VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

DATA SCIENCE STUDY PROGRAMME

Master's Thesis

Transformer-Based Lithuanian Text Stressing for Speech Synthesis

Transformeriu grįstas lietuviško teksto kirčiavimas kalbos sintezei

Robert Mackevič

Supervisor : Assoc. Prof. Gintautas Tamulevičius

Vilnius
2026

Acknowledgements

I would like to express my sincere gratitude to *UAB "Taikomasis dirbtinis intelektas"* (AAI-Labs) for providing the stress-annotated Lithuanian text data as well as high-quality studio speech recordings that made this research possible. Their support and willingness to share valuable resources contributed significantly to the successful completion of this work.

I would also like to thank my supervisor, Assoc. Prof. Gintautas Tamulevičius, for his guidance, insightful comments, and constructive feedback throughout the course of this project.

Summary

This thesis explores the integration of automatic stress prediction into Lithuanian text-to-speech (TTS) systems by developing and evaluating a Transformer-based model for assigning stress marks in written text. Lithuanian is a stress-sensitive and morphologically rich language where stress is typically omitted in writing and must be inferred from context, which poses challenges for natural-sounding speech synthesis. The proposed approach introduces a modular neural network that explicitly assigns stress marks prior to synthesis, aiming to improve stress realization in synthesized speech.

Two core hypotheses were investigated: (1) that a neural network model can outperform prominent rule-based stressing tools (such as "Kirčiuoklis" developed by Vytautas Magnus University) in stress prediction by leveraging sentence-level context, and (2) that using automatically stressed text as input leads to more context-aware stress realization in synthesized speech compared to using plain-text or phonemic input. Experimental results support these claims. The Transformer model achieved higher stress prediction accuracy than the prominent rule-based tools and was capable of generalizing to unseen words during inference. However, in the case of contextual awareness evaluation, the rule-based "Kirčiuoklis" tool outperformed the Transformer, revealing limitations in the model's ability to effectively leverage context.

Nevertheless, TTS models trained on stress-annotated text produced significantly more accurate stress realization in audio compared to plain-text or phoneme-based models. This confirms the practical benefit of explicit stress modeling, especially for low-resource languages. The findings suggest that while end-to-end TTS remains the long-term goal, a modular approach to stress assignment offers immediate and scalable improvements for Lithuanian speech synthesis. Future work should focus on expanding the annotated dataset and refining the model architecture to enhance context awareness and generalization.

Keywords: Transformer, Lithuanian text stressing, speech synthesis, natural language processing, context awareness.

Santrauka

Šiame darbe nagrinėjamas automatinio kirčiavimo integravimas į lietuvių kalbos tekstų sintezės (TTS) sistemas, sukuriant ir įvertinant transformeriu architektūros grįstą modelį, skirtą lietuviško teksto kirčiavimui. Lietuvių kalboje kirčiavimas yra reikšminis ir glaudžiai susijęs su žodžių morfologija, tačiau rašytiniuose tekstuose kirčio ženklai paprastai nenaudojami. Dėl to TTS modeliams kirčio vietą dažnai tenka nustatyti netiesiogiai, remiantis kontekstu, o tai apsunkina natūraliai skambančios kalbos sintezę. Šiame darbe siūlomas metodas įveda papildomą neuroninį tinklą, kuris prieš sintezę automatiškai pažymi kirčius, taip siekiant pagerinti jų perteikimą sugeneruotoje kalboje.

Buvo keliami du pagrindiniai hipotezių teiginiai: (1) kad neuroninis modelis, naudodamas sakinio lygmens kontekstą, gali pranokti pagal tikslumą taisyklėmis grįstus kirčiavimo įrankius (pvz., Vytauto Didžiojo universiteto sukurtą „Kirčiuoklį“) ir (2) kad naudojant automatiškai sukirčiuoto teksto įvestį sintezėje bus pasiekiama tikslesnė kirčių tarimo kokybė sintezuotuose įrašuose palyginus su foneminiais ar paprasto teksto įvesčių tipais. Eksperimentiniai rezultatai šias hipotezes patvirtino. Transformerio modelis pasiekė aukštesnį kirčio žymėjimo tikslumą nei žinomi taisyklėmis paremti įrankiai ir sugebėjo apdoroti anksčiau nematytus žodžius. Visgi, vertinant kontekstinio jautrumo aspektą, taisyklėmis grįstas „Kirčiuoklis“ pasirodė geriau nei neuroninis modelis, atskleisdamas modelio ribotumą efektyviai pasinaudoti sakinio kontekstu.

