English, this is not the case for low-resource languages, such as Lithuanian. Since grapheme-to-phoneme mapping is not a trivial task, other solutions may be preferable. This work aims to prove that using stressed text as speech synthesis neural network inputs improves the pronunciation of synthesized speech for languages considered higher-degree phonemic orthographies. Therefore, in this master's thesis, the speech datasets are pre-processed to contain stressed text labels. In later sections, the

results show that it improves the pronunciation as well as the naturalness of synthesized speech for multiple neural network architectures.

A Lithuanian language stressing tool developed by the researchers of Vytautas Magnus University [13] (further referred to as the VDU stressing tool) was used to stress the text labels of the collected speech corpora. The algorithm used by the tool is not revealed, but the author of this work assumes that it is rule-based or exploits decision trees or other machine learning algorithms. It uses a database of Lithuanian language word dictionaries with their respective morphological information. The tool takes a text segment of up to 5000 characters as an input and outputs a corresponding stressed text. The tools can accentuate words if they exist in the application database, while other words are left as-is. It is possible to select each stressed word and view its morphological information alongside the possible accentuation options. The words for which the stressing is ambiguous (words that can be stressed in multiple ways; hereafter referred to as homographs) are marked with a colored-background font. It is possible to change the accentuation of such words by selecting them and choosing the appropriate stressing option from the morphological information list provided by the tool. Other words are stressed unambiguously. Only morphological information can be inspected for these words, but the accentuation is fixed.

The tool tries to pick an appropriate accentuation for homographs automatically. The algorithm responsible for this functionality is assumed to be based on machine learning algorithms. However, its accuracy is not a perfect 100%, so these errors must be fixed. Furthermore, since the text labels were narrated by a human speaker reading a text that was not stressed, pronunciation errors occasionally occur when homographs are voiced in the recordings. These errors are harder to spot since the accentuation is correct, but the speaker pronounces it differently. In these situations, the text was stressed according to the speaker's pronunciation to make sure that neural networks using the datasets are misled as little as possible during training.

All the speech corpora text labels were exported to a TXT file first to simplify the dataset stressing process, where each line corresponded to a dataset label. The stressing was carried out by following these steps:

1. select, copy and paste multiple text labels (totaling up to 5000 characters) to the input text area of the VDU stressing tool

2. verify homograph accentuation and fix the errors by choosing the correct accentuation from the list provided by the tools

3. select, copy and paste the verified stressed text back to the TXT file, replacing the corresponding text without accentuation

4. repeat step 1 by selecting the following text segment

Using this workflow, most of the words of each of the previously described speech corpora sample labels were stressed. Each stress mark assigned to a particular character in a word is a

[13] https://kalbu.vdu.lt/mokymosi-priemones/kirciuoklis/

UTF-8 combining character combined with a previous letter in a word character sequence to form a stressed symbol. For example, two UTF-8 symbols *e* and *\u0300* are combined into a stressed letter *è*. The following three combining characters were used as accent symbol tokens in collected speech corpora and later for TTS model training: *\u0300* (grave), *\u0301* (acute), and *\u0303* (tilde).

2.2.1. Results

To summarize the process described above, the steps taken to create a single speech dataset with stressed text labels are the following:

1. Get an audiobook and a digital book pair
2. Convert the digital book to a TXT file format
3. Fix the conversion errors in the TXT file
4. Pre-process the book text
5. Split the book text into chapters corresponding to the audiobook
6. Force align the text and the audio files
7. Verify the samples
8. Export sample labels to TXT format files
9. Stress the text in TXT files
10. Replace the dataset labels with their stressed counterparts
11. Cut the audiobook audio and digital book TXT files into a sentence-long set of audio-text pairs
12. Convert the samples into a zipped dataset

This process is very time-consuming, so it was carried out thoroughly only for two books from table 2 that different speakers narrate - Aurimas Nausėdas (*Vakarų fronte nieko naujo*) and Giedrius Arbačiauskas (*Altorių šešėly*). The text for books narrated by Vytautas Radzevičius was only partially pre-processed and was not verified. Also, the text was stressed using the VDU tools but was not reviewed, and the errors were not fixed. Therefore, this dataset includes a number of outliers, which may impact the quality of the trained neural networks. On the other hand, the dataset contains many speech data (91 hours after processing) and may be suitable to be used for a base TTS model training. Even though the dataset contains outliers and may fail to produce natural-sounding speech by itself, it may learn the primary language features well, which could be transferred by fine-tuning the model on a higher-quality dataset. As shown in later sections, some neural network

architectures are more tolerant of data outliers, so the low-quality dataset may still be helpful. The quality of datasets is briefly reviewed in the following section.

In summary, three datasets of different sizes were assembled. Each dataset contains single-speaker recordings voiced by Aurimas Nausėdas, Giedrius Arbačiauskas, and Vytautas Radzevičius, henceforth referred to as Aurimas, Giedrius, and Vytautas datasets, respectively. The datasets totals 6.3, 26.6, and 91.9 hours of single-speaker speech data, respectively. More details on each dataset are presented in table 4 (following statistics provided by the LJSpeech dataset [14]) and figures 1 and 2

Table 4. Collected Lithuanian language datasets statistics.

Dataset Name:	**Aurimas**	**Giedrius**	**Vytautas**	**Total**
Total samples	4968	18058	66643	89669
Total characters	338294	1600039	5327451	7265784
Total words	44255	203434	672269	919958
Mean words per clip	8.91	11.27	10.09	10.26
Distinct words	16358	52177	141266	179243
Total duration	6 h 20 min	26 h 33 min	91 h 53 min	124 h 47 min
Mean sample duration	4.59 s	5.29 s	4.96 s	5.01 s
Min sample duration	0.8 s	0.30 s	0.38 s	0.30 s
Max sample duration	14.98 s	14.98 s	13.96 s	14.98 s

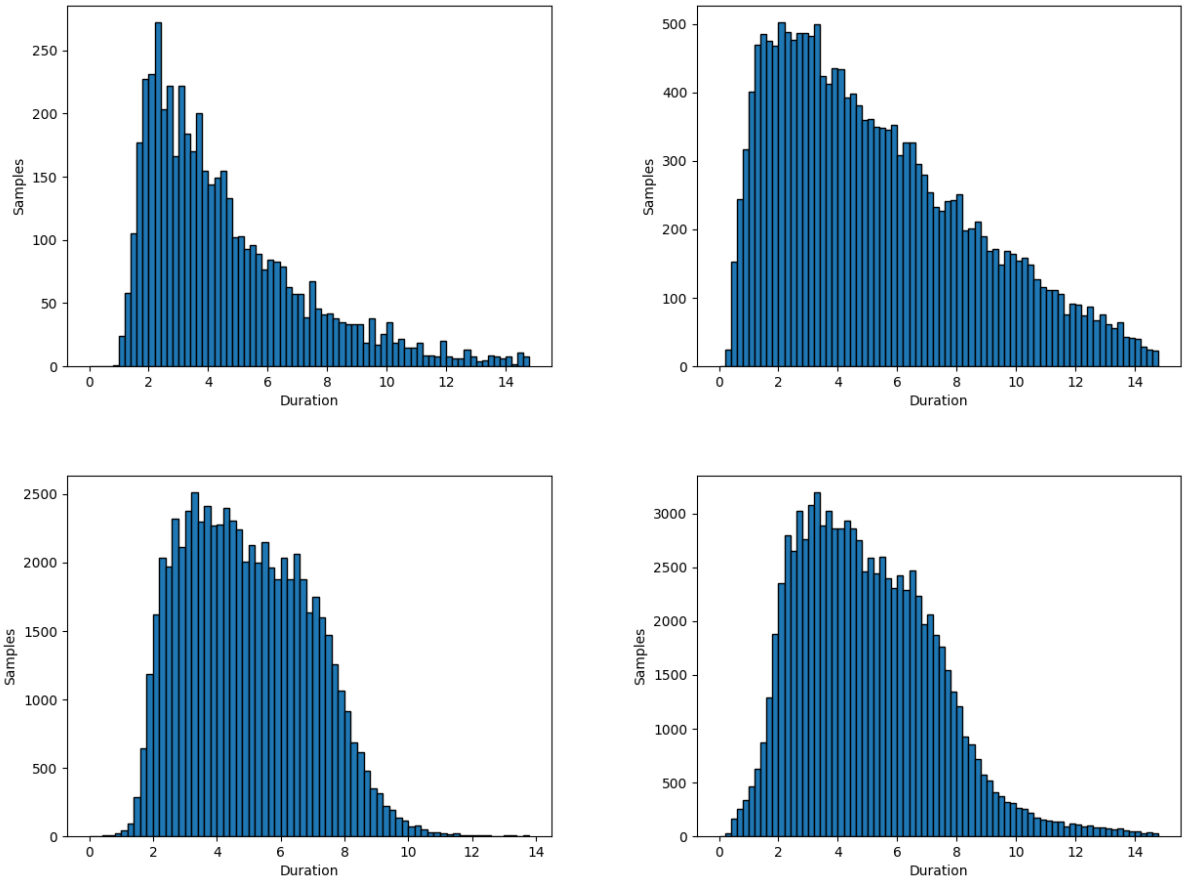
2.3. Data quality

The importance of data quality is at the core of a data-centric AI - an approach to building AI systems that are becoming mainstream in the AI community in recent years. Neural networks learn from the data it is given, so it is of the utmost importance that the data is of good quality. Even state-of-the-art deep learning algorithms perform poorly when using low-quality data to train the models. On the other hand, besides having good quality data, having lots of data may also prove beneficial [WRS+21]. For this reason, the poor-quality Vytautas dataset containing a large amount of single-speaker speech recordings was used for model training in this work. Vytautas dataset contains 91 hours of speech recordings. However, the dataset was not fully processed and cleaned, as mentioned before. Therefore it contains lots of outlier samples. Some distinguishing features of such samples are:

- Incorrect audio alignment with text: the speech at the start or the end of the sentence may start or end in the middle of a word.
- The speech recordings may not correspond to the text labels, e.g., the words spoken in an audio recording may differ from those in the corresponding text label.