Nepaisant to, TTS modeliai, apmokyti su kirčiuotu tekstu, žymiai tiksliau išreiškė kirčius sintezuotoje kalboje nei modeliai, naudoję paprastą tekstą ar fonemas. Tai patvirtina kirčiavimo prieš sintezę praktinę naudą, ypač mažai išteklių turinčių kalbų kontekste. Rezultatai rodo, kad nors ilgalaikis tikslas tebėra vientiso tipo (end-to-end) sintezė, modulinis kirčiavimo metodas jau dabar leidžia pasiekti reikšmingą ir lengvai plečiamą sintezės kokybės pagerėjimą. Ateities darbai turėtų koncentruotis į žymėtų duomenų plėtrą ir modelio architektūros tobulinimą, siekiant geresnio konteksto suvokimo ir bendresnio pritaikomumo.

Raktiniai žodžiai: Transformer, lietuviško teksto kirčiavimas, kalbos sintezė, natūralios kalbos apdorojimas, konteksto suvokimas.

Contents

Introduction

Speech synthesis, also commonly known as text-to-speech (TTS), is the artificial generation of human speech from digital text. Today, TTS technology plays a critical role in various applications, including virtual assistants, smart home systems, navigation tools, language learning platforms, audiobooks, and accessibility aids for individuals with visual or reading impairments.

The fundamental challenge of speech synthesis is to convert a relatively compact textual input into a detailed and nuanced speech waveform. This conversion must ensure intelligibility, naturalness, and expressiveness, closely emulating actual human speech. Historically, numerous methods have been proposed, dating back to the first mechanical speech-emulating devices of the late 18th century [24]. Early inventions generated only rudimentary vowel and consonant sounds, whereas modern computer-based TTS systems produce speech that is basically indistinguishable from natural human voices.

Phonemes and graphemes are essential components in the field of speech synthesis, as they represent the fundamental units of spoken and written language, respectively. A phoneme is the smallest segment of sound that can distinguish one word from another in a given language, i.e., the difference between /k/ and /g/ in "coat" and "goat." Graphemes, on the other hand, are the written symbols (such as letters or letter combinations) that represent these sounds in a specific writing system. While phonemes directly reflect how a word is pronounced, graphemes are based on orthographic rules and often represent pronunciation only approximately. This distinction becomes especially important in languages with complex or irregular spelling systems, where the same grapheme may correspond to multiple phonemes, or where pronunciation cannot be accurately inferred from spelling alone. As a result, converting written text into speech requires not just mapping letters to sounds, but understanding the linguistic rules that govern pronunciation, making accurate grapheme-to-phoneme (G2P) conversion a non-trivial step in speech synthesis systems.

Traditionally, statistical parametric methods dominated speech synthesis. However, in recent years, neural network-based approaches have emerged as superior, generating speech of higher naturalness and intelligibility. Neural networks learn directly from the data, reducing the need for extensive linguistic knowledge and feature engineering. However, these models typically require large amounts of speech data, posing challenges for languages lacking significant resources, often referred to as low-resource languages. Most neural TTS research targets high-resource languages like English, benefiting from abundant datasets and G2P converters. Low-resource languages, in contrast, face difficulties due to the scarcity of publicly available datasets and linguistic resources, making it challenging to replicate successful TTS methodologies. Given these constraints, alternative approaches become necessary for low-resource languages such as Lithuanian.

Lithuanian language, characterized by high phonemic orthography, presents a unique opportunity where graphemes closely correspond to phonemes, which simplifies pronunciation

modeling. However, Lithuanian pronunciation includes additional prosody nuances such as vowel length and word stress, typically marked with stress marks (in Lithuanian, *"kirčio ženklai"*). Lithuanian is a free-stress, pitch-accent language – the position of stress in a word is not fixed by syllable count, and stressed syllables can carry different tonal accents (falling vs. rising) [12]. Unlike languages with fixed stress, Lithuanian's word stress must often be deduced from lexical knowledge. Standard written Lithuanian does not mark stress, so automatic stressing is needed for applications like TTS to insert the correct stress marks on vowels. This is a non-trivial task because stress placement depends on morphological word forms and can even distinguish meanings for homographs (words spelled the same but pronounced differently). For example, *"pãstato"* (noun, genitive singular, masculine) means "of the building", while *"pastãto"* (verb, present tense, third person) means "he/she builds" or "he/she places something". In context, this contrast is evident in a sentence such as: *"Jonas mašiną pastato už pastato"*, which translates to "Jonas parks his car behind the building".