[14] https://keithito.com/LJ-Speech-Dataset/

Figure 1. Sample duration distributions in datasets (Aurimas - upper left, Giedrius - upper right, Vytautas - lower left, total - lower right)



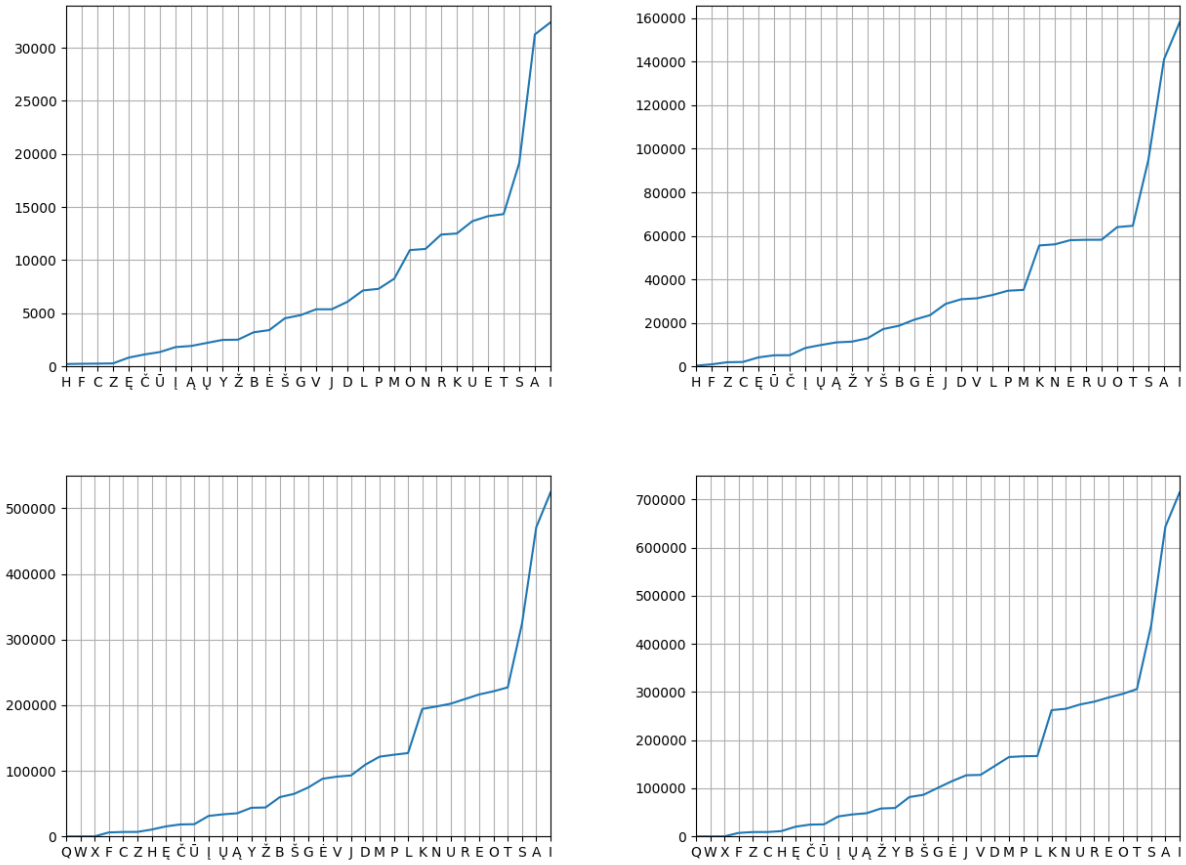
- Text labels may contain grammatical errors, out-of-vocabulary symbols, or other text artifacts.
- The speaker's pronunciation of some words may not correspond to the correct language phonemics. That is especially true for foreign language words that may occur in digital books and are narrated by the speaker. Also, due to the ethnic background of the speaker, some words may be mispronounced.
- Other unobserved anomalies.

Experiments in later sections show that training a TTS model on such a dataset may still produce a reasonable-quality TTS system, although it can occasionally produce artifacts. Furthermore, it is shown that using such a pre-trained model may significantly reduce the amount of time needed to fine-tune a model on new, higher-quality datasets.

3. Automatic Lithuanian Text Stressing Model

The Lithuanian TTS neural networks described in this paper use short audio clips and stressed text

Figure 2. Letter frequency distributions in datasets (Aurimas - left, Giedrius - middle, Vytautas - right, total - lower right)



label pairs for training. Therefore, accentuated word text sequence also needs to be fed to the trained model for it to infer a speech waveform (synthesize speech). The accent marks can be assigned to words manually if the speech synthesizer is used in a closed domain where a list of sentences that needs to be synthesized is known. However, this becomes tedious if the synthesizer is used for general purposes where the sentence may be different each time. In this case, the system user would need to write the UTF-8 stress mark tokens to appropriate letters in each sentence's words to be synthesized. Such a workflow is not suitable for the production environment, and the words should preferably be stressed automatically. As described in the literature review, such stressing models are usually not open-source. The most successful systems are based on morphological analysis, but this requires expert knowledge, time-consuming manual engineering, and development hours. Since a stressed text dataset was built for text-to-speech system use, the same data may also be used to train a neural network for the task. The task of training an automatic accentuation neural network is not widely researched in academia, so multiple approaches were considered to build such a system. The sections below describe the data used, the problem formulation, and experiments conducted to train a Lithuanian language automatic text stressing model.

Note that the experiments described in this section do not provide the main results of the work

and are only meant to explore the possibilities for future work on the topic.

3.1. Problem formulation

The purpose of an automatic text stressing system is to assign a correct stress mark to an appropriate letter in a word given a context (nearby words in the sentence). That is a natural language processing (NLP) problem since it concerns natural language text. Therefore, recurrent neural networks (RNN) [HS97; RHW86] or Transformer networks [VSP+17] may work best for the task. In this case, the problem would be formulated as a many-to-many problem, where the input tokens represent the raw text, and the output tokens represent the corresponding stressed text. Multiple approaches to the representations mentioned above are possible for a neural network to learn to perform such a task. This work covers only one approach to model input and output representations to explore the possibility of using neural networks for the task. However, it is worth noting that other approaches to the problem may work better for the task (plans for potential future work are presented in the final section below).

In this paper, a character-level many-to-many (N to N) problem is solved for the automatic stressing model. The inputs are a sequence of tokens representing the text character sequence. The model's outputs are also a sequence of tokens, representing the classification of each input token. The input vocabulary contains lowercase Lithuanian language alphabet letters, a whitespace token, a comma token, and the following special tokens: padding and an unknown symbol. The output vocabulary (or output classes) contains a token for padding, pitch accents (grave, acute, and tilde), and a symbol token (that represents any other symbol - a letter, punctuation, or whitespace). The input and output vocabularies are presented in the tables 5 and 6, respectively.

Table 5. Input vocabulary tokens. Symbols enclosed between angle brackets are special tokens (e.g., padding, unknown, and whitespace tokens)

Symbol	Token	Symbol	Token	Symbol	Token	Symbol	Token
<pad>	0	f	9	o	18	ą	27
<unk>	1	g	10	p	19	č	28
<whitespace>	2	h	11	r	20	ę	29
,	3	i	12	s	21	ė	30
a	4	j	13	t	22	į	31
b	5	k	14	u	23	š	32
c	6	l	15	v	24	ų	33
d	7	m	16	y	25	ū	34
e	8	n	17	z	26	ž	35

Table 6. Output vocabulary tokens.

Class	Token
padding	0
other symbols	1
grave accent	2
acute accent	3
tilde accent	4

An illustration of possible inputs and outputs in human-readable and tokenized (machine-readable) forms is presented below:

Human readable:

jei ji būtų sugavusi slogą

↓

jéi jì bū̃tų sugãvusi slógą

Tokenized:

13 8 12 2 13 12 2 5 33 22 34 2 21 23 10 4 24 23 21 12 2 21 15 18 10 27

↓

1 3 1 5 1 2 5 1 3 1 1 5 1 1 1 4 1 1 1 1 5 1 1 4 1 1

3.2. Data

The data used to train an automatic stressing neural network was extracted from the described speech corpora. As can be seen in table 4, in total, the resulting text corpus contains 919958 words across 89669 samples. Further data pre-processing was needed to use it for model training.

3.2.1. Pre-processing

To begin with, foreign language letters, irrelevant punctuation, and other artifacts needed to be replaced by alternatives. The corresponding sample was removed from the dataset if the alternatives were not available for such artifacts. Next, the sentences were converted into a format suitable for model training. The resulting list would contain a list of stressed text corpus samples. The text dataset size is insignificant (compared to other production datasets containing 5 GB of text data and more), so a simple CSV file format was chosen to store the dataset samples. The CSV file contains the input and the target (label) columns. The original stressed text corpus was stripped from all accent symbols to fill the input column with values. The values in the input column contain only the characters from the input vocabulary described in the previous subsection. Next, the output column was populated by mapping each symbol from the original dataset to the output vocabulary tokens, e.g., accents were mapped to tokens 2, 3, and 4 (acute, grave, and tilde, respectively), and other symbols were mapped to token 1. The resulting dataset was split into train, validation, and

test sets by ratios of 98%, 1%, and 1%, respectively. These splits contain 87876, 897, and 896 sentences, respectively.

3.3. Experiments

At the time of writing, only vanilla GRU networks [CVG+14] have been used to train an automatic accentuation neural network [Mos20]. The described network architecture is relatively simple, but it achieves a decent accuracy of 95.5%. Knowing this, the research in this work also started by training a simpler neural network architecture to have a comparable baseline model when more complex models are trained in the future. Specifically, a vanilla LSTM neural network architecture was used in the baseline experiment. Later, a more complex Transformer model (specifically, a modification of BERT [DCL+18] model - BERT for token classification [15]) architecture was used. These networks are known to produce state-of-the-art results in NLP tasks and, in theory, should provide better performance over the simple LSTM network. The details of the experiments are provided below.

3.3.1. Hardware

For both experiments, the same computational resources were used. The details of the hardware used are provided in table 7.

Table 7. Hardware used for stressing model training experiments.

Device	Name
Processor	Intel(R) Core(TM) i5-8600K CPU @ 3.60 GHz; 6 cores; 6 threads
GPU	EVGA GeForce GTX 1080 SC 8GB GDDR5X
Memory	2x Corsair 8GB 2933MHz C15 DDR4 + 1x G-Skill 16GB 2933MHz C16 DDR4
Storage	Samsung SSD 860

3.3.2. LSTM

The architecture for the LSTM neural network is relatively straightforward. First, the input tokens go to the embedding layer, and the embedding vectors are fed to an LSTM layer. The outputs of the LSTM layer are then fed to a linear layer. Finally, a softmax is used to calculate the probabilities of each token belonging to a specific task (whether the token is a symbol or a stress mark). The model architecture is illustrated in figure 3. Adam [KB14] is used to optimize a cross-entropy loss. Each predicted class is compared to the respective label to calculate the loss. Full model hyperparameters are presented in table 8.