This thesis explores the use of Lithuanian stress annotated text as an alternative to solely grapheme- or phoneme-based representations for speech synthesis. The core idea is to utilize a speech synthesis system based on stress marks, leveraging their accessibility and the high degree of phonemic orthography in the Lithuanian language. Stressing is universally taught in Lithuanian schools, making stress marks familiar to the general population, whereas phonemic transcriptions require specialized linguistic knowledge, limiting their practical use and the scalability of labeled data collection. This accessibility could make it substantially easier to produce, verify, and expand high-quality training data for TTS systems compared to phoneme-based approaches. From a computational perspective, stress-based synthesis also offers a slight advantage. Phonemization typically requires a much larger vocabulary, such as the International Phonetic Alphabet (IPA) with more than 100 symbols, while Lithuanian stress marks comprise only three symbols, which can be appended to the regular Lithuanian alphabet.

Although the use of stress marks in speech synthesis is not a new concept, the main contribution of this work is the development of a neural network–based model capable of automatically assigning Lithuanian stress marks while incorporating contextual information at the sentence-level. The second contribution of this study is the comparative evaluation of multiple TTS pipeline configurations using different forms of input: plain graphemic text, phonemized text, and text annotated with stress marks. This comparison aims to determine which representation leads to the most accurate stress realization in synthesized Lithuanian speech.

Despite the critical role that correct stress placement plays in Lithuanian, fully automated, data-driven approaches to Lithuanian stress assignment remain underexplored. Existing neural methods have either not been sufficiently documented in academic research or have failed to produce consistently reliable results due to methodological flaws. At present, the most widely used solution is a rule-based system developed by Vytautas Magnus University (hereafter referred to as VDU), which relies on a manually constructed lexicon and

morphological rules. While generally effective, this system has notable limitations: its static, rule-based design makes it poorly suited for handling context-dependent stress patterns and out-of-vocabulary words. Moreover, it requires manual input, preventing seamless integration into fully automated pipelines. **The hypothesis of this thesis is therefore twofold:** (1) that a neural network–based model, by incorporating sentence-level context, can achieve more accurate Lithuanian stress assignment than existing prominent rule-based methods; and (2) that the integration of automatically stressed text into TTS pipelines leads to more faithful and natural stress realization in synthesized speech compared to pipelines using unstressed graphemic or phonemic input.

Given the preliminary nature of this research, the primary focus is placed on the development and validation of the stress assignment model itself. To assess the first part of the hypothesis, the model's stress prediction accuracy is evaluated quantitatively by comparing its output against a human-annotated reference dataset. These results are also benchmarked against existing stress assignment tools to establish comparative performance. Following this, a set of speech synthesis models are trained using three distinct types of input representations: plain graphemic text, phonemic transcriptions, and text annotated with stress marks. The resulting synthesized outputs are then analyzed in terms of the accuracy of stress realization to determine whether incorporating stress-annotated input yields a measurable improvement in the correctness of synthesized Lithuanian speech.

1 Literature review

This literature review explores existing research, methodologies and tools relevant to Lithuanian text stressing, stress-based speech synthesis, and recent advancements in neural models for natural language processing (NLP). The goal is to review prior efforts, highlighting key challenges and gaps that this thesis seeks to address.