[15] https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertForTokenClassification

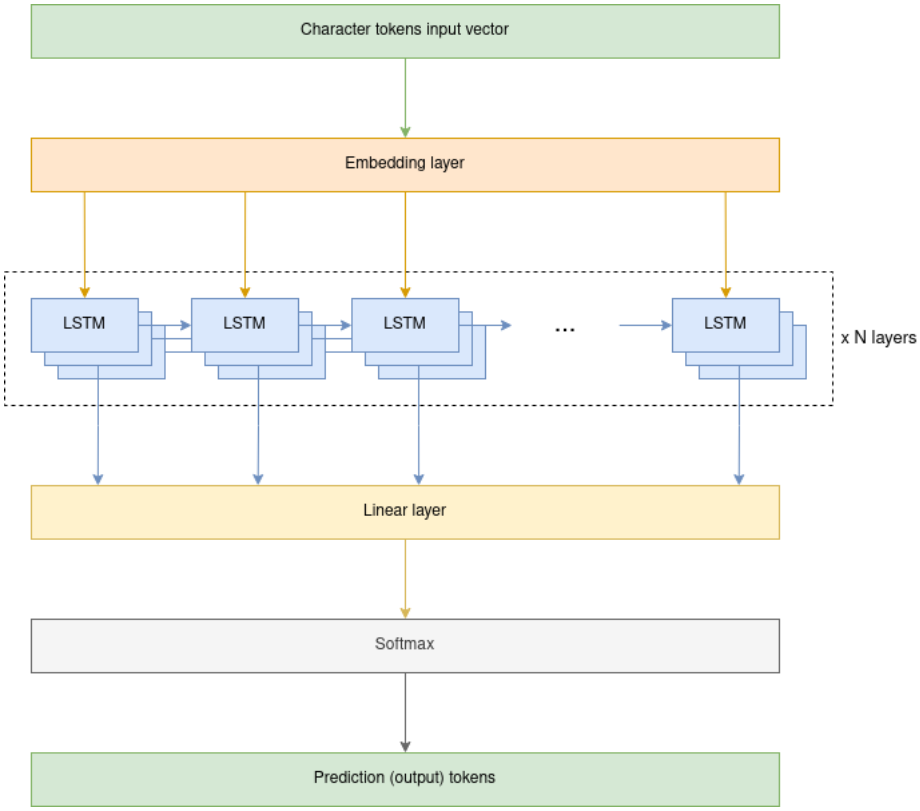


Figure 3. LSTM accentuation model architecture

Table 8. LSTM stressing model hyperparameters.

Hyperparameter	Value
embedding_dim	1024
hidden_dim	2048
lstm_layers	4
batch_size	64
learning_rate	$1e-5$
epochs	10

3.3.2.1. Results

The LSTM network was trained for 32440 steps (23 epochs), which took 11 hours. The model reached 93.4% accuracy on the test set. However, the accuracy and loss are both measured at a character level, i.e., the weight of loss is the same for each character classification, whether that is an incorrectly predicted or not-predicted accent mark for some character or any other symbol. Therefore the score may not show the actual ability of the model to assign the stress marks to words. A few inference results are presented in the table 9 below (the input text is shown before tokenization):

Table 9. LSTM stressing model inference results

Input	Output (human-readable)	Output tokenst
vienas lauke ne karys	víeñãs lãùk̃è nè ḱár̃ỹs	131441114242012133441
kaip šauksi taip atsišauks	kàĩp šãũksí tàĩp̀ atsíšãũk̀s	124111441131124121113444431
eikite tiesiai iki pastato galo	éìkìtè tìèsiài ìk̃ì pàstàtò gàlò	341313112411311342114212441141 4

The example shows that although the LSTM model tries to stress input tokens, it does not learn the pattern that only one pitch accent token can be classified between whitespace tokens. In other words, the model can assign multiple pitch accents to a single word. Although it can be confirmed that a word can contain multiple pitch accents in some languages (Swedish, Serbo-Croatian, etc.), that is not the case for the Lithuanian language. The data used for training did not contain words with multiple pitch accents assigned, so it might be the case that the cause of this problem is the loss function that may be too trivial for the task. By penalizing model predictions in case multiple or no pitch accents are assigned to a word in a sentence, the model may learn to assign exactly one stress mark for a word. That may be more easily implemented with bi-directional LSTMs, as in that case, the model could recognize the boundaries (whitespaces) of the words. Nevertheless, the research on this solution is left for future work.

It may be concluded that the model learned to predict the most probable accent assigned to each token in the input. It does not take significant consideration of the context of each token. Therefore, the model tends to assign a tilde accent for most *a* letters, an acute accent for most *i* letters, etc. These accents are often assigned to the individual letters in the dataset, so the pattern seems reasonable from the RNN network's perspective. However, the model is not usable for TTS input pre-processing cases and needs further improvements.

3.3.2.2. Motivation for choosing LSTM networks

As mentioned before, GRU networks were used to train an automatic text stressing model. Released decades after the introduction of LSTM networks, the architecture of GRUs is similar to LSTMs, but is relatively less complex and needs fewer parameters. Because of that, GRU networks offer faster convergence and inference speeds while retaining similar performance to LSTMs. Regardless, empirical research [YYZ20] shows that it only holds when the network is trained on smaller datasets where samples contain shorter sequences and fewer computational resources are used. In other scenarios, GRU networks demonstrate performance loss. The datasets and computational resources used during the experiments described in this work are relatively not small, so LSTM networks were chosen in hopes of gaining better model performance.

3.3.3. BERT

A modified version of the BERT model was used to solve the same problem as the LSTM accentuation model case - BERT for token classification. The model is made openly available by the

HuggingFace community [16]. The architecture deviates from vanilla BERT in that it stacks a linear layer on top of BERT's hidden state outputs. That way, the model can use the benefits provided by BERT to classify a sequence of input tokens. Originally, the model was created for tasks like named entity recognition (NER), where tokens are at a word level, but experiments are conducted using the model in the automatic accentuation domain to classify character-level tokens. A high-level overview of the architecture is illustrated in figure 4. The embedding layer is not represented since the BERT block includes it. More details on BERT architecture can be found in the original paper [DCL+18]. As in the LSTM model case, Adam was used to optimize a cross-entropy loss. Full model parameters are provided in table 10.

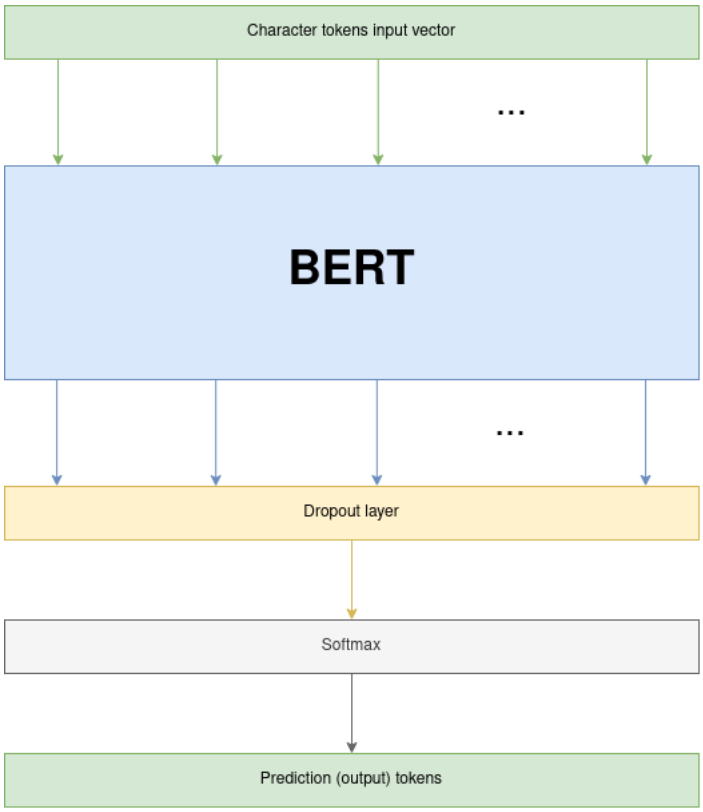


Figure 4. BERT accentuation model architecture

[16] https://huggingface.co/

Table 10. BERT stressing model hyperparameters (used in the original BERT paper [DCL+18])

Hyperparameter	Value
hidden_size	768
num_hiden_layers	12
num_attention_heads	12
intermediate_size	3072
hidden_dropout_prob	0.1
max_position_embeddings	5122
attention_probs_dropout_prob	0.1
batch_size	8
learning_rate	$1e-5$
epochs	10

3.3.3.1. Results

The BERT model was trained for 259500 steps (23 epochs) with a batch size of 8, which took 14 hours. Metrics show that the model reaches 44.8% accuracy after 10 epochs, which seems false since the model learns to output every input token as a letter token, as seen in the inference result table 11. The same cross-entropy loss is used in the LSTM example, so the accuracy should be higher. That allows concluding that the experiment failed. The failure may be caused by the fact that incorrect representations of inputs and outputs - the same as in the LSTM example - were chosen, and the BERT model requires a specific format for these.

Table 11. BERT stressing model inference results

Input	Output (human-readable)	Output tokenst
vienas lauke ne karys	vienas lauke ne karys	11111111111111111111
kaip šauksi taip atsišauks	kaip šauksi taip atsišauks	11111111111111111111111111
eikite tiesiai iki pastato galo	eikite tiesiai iki pastato galo	1111111111111111111111111111111

To conclude, similarly to the LSTM model experiment, the BERT model is not usable as a component for the TTS pipeline.

3.4. Summary

This section details the experiment of using two architectures based on LSTM and BERT neural networks to train models to accentuate text. The models work at the character level - each character in the input sequence is classified as either a non-accent symbol or grave, acute, and tilde symbols. The experiment proved to be a failure since both types of networks failed to produce reasonable stressing results. In conclusion, further work is needed to explore the possibilities of neural networks for the task of automatic accentuation.

4. Lithuanian Speech Synthesis

The literature review section provided an overview of multiple TTS neural network architectures with open-source code. In this work, two of the systems were trained on the collected Lithuanian language datasets: NVIDIA's implementation of Tacotron 2 [17] and VITS [18].

The decision to use Tacotron 2 is based on the reasoning that the system is highly regarded among the TTS community and offers high-quality speech synthesis. Moreover, the model can be trained using grapheme (character) representations as inputs. The downside of the system is that it is based on an acoustic-vocoder model architecture (meaning extended training and inference times), and the system is known to produce speech artifacts during synthesis due to its autoregressive nature.

On the other hand, VITS is a relatively new system, offering a complete end-to-end TTS model (no acoustic-vocoder model flow). The system is non-autoregressive and can utilize parallel computations for synthesis, so the inference speed is much faster than the Tacotron model. On the other hand, training VITS requires more computational resources. Also, in the original paper, the authors use phonemes as model inputs. However, the experiment results of this work show that stressed text works just as well.

This section describes multiple experiments of training Tacotron 2 and VITS models. We reasoned in the previous sections that there is no straightforward way to evaluate TTS models using objective metrics reliably. Therefore, a survey was conducted to collect subjective model scores and calculate a MOS (mean opinion score) - a subjective TTS evaluation metric. More details on synthesis evaluation are in the following sections. Nevertheless, the speech synthesis samples produced by each trained model can be found here [19], so the reader can form subjective conclusions.