1.1 Background on Lithuanian Stressing Automation

The earliest work on automatic Lithuanian text stressing was initiated by P. Kasparaitis in 2000, who proposed a dictionary-based word-level stressing method that decomposed words into stems, prefixes, and suffixes, stored alongside metadata to reconstruct grammatical forms and assign stress [10]. This system, covering over 62,000 entries across various word classes, achieved correct stress of words in 82.57% of journalism and 81.53% of fiction texts, with very low incorrect stress rates (0–0.2%). However, it struggled with unseen or ambiguous words, 3–4% were out-of-vocabulary, and 13–17% had multiple possible stress patterns, highlighting its reliance on dictionary completeness and lack of contextual understanding. In 2001, Kasparaitis extended this work by introducing formal stress assignment rules targeting morphologically complex or derived forms not covered by the dictionary [11]. These rules, integrated into the same data structure as dictionary stems, were either manually encoded or automatically generated from a lexicon of over 50,000 stems. Evaluation on the same corpus showed a slight improvement in correct stress rates, 83.05% for journalism and 82.73% for fiction texts, but also a rise in incorrect stress, reflecting the risk of rule overgeneralization. All in all, rule integration reduced dictionary size by over 60% and enhanced coverage, making the system more flexible for morphologically rich text, though at the cost of some accuracy.

In 2010, T. Anbinderis proposed a data-driven method for automatic Lithuanian word stressing using decision trees, relying solely on letter sequence patterns (beginnings, endings, and middles) to predict stress placement in a word [1]. Trained on nearly one million manually corrected examples, the best-performing model achieved 95.5% accuracy, only slightly below more complex morphology-based methods. This approach was notably fast, portable, and independent of linguistic annotation, though it required a large stressed corpus and many decision rules. In a separate study, Anbinderis also explored mathematical and probabilistic modeling for stress assignment, categorizing words by stress predictability and proposing frequency-based models that achieved 75–80% accuracy for nouns, adjectives, and numerals [2]. While these models were simple and well-suited as preliminary filters or lightweight alternatives, they struggled with ambiguous cases, underscoring the need for hybrid approaches that combine statistical, rule-based, and contextual information.

Kazlauskienė, Raškinis, and Vaičiūnas (2010) developed a rule-based algorithm for Lithuanian stress assignment, using detailed morphological analysis and lexicons to accurately place stress marks at the word level [14]. This work became the basis for “Kirčiuoklis”,

a free online tool maintained by Vytautas Magnus University (VDU) and last updated in 2021 [32]. The "Kirčiuoklis" tool is part of a larger language-learning platform, created under a project that promotes collaboration between Baltic studies centers abroad and Lithuanian research institutions. The "Kirčiuoklis" module enables users to automatically stress entire texts, accommodating nearly all stress variants recognized in major dictionaries and VLKK (Commission of the Lithuanian Language) guidelines. However, as the tool is fundamentally rule-based, it has limited capacity to handle context-dependent stress assignments and is unable to stress words not present in its internal vocabulary.

While the VDU "Kirčiuoklis" remains the most prominent Lithuanian stress-marking system, several other solutions exist, although they are not documented in academic literature and are not as prominent. The most notable alternative efforts are the automatic stressing models and tools developed by Aleksas Pielikis. Although no formal publications describing these systems are available, their implementation and experimental setup are documented through publicly available GitHub repositories. Two of such repositories provide code for training and evaluating Lithuanian Transformer-based stressing models using the Tensor2Tensor (T2T) framework. Tensor2Tensor is a deep learning library developed by the Google Brain team to facilitate research in sequence-to-sequence modeling, including neural machine translation [29]. While T2T has since been superseded by newer frameworks, it was widely used at the time of these experiments.

The first repository, last updated in 2018, implements a Lithuanian text stressing model using a Transformer architecture trained within the T2T and TensorFlow frameworks. The system formulates stress-mark assignment as a monolingual sequence-to-sequence text translation task, mapping unstressed Lithuanian text to stress-annotated output (LT $\rightarrow$ LT+stress) [19]. Training data consist of parallel pairs of unstressed and stressed text, and inference is performed using standard Transformer decoding with beam search. Model evaluation is conducted using a separate dataset derived from *"Kirčiuotų tekstų chrestomatija"*, a collection of 50 stress-annotated literary texts by Lithuanian authors spanning multiple genres [13]. The second repository, last updated in 2019, extends this approach by addressing joint text normalization and stress assignment as a single sequence-to-sequence problem [20]. In this setup, the Transformer model learns to convert raw Lithuanian text into a normalized and stress-annotated form, simultaneously standardizing non-canonical text (e.g., numerals) and inserting appropriate stress marks. However, neither repository reports quantitative accuracy metrics, making direct comparison with other stressing tools difficult.

In addition to neural approaches, Pielikis also developed the "Phonology Engine", a Lithuanian phonological processing and stressing library originally extracted from the LIEPA[1] speech synthesizer. This tool performs text normalization, word and phrase segmentation, syllabification, and stress assignment based on morphological analysis and handcrafted linguistic rules [21]. While this rule-based approach is interpretable and linguistically

[1]LIEPA, short for *"LIEtuvių šneka valdomos PAslaugos"*, is a Lithuanian national research and development project focused on creating Lithuanian speech technologies to enable services controlled by spoken Lithuanian.

consistent, it may lack robustness in context-dependent or ambiguous cases such as other similar rule-based tools.

Another, less widely used stressing tool is *"Kirtis"*, an open-source, web-based Lithuanian stress-marking application developed with support from *UAB "Sistemium"* for the Computational Linguistics Centre [27]. It represents a practical, rule-based solution intended primarily for public and educational use rather than data-driven or neural modeling. However, its functionality is limited to stressing individual words rather than full sentence-level context, and its last major update was in 2018.

The most recent exploration of neural approaches to automatic Lithuanian text stressing was conducted by Radzevičius (2022) as part of his master's thesis on Lithuanian speech synthesis [22]. In a supplementary experiment, he developed and evaluated stress prediction models to serve as a pre-processing component for stress-aware TTS systems. Two neural architectures were explored: a sequence labeling model based on Long Short-Term Memory (LSTM) networks and a fine-tuned Bidirectional Encoder Representations from Transformers (BERT) model. Both models were trained on a manually annotated corpus of Lithuanian text with explicit stress markings, with the aim of predicting both the position and type of stress at the character level. The LSTM-based model was reported to achieve up to 93% character-level accuracy; however, it exhibited a critical flaw—over-predicting stress by assigning accent marks to nearly every vowel, rather than correctly identifying a single stressed syllable per word. The BERT-based model also underperformed in practical terms, showing poor generalization to unseen words, resulting in significant accuracy degradation outside the training data. As Radzevičius concluded, neither model proved sufficiently robust for deployment in a production TTS pipeline. The main limitations were attributed to the lack of sentence-level contextual modeling, the small size of high-quality annotated data, and the models' limited ability to capture Lithuanian's rich inflectional morphology. While the experiments demonstrated that neural networks can learn some stress patterns, the results highlighted the need for more sophisticated architectures and larger, contextually rich datasets to achieve reliable performance.

1.2 Stress-Based Speech Synthesis

Radzevičius, Raudys, and Kasparaitis (2021) introduced a stress-based speech synthesis method designed specifically for languages with a high degree of phonemic orthography, using Lithuanian as a case study [23]. Instead of relying on traditional phoneme-based input, their approach involved augmenting training data with UTF-8 stress marks (grave, acute, tilde) directly embedded in the raw text. This technique allows neural models to learn stress placement as part of the character embedding, bypassing the need for G2P conversions. The model architecture followed Nvidia's implementation of Tacotron 2 paired with the WaveGlow vocoder, fine-tuned on a 91-hour Lithuanian speech corpus where stress labels were semi-automatically assigned and manually verified. Results from MOS (Mean Opinion Score) evaluations showed that models trained with stressed labels performed notably better

(MOS 3.53) than those trained on raw text (MOS 2.61–2.62). While the top-performing model still lagged behind human speech quality (MOS 4.79), the stress-enhanced input clearly improved pronunciation and prosody.

In the same thesis by Radzevičius (2022), which also included the most recent neural text stressing experiments, a comparative analysis was conducted to evaluate the impact of stress-annotated input on the quality of synthesized speech [22]. The study compared models trained on manually stress-marked text with those trained on plain graphemic input using two popular TTS architectures: Tacotron 2 and VITS. Although automatic stress prediction models were developed as part of the research, they were not used in training due to insufficient accuracy. Instead, stress annotations were applied manually to ensure data quality. The results demonstrated that, across all configurations, VITS consistently outperformed Tacotron 2 in terms of Mean Opinion Score (MOS). Furthermore, models trained with stress-annotated input yielded noticeably higher MOS ratings compared to their plain-text counterparts.