4.1. Tacotron 2

Two neural network components need to be trained to build a Tacotron 2 TTS system: acoustic and vocoder models (Tacotron and Waveglow, respectively). The models were not trained from scratch using the collected Lithuanian language datasets in this work. Instead, the pre-trained acoustic [20] and vocoder [21] models provided by NVIDIA in their official Tacotron 2 Git repository were used to fine-tune on the Lithuanian speech data. The decision was made to reduce the computational costs since the authors of the original Tacotron paper state that 8 Nvidia V100 GPUs were used to train the model for 580000 iterations from scratch with a batch size of 24 on each GPU.

For this work, the Tacotron and Waveglow models were first fine-tuned on the 92-hour Vytautas dataset to create base Lithuanian TTS models, which were later used to fine-tune on higher-quality datasets (specifically - Aurimas and Giedrius datasets). The same hyperparameters were used for the

[17] https://github.com/NVIDIA/tacotron2
[18] https://github.com/jaywalnut310/vits
[19] https://arnasrad.github.io/lithuanian_synthesis_samples/index.html
[20] https://drive.google.com/file/d/1c5ZTuT7J08wLUoVZ2KkUs_VdZuJ86ZqA/view
[21] https://drive.google.com/file/d/1rpK8CzAAirq9sWZhe9nlfvxMF1dRgFbF/view

experiment as in the original Tacotron 2 paper (besides the learning rate, batch size, and vocabulary size). The Tacotron and Waveglow hyperparameters for training are provided in table 12. Adam optimizer and mean squared error (MSE) loss combined with binary cross-entropy (BCE) loss with logits were used during training.

Table 12. Tacotron 2 training hyperparameters (acoustic model - left, vocoder model - right)

Hyperparameter	**Value**
Character embedding dimension	512
Encoder kernel size	5
Encoder number of convolutions	3
Encoder embedding dimension	512
Decoder number of frames per step	1
Decoder RNN dimension	1024
Decoder pre-network dimension	256
Max decoder steps	1000
Decoder gate threshold	0.5
Pre-attention dropout	0.1
Pre-decoder dropout	0.1
Attention RNN dimension	1024
Attention dimension	128
Attention location number of filters	32
Attention location kernel size	31
Mel-post-net embedding dimension	512
Mel-post-net kernel size	5
Mel-post-net number of convolutions	5

Hyperparameter	**Value**
Mel-spectrogram channels	80
Coupling layers	12
Invertible 1x1 convolutions	12
Coupling layer dilated convolutions	8
Residual connections	512
Skip connections	256
Early channel output every N layers	4
Number of early channel outputs	2

The Tacotron acoustic and the Waveglow vocoder models were fine-tuned on three datasets - Aurimas, Giedrius, and Vytautas. Nevertheless, the models were trained using both the stressed and non-stressed text labels in the acoustic model case. So, nine models in total were trained for the Tacotron 2 experiment (6 acoustic and 3 vocoder models). During inference, the vocoder model was used to match the acoustic model target speaker, i.e., if the acoustic model was trained using the Aurimas dataset, then the vocoder model that was fine-tuned on the Aurimas dataset was used alongside during inference. The vocoders were fine-tuned for the target speaker to improve the quality of the final synthesized speech [HWH+19].

4.1.1. Experimental Setup

To train the Tacotron 2 speech synthesis model, the code from the original NVIDIA repo was forked and adapted to accept the zipped dataset format [22]. Next, to train the acoustic Tacotron model, the input vocabulary was modified to contain the Lithuanian alphabet letters and the three accent symbols (grave, acute, and tilde). The total vocabulary size then is 79. The support for ARPABET symbols available for the English synthesizer in the original repository was removed

[22] https://github.com/arnasRad/tacotron2

to reduce the vocabulary size and ease the training since the symbols would not be used for the Lithuanian language. The complete list of symbols supported by the Tacotron models is provided in table 13.

Table 13. Symbols supported by the Tacotron model for the Lithuanian language. <whitespace> is the whitespace symbol. $\backslash u0300$, $\backslash u0301$, and $\backslash u0303$ are the combining characters corresponding to the grave, acute, and tilde accent, respectively.

Symbol
_
<whitespace>
!
’
(
)
,
.
:

Symbol
;
?
-
A
a
Ą
ą
B
b

Symbol
C
c
Č
č
D
d
E
e
Ę

Symbol
ę
Ė
ė
F
f
G
g
H
h

Symbol
I
i
Į
į
Y
y
J
j
K

Symbol
k
L
l
M
m
N
n
O
o

Symbol
P
p
R
r
S
s
Š
š
T

Symbol
t
U
u
Ų
ų
Ū
ū
V
v

Symbol
Z
z
Ž
ž
$\backslash u0300$
$\backslash u0301$
$\backslash u0303$

The datasets were split into training, validation, and test sets by 98% - 1% - 1%. A relatively small number of samples for validation and test sets were used because the datasets contain many samples (4968, 18058, and 66643 samples for Aurimas, Giedrius, and Vytautas datasets, respectively). The test set does not require a large number of samples when evaluating TTS systems using a MOS metric because each sample needs to be evaluated manually, so testing a large number of samples may be too costly. The training, validation, and test set split sizes are shown in table 14.

Table 14. Sample number in training, validation, and test sets of each speech dataset used in this paper.

Set	Aurimas	Giedrius	Vytautas
Training	4869	17697	65310
Validation	50	181	667
Test	49	180	666

Training Tacotron and Waveglow models require less computational resources compared to VITS model training, so all the training experiments were carried out on the machine detailed in

table 7.

4.1.2. Training on Vytautas dataset

The training results for both the acoustic (using stressed and non-stressed text labels) and vocoder models are shown in table 15.

Table 15. Results of training a Tacotron 2 model on Vytautas dataset (*non-stressed* denotes the training on a dataset containing no accentuation, while *stressed* denotes the training on stressed text labels).

	Acoustic (non-stressed)	**Acoustic (stressed)**	**Vocoder**
Batch size	16	16	12
Iterations	82000	89000	84000
Epochs	20	21	15
Training duration (hours)	62	67	111

The models trained on stressed and non-stressed text labels are hereafter referred to as *stressed* and *non-stressed* models, respectively.

The resulting stressed model produces intelligible speech with solid pronunciation features, although both the acoustic and vocoder models produce various artifacts. For example, the acoustic model occasionally skips words, synthesizes noise until the decoder reaches max steps, and makes unnecessary pauses during speech. Also, the vocoder model does not synthesize high-quality audio, as the speech sometimes sounds *robotic*. Moreover, the non-stressed model often generates speech with pronunciation errors, making the speech unintelligible.

It is worth noting that the training was only executed to obtain a model that can capture major Lithuanian language prosody features at this stage. The model was not expected to perform well and was meant to be used as a baseline model for fine-tuning on other, better-quality datasets. The quality of the Vytautas dataset is relatively poor. Therefore, it is not expected that a good synthesis model would be trained using it. Because of this, the training was stopped early to save computational resources. Usually, the training would take more epochs.

4.1.3. Training on Aurimas dataset

The pre-trained model described in the previous subsection was fine-tuned using the Aurimas dataset first. The training results for both the acoustic and vocoder models are shown in table 16.

Table 16. Results of training a Tacotron 2 model on Aurimas dataset.

	Acoustic (non-stressed)	**Acoustic (stressed)**	**Vocoder**
Batch size	16	16	4
Iterations	20000	30000	24000
Epochs	65	98	20
Training duration (hours)	17	26	23

The resulting stressed model produces a relatively natural-sounding speech. The artifacts from which the model trained on the Vytautas dataset suffers rarely occur when synthesizing utterances using this model. On the other hand, these artifacts are not entirely gone, as the model still occasionally makes unnecessary pauses and produces noise. Furthermore, the non-stressed model makes similar (compared to the Vytautas model) pronunciation errors, reducing the naturalness of the synthesized speech.

4.1.4. Training on Giedrius dataset

Similarly, the base model was fine-tuned on the Giedrius dataset. The training results for both the acoustic and vocoder models are shown in table 17.

Table 17. Results of training a Tacotron 2 model on Giedrius dataset.

	Acoustic (non-stressed)	**Acoustic (stressed)**	**Vocoder**
Batch size	16	16	8
Iterations	25000	17500	50000
Epochs	22	15	22
Training duration (hours)	22	15	23

The fine-tuned model often synthesizes low-quality speech in terms of audio quality, intelligibility, and naturalness. These are exciting results as the dataset used to fine-tune the model was cleaned and validated, unlike the Vytautas dataset. After multiple experiments, it is argued that poor synthesis quality may arise for two reasons. First, Giedrius (the speaker of the dataset) has a lower voice tone. Because of this, the lower frequency sounds dominate in the speech produced by Giedrius, and the corresponding spectrograms have a higher concentration of frequency intensities in the lower part of the spectrogram. In contrast, the upper part provides less information for the neural network to learn from. Other speaker spectrogram frequencies are distributed along the y-axis of the spectrograms more evenly. Therefore neural networks may receive more information. Secondly, poor results may be caused because Giedrius uses very expressive speech when narrating audiobooks. Because of such a variety in prosody, the neural network may struggle to learn to synthesize certain utterances unambiguously. A more monotonic speech may be required to improve the model performance. The two claims mentioned earlier were not validated in this work and maybe a topic for further research. Nevertheless, the pronunciation of speech generated by the stressed model is relatively good. However, similar to the Tacotron 2 experiments described above, the non-stressed model performance suffers from pronunciation errors.