Kasparaitis and Antanavičius (2023) investigated how different input alphabets affect the quality of end-to-end Lithuanian speech synthesis using the Tacotron 2 model [12]. They compared five input representations: full alphabet (including uppercase/lowercase), lowercase-only, accented (stressed) letters, a reduced set of accented letters, and letters with separate accent marks. All models were trained on the same Lithuanian speech corpus, and the synthesized speech was evaluated via human listening tests. Their experiments found that including stress marks in the input text significantly improved the naturalness and acceptability of the synthesized speech. Moreover, reducing the input alphabet size, such as removing distinctions between letters that always represent the same sound, had a modest positive effect. Of all representations tested, encoding accent marks as separate symbols provided the best results, even outperforming models that used accented letters. The study also confirmed that, while Tacotron 2 can partially learn Lithuanian stress patterns from plain text, explicit stress annotation is crucial for high-quality synthesis in morphologically complex, low-resource languages like Lithuanian.

The "Accentor" approach proposed by Geneva et al. addresses the problem of accurate stress placement in TTS systems, particularly for highly inflected and stress-variable languages such as Bulgarian [9]. The authors highlight that modern neural TTS models often struggle with stress prediction due to limited lexical coverage in available datasets, resulting in frequent stress errors. To overcome this, the authors introduce a two-stage pipeline: (1) a dedicated "Accentor" neural network is trained on a large, stress-annotated text corpus to predict and insert stress marks into raw text; (2) a controllable acoustic model is then trained to generate speech conditioned on these explicit stress annotations. The "Accentor" model uses FFT (Fast Fourier Transformation) blocks with multi-head self-attention and depth-wise separable convolutions, enabling effective use of contextual information to resolve ambiguities and predict stress placement. Trained on millions of utterances, the "Accentor" achieved a word-level stress error rate (SER) of 0.37% in evaluation, representing

a 12-fold reduction compared to standard FastPitch-based TTS systems. While this work focuses on Bulgarian, a similar methodology could be applicable to other languages with complex stress systems, including Lithuanian.

1.3 Transformer Models in Natural Language Processing

Early automatic text stressing systems relied on rule-based algorithms or shallow machine learning methods, which lacked the capacity to model long-range dependencies or capture the full context required for accurate stress assignment in morphologically rich languages such as Lithuanian [1, 11]. Recurrent neural networks and LSTMs mark some improvement in processing sequence data, but their sequential nature still limits their ability to attend to non-local context and makes them difficult to parallelize [25].

The introduction of the Transformer architecture represented a fundamental shift in NLP. By relying entirely on self-attention mechanisms, Transformers efficiently capture global dependencies within sequences and support fully parallel computation, making them highly effective for a wide range of tasks [30]. Transformer-based sequence-to-sequence models rapidly became the state-of-the-art for machine translation [18, 30], and other structured sequence prediction tasks. Their power lies in the attention mechanism (1), which allows each output token to be generated based on the entire input context, making them especially well-suited for tasks where the correct label (such as a stress mark) can depend on complex, sentence-level relationships.

The core operation of the Transformer is the scaled dot-product attention, mathematically defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, and V are the query, key, and value matrices, and d_k is the dimensionality of the key vectors. This mechanism enables the model to dynamically focus on relevant parts of the input sequence when producing each output token.

For morphologically complex and low-resource languages, Transformers have proven capable of learning robust representations from limited data, often outperforming older architectures [6]. In speech synthesis, recent studies have begun to adapt Transformer-based models for G2P conversion [8] and other sequence labeling tasks, achieving high accuracy in mapping text to linguistic features required for TTS. The flexibility and effectiveness of the Transformer architecture in these related tasks strongly support its application to Lithuanian text stressing, where the problem can be modeled as a sequence-to-sequence translation from plain text to stress-marked text.

Therefore, leveraging a Transformer for Lithuanian text stressing is well-motivated by both theoretical and empirical evidence: it enables the modeling of context-sensitive stress assignment, handles long-range dependencies, and has demonstrated success in closely related sequence labeling and low-resource NLP tasks.

1.4 Summary

Recent developments in neural and end-to-end speech synthesis have highlighted the importance of explicit stress annotation, particularly for languages with a high degree of phonemic orthography such as Lithuanian [12, 23]. Studies have demonstrated that providing stress-marked input can significantly improve the naturalness and intelligibility of synthesized speech. However, current neural approaches to automatic stress assignment—such as LSTM- and BERT-based models—have yet to achieve reliable accuracy. These models often suffer from issues like stress over-prediction or limited generalization beyond the training set, especially when faced with morphologically complex or unseen word forms [22]. Other neural approaches have not yet been rigorously tested and documented.

While rule-based systems such as VDU's "Kirčiuoklis" continue to perform well in many cases, their dependence on predefined morphological rules and dictionaries limits their ability to resolve context-dependent ambiguities or adapt to novel input.

A clear research gap remains: there is currently no robust, context-aware, automated Lithuanian text stressing model that is both accurate and suitable for integration into modern TTS pipelines. Most existing tools either lack sentence-level contextual understanding or have not demonstrated sufficient robustness for real-world deployment. Moreover, some recent neural approaches remain under-documented in academic literature, further limiting their reproducibility.

2 Methodology

2.1 Text Stressing Experiment

This section outlines the methods developed and employed for automatic Lithuanian text stress assignment. The methodology covers the complete experimental pipeline, beginning with a detailed description of the data sources and pre-processing procedures, followed by the construction and tokenization of the dataset. Subsequent subsections present the design and implementation of the Transformer model, training procedure, and inference algorithms. Quantitative and qualitative evaluation methods are then described, alongside the baselines and external methods used for comparison. Finally, the experimental environment and all training configurations are specified to ensure reproducibility.

2.1.1 Data Sources

The text data used for the stressing experiments was provided by *UAB "Taikomasis dirbtinis intelektas"* (AAI-Labs), a company focused on developing custom AI-driven services for businesses and government organizations. While AAI-Labs is not exclusively specialized in TTS technologies, they have experience working with stressed Lithuanian text datasets for applications in speech synthesis and related NLP tasks.

The primary dataset consists of various books and book excerpts written in Lithuanian that have been stress-annotated. The sources include books "*Balta drobulė*", "*Kur vasara amžina*", "*Laiškai Lucilijui*", "*Monologas savam kieme*", "*Poliarinis ratas*", "*Teka upė pro šalį*", and "*Žmogžudystė rytų eksprese*". Stress annotation in these texts was achieved through a combination of automatic methods and manual review, given that current automatic stressing methods are not reliable for fully automated sentence-level stressing. As a result, the quality of the stressed data is presumed to be high; however, the exact error rate remains unknown due to the mixed annotation process. This data was delegated for model training.

For evaluation, a separate dataset is sourced from the book "*Kirčiuotų tekstų chrestomatija*". This book contains 50 stress-marked texts by Lithuanian writers, poets, and prose authors, spanning various time periods, topics, and genres [13]. The book is designed according to contemporary didactic principles, supporting both linguistic and cultural education for non-native learners. Given its educational context and careful preparation, the stress annotations in this book are of high reliability and quality, making it a suitable benchmark for model testing.

2.1.2 Data Pre-processing

The raw data for this research was provided in plain text file (`.txt`) format, with each line corresponding to a sentence or text segment. The files contained standard Lithuanian characters as well as the three primary stress marks used in the Lithuanian language: grave (`U+0300`), acute (`U+0301`), and tilde (`U+0303`) [26]. These stress marks were inserted in

the text as separate UTF-8 combining symbols. This approach has been shown to yield higher synthesis quality compared to alternative methods of incorporating stress annotations, while introducing only a minimal increase in vocabulary size, adding just three additional symbols [12]. Before pre-processing, all training book files were concatenated into a single unified dataset to streamline further data manipulation.

The first stage of pre-processing was data normalization. The original data contained a few inconsistencies and artifacts since it likely was extracted from various sources and formats such as PDF and HTML. This normalization process included removing repeated whitespace and tabulation characters, standardizing dashes, ellipses, and quotation marks to a single symbol for each, and converting all text to lowercase, as stress assignment is not case-dependent. Further, all non-essential symbols were then removed from the dataset. For example, the auxiliary combining dot symbol (`U+0307`), which sometimes appears with stressed dotted letters (such as `ė` or `i`), was eliminated. This cleaning step reduces the vocabulary size and allows the model to focus more effectively on the core linguistic patterns relevant to stress assignment.