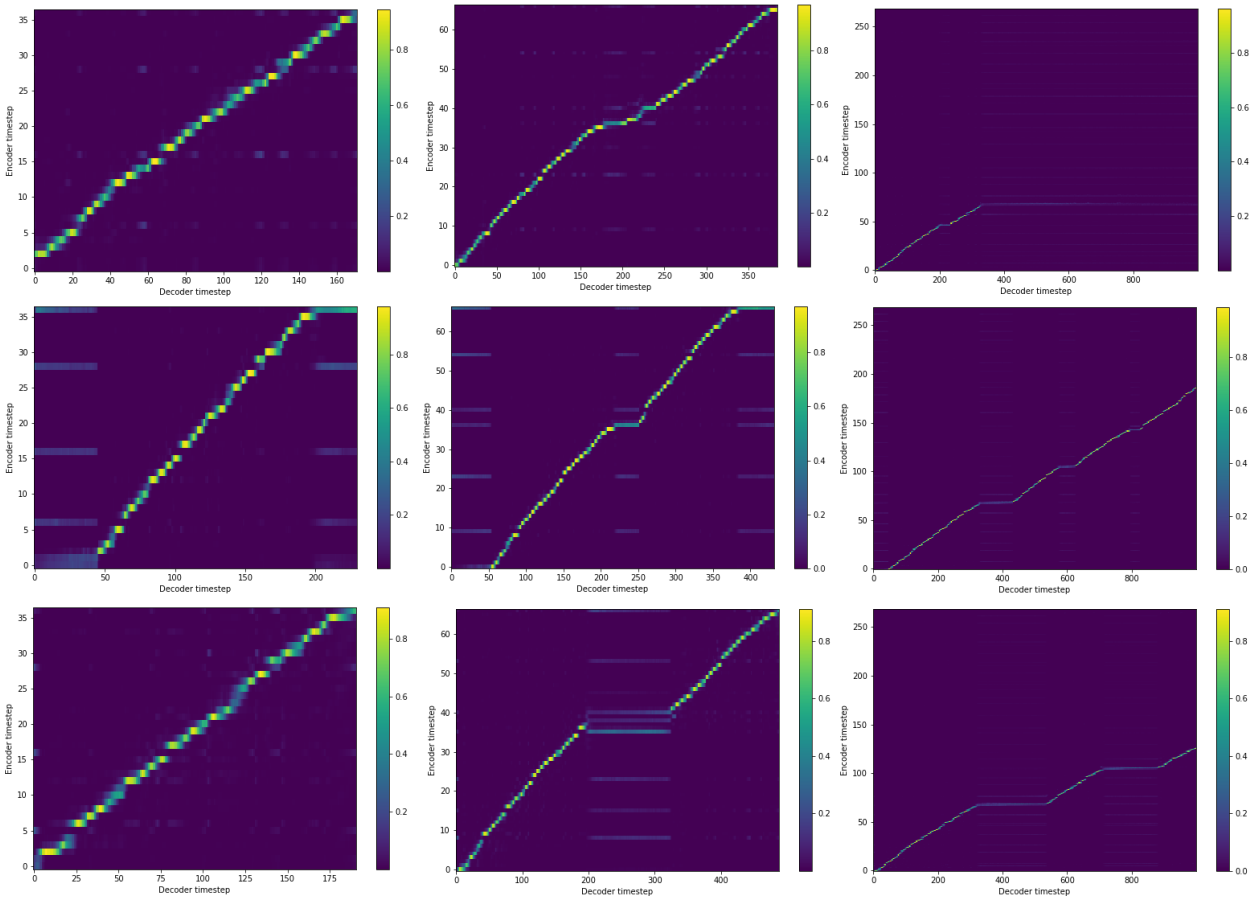
4.1.5. Summary

Nine models in total were trained during the Tacotron 2 experiment using three one-speaker datasets of different sizes: three Waveglow vocoder models, three acoustic models using text labels without accentuation, and three acoustic models using text labels with accentuation. Examples of attention

alignment plots of speech synthesized by stressed acoustic models are presented in figure 5. In the figure, the sentences used for synthesis are the following (left to right):

- *- Jūs klausote sintezuoto teksto.*
- *Vilniaus universiteto Matematikos ir informatikos fakultetas.*
- *Vasara paskutinių kursų studentams kasmet prasideda įtemptai - laukia baigiamųjų darbų gynimai. Vilniaus universiteto Matematikos ir informatikos fakultete šiais metais jie prasideda gegužės trisdešimt pirmąją ir baigsis birželio dešimtą dieną.*

Figure 5. Tacotron 2 inference alignment plots. Models used for inference (top to bottom): Vytautas, Aurimas, Giedrius.



We can conclude that the trained Tacotron 2 models attend to appropriate tokens rather well by observing the attention alignment plots. Although ideally, the alignment should be as diagonal as possible. Multiple flaws may be observed in the plots provided above. To begin with, the models cannot synthesize utterances longer than approximately 11 seconds (see the last column). Next, the models occasionally skip words and generate silence (see the last column of the first row). Also, if most dataset samples contain segments of silence at the beginning and end, a model trained on such a dataset learns to generate analogous segments of silence. Finally, some alignment plots are rather noisy, which causes lower quality synthesized utterances.

By comparing utterances synthesized by stressed and not-stressed models (see the samples provided on the website referenced above), we can clearly distinguish the pronunciation superiority of the stressed model, which dramatically impacts the naturalness of synthesized speech. Besides, by comparing utterances synthesized by the Aurimas model to the utterances synthesized by Vytautas or Giedrius models, we can see that a Tacotron 2 synthesis model works better for some speech data than others. That may be due to prosody or audio features of dataset speech recordings, like speech tone, expressiveness, etc. Overall, the speech generated by the trained Tacotron 2 models described above has various artifacts: 'noisy' voice, word/letter skipping, generating noise, etc. The artifacts impact the preferability of the models drastically. On the other hand, in most cases, the stressed models overcome the pronunciation errors that arise when generating speech without using accentuation.

As a last note, fewer computational resources are required to train the Tacotron 2 model, but two models (acoustic and vocoder) need to be trained to perform better quality inference. Because of this, the training takes more time overall compared to the VITS model described in the next section.

The Tacotron 2 base models (Waveglow vocoder, stressed and non-stressed acoustic models) are provided here [23].

4.2. VITS

Another neural network architecture used to train a TTS model on the datasets described above is VITS. Even though the original VITS paper authors claim to have used phonemes as inputs to the model, the experimental results provided in this work show that stressed text inputs also work. There is an open-source phonemizer tool [24] that can supposedly convert the Lithuanian language text to phonemes. A model was trained using the phonemized text labels by the author of this master's thesis. The phonemizer for Lithuanian languages provides poor results as the speech using the converted phonemes is unintelligible. The synthesized samples (Paulius dataset) can be seen by following the above link. The details on the Paulius dataset are not described here because it is out of scope for this work.

Similar to the Tacotron experiments, the VITS model was trained on Vytautas, Aurimas, and Giedrius datasets in this work. The difference here is that the models were only trained using stressed text labels because VITS training proved highly time-consuming. The reason is that the base stressed model was trained from scratch without fine-tuning on a pre-trained English model (unlike the Tacotron 2 experiment). The training consumed a huge amount of time. On the other hand, VITS was only trained from scratch once. The training was done using the Vytautas dataset. The resulting model was used as a baseline model to fine-tune on higher-quality Aurimas and Giedrius datasets. For both experiments, the same model hyperparameters were used as in the

[23] https://drive.google.com/drive/folders/1c7Y2H1ch0GeN$_Ljrf0kWaZgoTZVhktNW?usp = sharing$

[24] https://github.com/bootphon/phonemizer

original repository. The hyperparameters are provided in table 18.

Table 18. VITS training hyperparameters

Hyperparameter	Value
Inter channels	192
Hidden channels	192
Filter channels	768
Number of heads	2
Number of layers	6
Kernel size	3
Dropout	0.1
Resblocks	1
Resblock kernel sizes	[3, 7, 11]
Resblock dilation sizes	[[1, 3, 5], [1, 3, 5], [1, 3, 5]]
Upsample rates	[8, 8, 2, 2]
Upsample initial channel	512
Upsample kernel sizes	[16, 16, 4, 4]
Number of layers q	3

The Tacotron experiment has shown that the Lithuanian language TTS models make similar pronunciation errors during inference when using non-stressed text inputs regardless of whether the models were trained using accentuated text labels or not. Since VITS needed to be trained from scratch, and the training of a single VITS model took almost a month using the available resources, as shown in the later sections, the models described below were only trained using stressed text labels to save computational resources. Later, the stressed VITS model synthesized the non-stressed utterances in the MOS survey by feeding it text inputs without accentuation.

Non-stressed text labels were not used for training in this experiment. Also, the VITS model does not require separate acoustic and vocoder models for inference and can synthesize speech from a single generator model. Because of these reasons, only three VITS models were trained. The models were trained using the same Aurimas, Giedrius, and Vytautas datasets analogous to previous experiments.

4.2.1. Experimental Setup

Similar to the Tacotron experiment, the VITS code was forked from the original repo and adapted to support the zipped dataset format [25]. The input vocabulary was also modified for the Lithuanian alphabet (including stress marks). The resulting vocabulary is equivalent to the one used in Tacotron 2 experiment and is provided in table 13. The training, validation, and test set splits are also the same as in the Tacotron experiment (table 14.)

Training the VITS model requires more computational resources, so the computer setup used in the Tacotron experiment (table 7) is not sufficient. Therefore, all the VITS training experiments

[25] https://github.com/arnasRad/vits

were conducted using a paid cloud-based virtual machine (VM) provided by the GCP (Google Cloud Platform) Compute service. High-level VM hardware specifications are the following: 2-core CPU, 1 x Nvidia Tesla T4 GPU (16 GB VRAM), and 12 GB of RAM.

4.2.2. Training on Vytautas dataset

As in the Tacotron 2 experiment, a base Lithuanian speech synthesis model was trained using the 92-hour Vytautas dataset. The difference here is that the model was trained from scratch without using a pre-trained English speech synthesis model for fine-tuning. Likewise, VITS is not based on acoustic-vocoder model inference flow, so only a single model was trained. Furthermore, only stressed text labels were used to train the model. The resulting training information is provided in table 19.

Table 19. Results of training a VITS model on Vytautas dataset.

	Discriminator/Generator
Batch size	16
Iterations	3000000
Epochs	735
Training duration (hours)	946

The trained model produce high-quality audio that is almost identical to that of the speech recordings that were used to train the model, and relatively natural-sounding synthesized speech. On the other hand, the model generates a number of speech artifacts: reverb-like sounds, skipping letters, and pronunciation errors.

4.2.3. Training on Aurimas dataset

Again, similar to Tacotron 2 experiment, the base model was used to fine-tune it on higher-quality data. First, the Aurimas dataset was used for fine-tuning. The resulting training information is provided in table 20.

Table 20. Results of training a VITS model on Aurimas dataset.

	Discriminator/Generator
Batch size	16
Iterations	150000
Epochs	492
Training duration (hours)	70

As later observed from the MOS results, the fine-tuned model generates near-human naturalness and quality speech. The pronunciation, intonation, quality, and other speech features are pretty solid in this case. Also, artifacts are pretty rare in the speech generated by the model.

4.2.4. Training on Giedrius dataset

Finally, the VITS base model was fine-tuned on the Giedrius dataset. The resulting training information is provided in table 21.

Table 21. Results of training a VITS model on Giedrius dataset.

	Discriminator/Generator
Batch size	16
Iterations	525000
Epochs	474
Training duration (hours)	192

Unlike the Tacotron case, the resulting speech synthesized produces speech that has a quality similar to the original recordings. On the other hand, compared to the model trained on the Aurimas dataset, the model produces more speech artifacts and does not sound as natural. A subjective opinion for the potential reason for this is that the speaker of the dataset (Giedrius) is expressive and the speech tone variety in dataset samples is more significant than in the Aurimas dataset. Because of the variety, the model struggles to figure out the correct prediction for each phoneme during inference. Note that this is only an educated guess, and the claim is not validated. A possible way to evaluate the claim is to filter the dataset, leaving only the samples that sound as similar as possible, and train the VITS model on the filtered dataset.

4.2.5. Summary

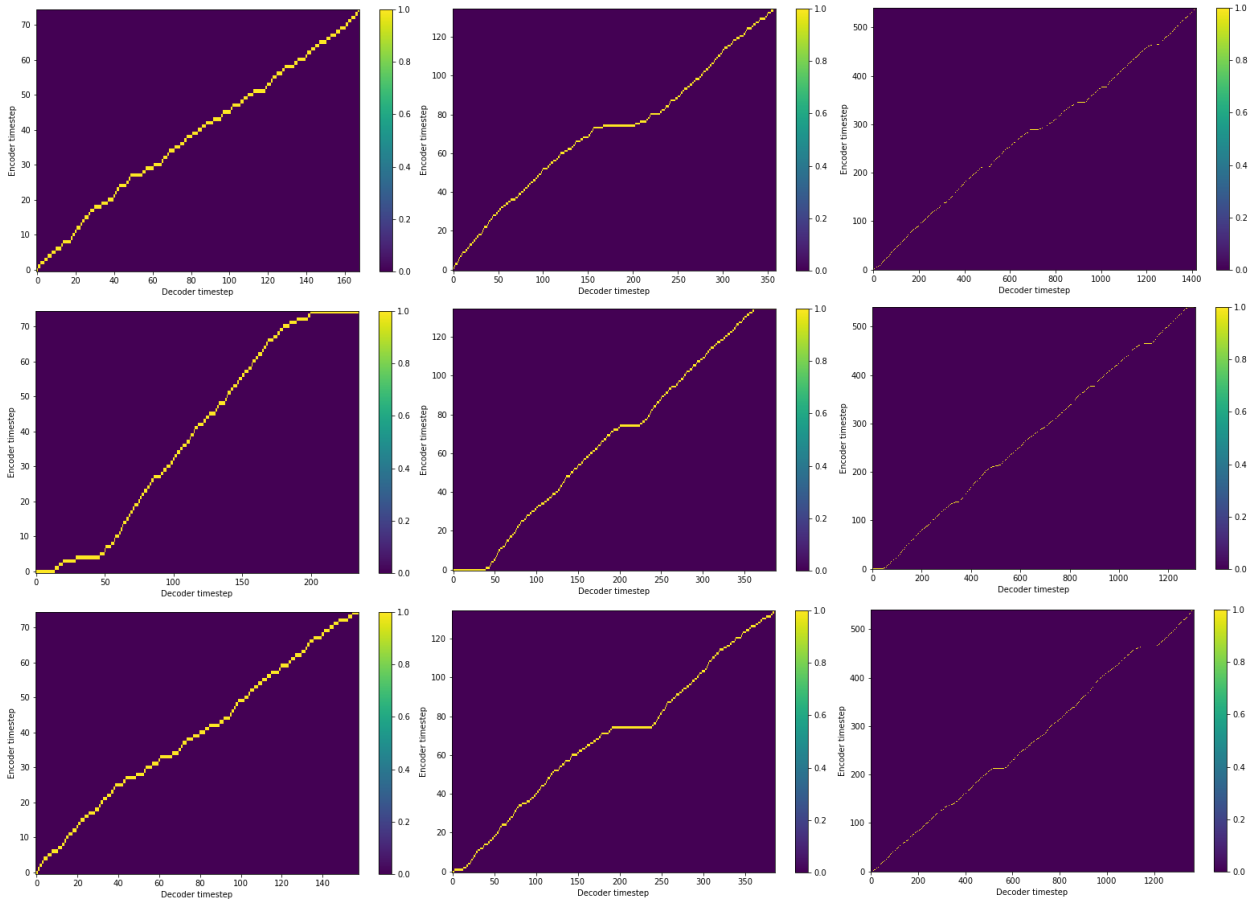
Three VITS models were trained using the same datasets as in the Tacotron 2 experiment. VITS model requires more computational resources to train compared to Tacotron 2. It is also very time-consuming: training the base model took more than a month. On the other hand, the inference process is more straightforward, as only one model is used to synthesize speech instead of the acoustic and vocoder models in the Tacotron 2 case. Examples of the attention alignment plots of speech synthesized by the stressed VITS models are presented in figure 6. The sentences used to generate the alignments are the same as the ones used in table 5.

By inspecting the alignment plots, we can see that the VITS model produces more diagonal alignments (except for the parts where the models react to punctuation). It also overcomes some of the flaws produced by the Tacotron 2 model:

- It can generate utterances longer than 11 seconds.
- It does not skip words or generate silence in place of words.
- The alignments are less noisy.

On the other hand, it learns to generate segments of silence the same as the Tacotron 2 model.

Figure 6. VITS inference alignment plots. Models used for inference (top to bottom): Vytautas, Aurimas, Giedrius.



Again, by comparing utterances (provided in the website referenced above) synthesized with and without accents, we can see that stressed text labels significantly improve the pronunciation of the synthesized speech.

The VITS base model can be downloaded here [26].

5. Results

As mentioned before, in practice, a mean opinion score (MOS) is used to evaluate TTS systems [RFZ+11]. It is a subjective evaluation metric that works by conducting a survey where participants are presented with a list of audio samples generated by multiple sources (ground-truth, various synthesis methods) and are asked to rate them by some arbitrary score (usually, 5-scale scoring is used).

In this work, a MOS survey was conducted to evaluate the models described in previous sections. The participants from various professional backgrounds were selected to conduct the survey. Before starting the survey, all the respondents received the same brief information about the context

[26] https://drive.google.com/drive/folders/1sDqsDlrzdSx12hOPQ89zelJWoksoIFth?usp=sharing

of the survey, its goals, and the instructions on how to complete it. The participants were instructed to use headphones for the duration of the survey.

Each survey question consists of a set of samples generated by different sources to be evaluated. Measures were taken to ensure that the participants were not biased towards particular speech sources. Some of them are provided below:

- The respondents did not receive any information about the speech sources
- Multiple samples generated by the same source cannot be present on the same question
- A unique sentence is used in exactly one sample set throughout the survey
- The same sentence generated by different sources is evaluated in the same sample set
- Exactly one ground-truth sample must be present in each question

In this work, two MOS surveys were conducted. The first one evaluated the trained TTS neural network models described in this work. The survey aimed to prove the main point of this work that stressed text labels improve the pronunciation, and therefore naturalness, of speech synthesis neural network models. The second survey also analyzes the best current Lithuanian speech synthesizers. The model that received the highest MOS score in the first survey was compared to other Lithuanian speech synthesizers. A free survey Web tool *QuestionPro* [27] was used to build both surveys and collect the results.

5.1. Trained Models Evaluation

First, a survey was conducted to evaluate the trained models described in the previous sections and tackle the primary goal of this work - whether the stressed text labels improve the quality of speech synthesis systems for higher degree phonemic orthography languages. The survey was created following the guidelines described above. 15 speech sources were evaluated: 12 TTS neural network model sources and 3 ground truth sources from the datasets used to train the models. The sources produce 3 different speakers' voices that narrated the 3 datasets: Aurimas, Giedrius, and Vytautas. Six Tacotron 2 and three VITS models were evaluated. These models are described in the previous sections. Note that only three VITS models were trained (using stressed text dataset labels). For the survey, three additional VITS speech sources were used by feeding the stressed models non-stressed text for synthesis.

30 unique sentences from each of the test sets of the Aurimas, Giedrius, and Vytautas datasets were randomly sampled to generate samples for the survey. In total, 90 unique sentences were used to generate 450 samples evaluated in the survey (30 utterances for each speech source). The duration of the samples ranges from 1 to 17 seconds. The duration of the samples alone makes up approximately 45 minutes, and the time it takes to complete the actual survey is approximately 1

[27] https://www.questionpro.com

hour and 30 minutes. Because of this, it was hard to find respondents who were willing to complete the survey, so only 11 respondents participated. For reference, the link to the survey is provided in the footnotes [28].

Note that the synthesis of all the utterances from all the sources was done in one go without fixing the errors. As a result, the generated samples reflect the robustness of each model. Also, some answers contained missing values (scores assigned to samples). These were replaced by the average of other participants' scores

The results of the MOS survey were grouped by the ground-truth sources (Vytautas, Aurimas, and Giedrius) and are provided in the tables 22, 23, 24 below.

Table 22. Comparison of evaluated mean opinion score with 95% confidence intervals on the Vytautas dataset.

Model	**MOS ±CI**
Tacotron 2 (non-stressed)	1.77 ±0.09
Tacotron 2 (stressed)	1.88 ±0.10
VITS (non-stressed)	1.88 ±0.10
VITS (stressed)	2.61 ±0.12
Ground truth	3.68 ±0.11

Table 23. Comparison of evaluated mean opinion score with 95% confidence intervals on the Aurimas dataset.

Model	**MOS ±CI**
Tacotron 2 (non-stressed)	3.26 ±0.13
Tacotron 2 (stressed)	4.31 ±0.10
VITS (non-stressed)	2.62 ±0.13
VITS (stressed)	4.42 ±0.08
Ground truth	4.73 ±0.06

Table 24. Comparison of evaluated mean opinion score with 95% confidence intervals on the Giedrius dataset.

Model	**MOS ±CI**
Tacotron 2 (non-stressed)	2.19 ±0.13
Tacotron 2 (stressed)	2.94 ±0.13
VITS (non-stressed)	2.33 ±0.12
VITS (stressed)	3.97 ±0.10
Ground truth	4.58 ±0.07

The results show that the samples generated by stressed text models outperform their non-stressed counterparts in all cases by a significant margin. That is especially true for the VITS models, possibly because its architecture was built to support phonemes as inputs that comprise

[28] https://questionpro.com/t/ASd5gZsp35

a more extensive vocabulary size, so grapheme representations provide too little information for the network. Also, the VITS models were trained using stressed text labels, so feeding them a text with no accentuation may have significantly impacted their performance. On the other hand, 3 additional accentuation tokens seemingly fix the problem. That proves the central claim of this work - stressed text improves the performance of speech synthesis systems based on neural networks for higher-degree phonemic orthography languages.

Another interesting observation is that the ground-truth samples of the Vytautas dataset (table 22) score rather poorly. That may be because the participants also considered how much they preferred each speaker when rating the samples. Overall, the respondents seem to prefer certain voices or speakers over others. Therefore we can conclude that it is imperative to consider the speaker's voice seriously before collecting a speech dataset for TTS model training.

Finally, the mean opinion score evaluation shows that overall the models trained using the Aurimas (table 23), following Giedrius (table 24), datasets were preferred the most. The models trained on the Vytautas dataset (table 22) scored significantly lower. It could be argued that this is since other speakers were preferred more by the respondents, but we can observe from the MOS tables that the best model trained on the Vytautas dataset lags behind the ground-truth score the most compared to the other two datasets. As described earlier, the Vytautas dataset was not validated and, therefore, low quality. This fact concludes that the quality of a dataset is of great importance when training TTS neural networks.

5.2. Lithuanian Speech Synthesizers Evaluation

The second survey aimed to compare the performance of the best-known Lithuanian speech synthesizers with the best model developed in this work. Here, 4 speech sources were evaluated: the Aurimas model (stressed) developed for this work, LIEPA's Edvardas [29], Microsoft's Leonas [30], and a synthesizer developed by the VDU (Vytautas Magnus University) scientists in collaboration with LRT (Lithuanian National Television and Radio), henceforth - the VDU synthesizer. The Aurimas, Microsoft's Leonas, and VDU synthesizers are based on neural networks, while the LIEPA's Edvardas is based on traditional concatenative synthesis methods.

36 unique sentences were used to generate samples for the survey by each source. In total, 144 samples were evaluated in the survey (36 utterances for each speech source). The duration of the samples range from 1 to 17 seconds. The survey takes approximately 30 minutes to complete. 12 respondents completed the survey in total. To reduce additional bias, the respondents that completed the first survey described above were not allowed to participate in this survey. Again, for reference, the link to the survey is provided in the footnotes [31]. The results of the survey are provided in table 25.

Before describing the survey setup and results, it is paramount to note that the VDU synthesizer

[29] https://liepa.rastija.lt/Ie%C5%A1kotuvas/Teksto-sintezatorius

[30] https://azure.microsoft.com/en-us/services/cognitive-services/text-to-speech

[31] https://questionpro.com/t/ASd5gZsxe5

is not openly available. The only way to listen to the speech synthesized by the model is to listen to the synthesized articles available at the official LRT news website [32]. This circumstance may introduce bias favoring the VDU synthesizer in the survey results. The samples generated for the survey by the VDU TTS system are the segments of the synthesized articles mentioned earlier. Therefore, other speech sources needed to generate the same sentences as the cut article segments.

Some subtle noises like mouse-clicking are heard in the complete article recordings that the VDU model synthesized. In addition, in some cases, the tone of the speaker changes in the middle of a spoken utterance. That may lead to assuming that the synthesized text was edited or the utterances that comprise the full recording were re-synthesized at the least. The utterances generated by the other three evaluated synthesizers (LIEPA's Edvardas, Microsoft's Leonas, and Aurimas model) were not edited or re-synthesized.

Furthermore, the VDU synthesizer was developed for the media domain to synthesize the text of the news articles specifically. In contrast to the VDU synthesizer, the Aurimas model was trained to synthesize more expressive language that may resemble casual conversations or audiobooks. Since all the utterances used to generate speech samples for the survey use text from actual news articles, the respondents may have assumed that the synthesizers were meant to speak in a formal tone representing a news reporter. This circumstance may have swayed their preference toward the VDU synthesizer since the tone of the speech synthesized by the Aurimas model is informal.

Table 25. Comparison of evaluated mean opinion scores with 95% confidence intervals. Four speech sources were evaluated: the stressed Aurimas model described in this work and three other Lithuanian synthesizers selected either because of their popularity or performance: LIEPA's Edvardas, Microsoft's Leonas, and a synthesizer developed by VDU (Vytautas Magnus University) scientists in collaboration with LRT (Lithuanian National Television and Radio), henceforth - VDU synthesizer.

Model	**MOS $\pm$CI**
Aurimas	4.27 $\pm$0.07
LIEPA Edvardas	1.72 $\pm$0.08
Microsoft's Leonas	4.01 $\pm$0.08
VDU	4.30 $\pm$0.07

We can see that the participants have chosen the VDU as their favorite synthesizer. According to the evaluation results, the VDU synthesizer only slightly beats (by 0.03 points) the synthesizer developed in this work. Again, note that the survey results may be biased in favor of the VDU synthesizer, and the actual preference results may differ in a different evaluation environment. However, by observing the results, we can see that all the synthesizers perform rather well except for the LIEPA synthesizer, which is relatively outdated and based on concatenative synthesis methods.

For a final note, this survey's results are not accurate as some good practices of conducting a MOS survey were violated when creating it. However, the survey was not meant to provide accurate results and was meant to give a rough estimate of how much the current Lithuanian synthesizers

[32] www.lrt.lt

differ in terms of quality. Nevertheless, the survey provided a reasonable estimate of user preference over a speech synthesizer in a news articles domain.

5.3. Summary

The MOS evaluation of multiple TTS neural network models proved the central claim of this work - stressed text improves the TTS model performance for higher-degree phonemic orthography languages. Also, the following other conclusions were drawn:

- The preferability of a trained TTS neural network depends significantly on a speaker that voices the recordings of a dataset that is used to train the model.
- The quality of a dataset is of great importance when training TTS neural networks.

6. Conclusions

This master's thesis work proposes an approach to using stressed text instead of phonemes or graphemes as TTS neural network inputs when training models for higher-degree phonemic orthography languages. Two TTS neural network architectures were used to train the models on Lithuanian language datasets with and without stressed text labels. These architectures are Tacotron 2 and VITS. The experimental results prove the central hypothesis of this work - using stressed text labels improves the pronunciation of speech synthesized by neural network models. The link to speech samples synthesized by the developed models may be found here [33].

Three single-speaker Lithuanian language speech corpora were collected to train the neural network models needed for the experiments: Aurimas, Giedrius, and Vytautas totaling 6, 27, and 92 hours of speech data, respectively. The Aurimas and Giedrius datasets were processed and manually validated for quality, but some processing and validation steps were skipped while collecting the Vytautas dataset. Therefore, the dataset contains many outliers.

Multiple Lithuanian speech synthesis neural network models were trained using earlier datasets. The experiments resulted in 9 trained TTS neural networks: six Tacotron 2 (stressed and non-stressed) and 3 VITS (stressed) models. A MOS survey was conducted to evaluate the trained models. The base Lithuanian language Tacotrons 2 and VITS models are open-sourced for the reader.

The results of evaluating the models show that using a low-quality dataset for TTS neural network training considerably impacts the performance of the resulting speech synthesizer. So, promising results were not expected from the models trained on the Vytautas dataset, and they were used mainly as base models. The base models were fine-tuned on better quality Aurimas and Giedrius datasets. The experimental results indicate that training TTS neural network models on a larger but lower-quality dataset and fine-tuning on smaller but higher-quality datasets decreases the

[33] https://arnasrad.github.io/lithuanian_synthesis_samples/index.html

time required to train the models. Lastly, it was concluded that it is imperative to choose a speaker with good prosody and other speech features when collecting a single-speaker speech corpus for the TTS system use case.

In addition, to explore future work possibilities for the topic, the initial experimental results of training a neural network-based stressing model are provided. Labels extracted from the collected speech corpora were used as a dataset in the automatic stressing model experiment. The data was used to train models of two neural network architectures: a multi-layer LSTM network and a BERT model modified for token classification. Although the LSTM experiment showed relatively good performance in terms of accuracy metric (93.4% accuracy), both experiments failed to produce valuable results. Therefore the model was not included in the TTS pipeline and will require further research.

In conclusion, using stressed text as the inputs to the Lithuanian TTS neural network models, the speech produced by the best synthesizer (Aurimas) developed during this master's thesis work generates near-human sounding speech. The MOS evaluation results comparing the model to other best-known Lithuanian speech synthesizers suggest that the Aurimas model performs the best. However, further work is needed to create a high-accuracy automatic accentuation model if the synthesizer is to be used for general use cases.

7. Future work

Using stressed text to train TTS neural networks provides a significant performance boost to TTS neural networks. However, the models are not provided with a pre-processing component that automatically stresses the input text. When creating the datasets described in this work, the labels were accentuated manually using a third-party tool developed by VDU. A text also needs to be stressed manually to perform the model inference. Such workflow is highly inconvenient if a synthesizer is to be used in a production environment. So it is essential to improve the performance of the automatic stressing model, of which the initial experiments and results are described in the earlier sections. Below, the authors' thoughts and notes on possible future work directions to improve the performance of the stressing model are provided

Try more hyperparameter combinations. Analyze what hyperparameters impact the performance the most, try to find the optimal set of hyperparameters.

Improve the LSTM architecture. The current LSTM architecture is trivial: it is straightforwardly composed of embedding, LSTM, and linear layers. Additional components, like a dropout layer or different combinations and ordering of the layers, may improve the network performance. Using a bi-directional version of the LSTM network is bound to be helpful, too, since the model may learn to capture word boundaries and infer that only one pitch accent should be assigned to a word. Also, it may be worthwhile to train a similar model but using GRUs instead of LSTM units.

Try out different loss functions. Currently, the system works at a character level, calculating the cost by comparing whether each predicted token corresponds to label tokens. On the other hand,

the objective is to penalize the model when it incorrectly predicts the letter position or the type of accent assigned at the word level. A custom loss function may be required for the system to work at the character level.

Fix the BERT model. In theory, Transformer networks should perform better than a trivial LSTM network, especially if it is uni-directional. There might be flaws in the current architecture of the BERT stressing model. For example, the format in which the input/label pairs are fed to the BERT model may be incorrect. Further analysis of the model is required.

Try word-level or word-piece-level tokens. Use the Tokenizer code infrastructure provided by HuggingFace to train the input and target tokenizers. The BERT model is known to work best with word-piece tokens, so doing the modification may improve the model's performance considerably. In this case, publicly available pre-trained language models like LitLat [34] [UR20] may be utilized to further increase the model's performance.

Make use of a pre-trained BERT model. BERT models are also known to demonstrate the best performance when a pre-trained model is used for transfer learning or fine-tuning in a new domain.

Reformulate the problem solved by the stressing model. Currently, a token classification task is performed. An important observation is that most of the words in Lithuanian (and other higher-degree of phonemic orthography languages) are accentuated unambiguously. The main problem is the pronunciation of homographs - words that may be pronounced differently depending on the context (nearby words in the sentence). By collecting a list of unambiguously-stressed words and a list of homographs, the problem may be reformulated to use the neural network to predict a masked token in a sequence. In this case, the masked token would be a stressed homograph. Nearby words would give context to the model. If the model would learn to assign a correct pitch accent to a correct letter of a homograph, the other words in the sentence that are stressed unambiguously could be stressed programmatically.

[34] https://huggingface.co/EMBEDDIA/litlat-bert

References

[ABD+19] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, et al. Common voice: a massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

[Aka98] Hirotugu Akaike. Autoregressive model fitting for control. In *Selected Papers of Hirotugu Akaike*, pp. 153–170. Springer, 1998.

[Anb10] Tomas Anbinderis. Automatic stressing of lithuanian text using decision trees. *Information Technology and Control*, 39(1), 2010.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[Bla] E. Blaževič. Nuo šiol portale lrt.lt visus straipsnius galima ne tik skaityti, bet ir klausyti. URL: `https://www.lrt.lt/naujienos/tavo-lrt/15/1340747/nuo-siol-portale-lrt-lt-visus-straipsnius-galima-ne-tik-skaityti-bet-ir-klausyti`.

[Chu86] Kenneth Church. Stress assignment in letter to sound rules for speech synthesis. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 11, pp. 2423–2426. IEEE, 1986.

[CLS+20] Michael Chinen, Felicia SC Lim, Jan Skoglund, Nikita Gureev, Feargus O'Gorman, and Andrew Hines. Visqol v3: an open source production ready objective speech and audio metric. In *2020 twelfth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6. IEEE, 2020.

[CVG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[DBJ+09] Qing Dou, Shane Bergsma, Sittichai Jiampojamarn, and Grzegorz Kondrak. A ranking approach to stress prediction for letter-to-phoneme conversion. In *Proceedings of the Joint Conference of the 47th annual meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 118–126, 2009.

[DCL+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[DDB+20] Jeff Donahue, Sander Dieleman, Mikołaj Bińkowski, Erich Elsen, and Karen Simonyan. End-to-end adversarial text-to-speech. *arXiv preprint arXiv:2006.03575*, 2020.

[DK09] Ineta Dabašinskienė and Laura Kamandulyte. Corpora of spoken lithuanian. *Eesti Rakenduslingvistika Ühingu aastaraamat*, 5:67–77, 2009.

[DPP+96] Thierry Dutoit, Vincent Pagel, Nicolas Pierret, François Bataille, and Olivier Van der Vrecken. The mbrola project: towards a set of high quality speech synthesizers free of use for non commercial purposes. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 3, pp. 1393–1396. IEEE, 1996.

[FM06] David Frontini and Mario Malcangi. Neural network-based speech synthesis. *DSP Application Day*, 2006.

[GL84] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, 32(2):236–243, 1984.

[HB96] Andrew J Hunt and Alan 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, pp. 373–376. IEEE, 1996.

[HCC+14] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, et al. Deep speech: scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[Hoy18] Matthew B Hoy. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[HS13] Keith Hall and Richard Sproat. Russian stress prediction using maximum entropy ranking. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 879–883, 2013.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HWH+19] Wen-Chin Huang, Yi-Chiao Wu, Hsin-Te Hwang, Patrick Lumban Tobing, Tomoki Hayashi, Kazuhiro Kobayashi, Tomoki Toda, Yu Tsao, and Hsin-Min Wang. Refined wavenet vocoder for variational autoencoder based voice conversion. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5. IEEE, 2019.

[IC01] Akemi Iida and Nick Campbell. A database design for a concatenative speech synthesis system for the disabled. In *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis*, 2001.

[IS95] Naoto Iwahashi and Yoshinori Sagisaka. Speech spectrum conversion based on speaker interpolation and multi-functional representation with weighting by radial basis function networks. *Speech Communication*, 16(2):139–151, 1995.

[Ito17] Keith Ito. Tacotron. `https://keithito.github.io/audio-samples/`, 2017.

[JYB+22] Rishabh Jain, Mariam Yiwere, Dan Bigioi, Peter Corcoran, and Horia Cucu. A text-to-speech pipeline, evaluation methodology, and initial fine-tuning results for child speech synthesis. *IEEE Access*, 2022.

[Kas+16] Pijus Kasparaitis et al. Lietuviško balso sintezatorių kokybės vertinimas. *Kalbų studijos*, (28):80–91, 2016.

[Kas00] Pijus Kasparaitis. Automatic stressing of the lithuanian text on the basis of a dictionary. *Informatica*, 11(1):19–40, 2000.

[Kas01a] P Kasparaitis. Lietuvių kalbos kompiuterinis sintezatorius „aistis “. *Garso korta 2001 (CD)*, 2001.

[Kas01b] Pijus Kasparaitis. Automatic stressing of the lithuanian nouns and adjectives on the basis of rules. *Informatica*, 12(2):315–336, 2001.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KB91] Wolfgang von Kempelen and Herbert E Brekle. Mechanismus der menschlichen sprache nebst beschreibung einer sprechenden maschine. *Grammatica universalis*, 1791.

[KKK+20] Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: a generative flow for text-to-speech via monotonic alignment search. *arXiv preprint arXiv:2005.11129*, 2020.

[KKS21] Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. *arXiv preprint arXiv:2106.06103*, 2021.

[KKW18] Karolina Kuligowska, Paweł Kisielewicz, and Aleksandra Włodarz. Speech synthesis systems: disadvantages and limitations. *Int J Res Eng Technol (UAE)*, 7:234–239, 2018.

[Kla80] DH Klatt. Klsyn: a formant synthesis program, 1980.

[Kra81] Christian Gottlieb Kratzenstein. *Tentamen resolvendi problema ab Acad. Petropolit. 1780 propositu qualis sit natura litterarum vocalium a, e, i, o, u*. 1781.

[Lem+99] Sami Lemmetty et al. *Review of speech synthesis technology*. MA thesis, 1999.

[LLL+19] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33 of number 01, pp. 6706–6713, 2019.

[LTK+18] Sigita Laurinčiukaitė, Laimutis Telksnys, Pijus Kasparaitis, Regina Kliukienė, and Vilma Paukštytė. Lithuanian speech corpus liepa for development of human-computer interfaces working in voice recognition and synthesis mode. *Informatica*, 29(3):487–498, 2018.

[MC90] Eric Moulines and Francis Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6):453–467, 1990.

[MJV+98] Pedro J Moreno, Christopher F Joerg, Jean-Manuel Van Thong, and Oren Glickman. A recursive algorithm for the forced alignment of very long audio segments. In *ICSLP*, vol. 98, pp. 2711–2714, 1998.

[Mos20] Anna Mosolova. The algorithm of automatic accentuation with respect to the speaking norm of a given, 2020.

[MPvS07] Taniya Mishra, Emily Tucker Prud'hommeaux, and Jan PH van Santen. Word accentuation prediction using a neural net classifier. In *SSW*, pp. 246–251. Citeseer, 2007.

[MTK+97] Takashi Masuko, Keiichi Tokuda, Takao Kobayashi, and Satoshi Imai. Voice characteristics conversion for hmm-based speech synthesis system. In *1997 IEEE international conference on acoustics, speech, and signal processing*, vol. 3, pp. 1611–1614. IEEE, 1997.

[NLS+11] Robert Neßelrath, Chensheng Lu, Christian H Schulz, Jochen Frey, and Jan Alexandersson. A gesture based system for context–sensitive interaction with smart homes. In *Ambient Assisted Living*, pp. 209–219. Springer, 2011.

[ODZ+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: a generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[OLB+18] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, et al. Parallel wavenet: fast high-fidelity speech synthesis. In *International conference on machine learning*, pp. 3918–3926. PMLR, 2018.

[PCP+15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

[PPC18] Wei Ping, Kainan Peng, and Jitong Chen. Clarinet: parallel wave generation in end-to-end text-to-speech. *arXiv preprint arXiv:1807.07281*, 2018.

[PPS+19] Kainan Peng, Wei Ping, Zhao Song, and Kexin Zhao. Parallel neural text-to-speech, 2019.

[PVC19] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: a flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

[Rec94] ITUT Recommendation. Telephone transmission quality subjective opinion tests. a method for subjective performance assessment of the quality of speech voice output devices, 1994.

[RFZ+11] Flávio Ribeiro, Dinei Florêncio, Cha Zhang, and Michael Seltzer. Crowdmos: an approach for crowdsourcing mean opinion score studies. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 2416–2419. IEEE, 2011.

[RHQ+20] Yi Ren, Chenxu Hu, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: fast and high-quality end-to-end text-to-speech. *arXiv preprint arXiv:2006.04558*, 2020.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[RLT+20] Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. A study of non-autoregressive model for sequence generation. *arXiv preprint arXiv:2004.10454*, 2020.

[RR02] Algimantas Rudzionis and Vytautas Rudzionis. Lithuanian speech database ltdigits. In *LREC*, 2002.

[RRK21] Arnas Radzevičius, Aistis Raudys, and Pijus Kasparaitis. Speech synthesis using stressed sample labels for languages with higher degree of phonemic orthography. In *International Conference on Information and Software Technologies*, pp. 378–387. Springer, 2021.

[RRT+19] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: fast, robust and controllable text to speech. *arXiv preprint arXiv:1905.09263*, 2019.

[SBV+06] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, and Sam Britton. Real-time corpus-based concatenative synthesis with catart. In *9th International Conference on Digital Audio Effects (DAFx)*, pp. 279–282, 2006.

[Šef06] Tomaž Šef. Automatic accentuation of words for slovenian tts system. In *Proceedings of the 5th WSEAS International Conference on Signal Processing*, pp. 155–160, 2006.

[Sga87] Petr Sgall. Towards a theory of phonemic orthography. *Orthography and phonology*:1–30, 1987.

[SPW+18] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[SW12] F Reena Sharma and S Geetanjali Wasson. Speech recognition and synthesis tool: assistive technology for physically disabled persons. *International Journal of Computer Science and Telecommunications*, 3(4):86–91, 2012.

[TNT+13] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech synthesis based on hidden markov models. *Proceedings of the IEEE*, 101(5):1234–1252, 2013.

[TYM+00] Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Speech parameter generation algorithms for hmm-based speech synthesis. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, vol. 3, pp. 1315–1318. IEEE, 2000.

[UR20] Matej Ulčar and Marko Robnik-Šikonja. Litlat bert, 2020.

[URB+15] Raphael Ullmann, Ramya Rasipuram, Hervé Bourlard, et al. Objective intelligibility assessment of text-to-speech systems through utterance verification. In *Proceedings of Interspeech*, number CONF, 2015.

[Vai+01] Martti Vainio et al. Artificial neural network based prosody models for finnish text-to-speech synthesis, 2001.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[Wil87] Briony Williams. Word stress assignment in a text-to-speech synthesis system for british english. *Computer Speech & Language*, 2(3-4):235–272, 1987.

[WRS+21] Steven Euijong Whang, Yuji Roh, Hwanjun Song, and Jae-Gil Lee. Data collection and quality challenges in deep learning: a data-centric ai perspective. *arXiv preprint arXiv:2112.06409*, 2021.

[WSB+21] Ron J Weiss, RJ Skerry-Ryan, Eric Battenberg, Soroosh Mariooryad, and Diederik P Kingma. Wave-tacotron: spectrogram-free end-to-end text-to-speech synthesis. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5679–5683. IEEE, 2021.

[WSS+17] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, et al. Tacotron: towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[WWT+12] Linfang Wang, Lijuan Wang, Yan Teng, Zhe Geng, and Frank K Soong. Objective intelligibility assessment of text-to-speech system using template constrained generalized posterior probability. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[YSK20] Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. Parallel wavegan: a fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6199–6203. IEEE, 2020.

[YYZ20] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: taking yelp review dataset as an example. In *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, pp. 98–101. IEEE, 2020.

[ZDC+19] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: a corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

[ZLD21] Xiao Zhou, Zhen-Hua Ling, and Li-Rong Dai. Unitnet: a sequence-to-sequence acoustic model for concatenative speech synthesis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2643–2655, 2021.

[ZTB09] Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *speech communication*, 51(11):1039–1064, 2009.