Next, sequence length normalization was applied. Some lines in the dataset were excessively long or contained multiple sentences. To address this, any sequence exceeding 200 characters was split. Splitting was performed at punctuation marks whenever possible to maintain sentence-level context. If no appropriate punctuation was found, the split was made at a whitespace. This approach ensured that sequences remained within a manageable length for model training while preserving as much contextual information as possible.

A filtering and validation step was performed to ensure dataset quality. Each word was checked to confirm that it contained exactly one stress mark, and that the mark appeared on one of the 16 Lithuanian letters capable of bearing stress: `a, ą, e, ę, ė, i, į, y, l, m, n, o, r, u, ų, ū` [14]. Any text that failed this rule-based validation was excluded from the final dataset, ensuring that only well-formed and linguistically valid examples were used for model training. However, the overall data quality was sufficiently high that no such exclusions were necessary.

2.1.3 Tokenization and Data Analysis

Once the data was cleaned, the next step was to convert the text into numerical sequences through tokenization, requiring the construction of both source and target vocabularies. Following the analogy with machine translation, Lithuanian stress assignment can be modeled as an $N \to M$ mapping problem: the source sequence uses a vocabulary of Lithuanian letters and symbols, while the target sequence is limited to stress marks and special tokens.

The original stressed Lithuanian text serves as the starting point. To generate the input (source) sequence, all stress marks are removed from the text, leaving only plain Lithuanian characters.

For the target sequence, the goal is to simplify the representation so that the model predicts only the presence and type of stress mark at each position, rather than reconstructing

the entire text. This is achieved by replacing every non-stress-mark character in the original (fully stressed) text with a special `UNK` token ("#", functioning as a placeholder). As a result, the target sequence contains stress marks at the correct positions and placeholders elsewhere, directing the model's attention to predicting stress placement alone.

Alignment between the source and target sequences presented another challenge, as Lithuanian stress marks are Unicode combining characters that increase the length of the original text relative to the plain (unstressed) source sequence. To enforce strict position-wise alignment, each stress mark in the target sequence is shifted to replace the preceding letter it modifies, so that both sequences become the same length. This allows for direct, position-to-position learning during training and reduces modeling complexity, enabling the model to focus on learning the correct mapping from input text to stress pattern (see Table 1).

Table 1 *Character-level alignment for stress prediction. Each column shows the original (stressed) character, the corresponding source (unstressed) character, and the target sequence label (1: none, 4: tilde (~), 5: grave (`), 6: acute (´)) for each position.*

Original text	š	i	õ	s		f	ù	n	k	c	i	j	o	s		à	p	i	m	a
Source text	š	i	o	s		f	u	n	k	c	i	j	o	s		a	p	i	m	a
Target label	1	1	4	1	1	1	5	1	1	1	1	1	1	1	1	5	1	1	1	1
Original text		n	ú	o	t	a	i	k	ą		i	̃r		m	i	ẽ	g	ą	.	
Source text		n	u	o	t	a	i	k	ą		i	r		m	i	e	g	ą	.	
Target label	1	1	6	1	1	1	1	1	1	1	1	4	1	1	1	4	1	1	1	

Both source and target vocabularies include four special tokens that are crucial for managing sequence modeling and batch processing (see Table 2).

Table 2 *Special tokens used in source and target vocabularies.*

ID	Token	Name	Description
0	*	PAD	Used for padding sequences within a batch to ensure all sequences are of equal length.
1	#	UNK	Represents any out-of-vocabulary (OOV) symbol.
2	<	SOS	Indicates the start of a sequence.
3	>	EOS	Indicates the end of a sequence.

The constructed source vocabulary comprises 47 unique symbols, covering all lowercase Lithuanian letters, relevant punctuation marks, and the defined special tokens. The most frequent characters are the space character and the most common Lithuanian vowels and consonants, such as `i`, `a`, `s`, `t`, and `e`. In contrast, the least frequent symbols include less commonly used punctuation (such as parentheses and semicolons) and rare consonants like `h`, `c`, and `f` (see Table 3).

Table 3 *Top 10 most and least frequent tokens in the source vocabulary.*

Most Frequent		Least Frequent	
Token	**Count**	**Token**	**Count**
(space)	453,557	* (PAD)	0
i	374,086	# (UNK)	0
a	340,602	)	241
s	220,219	(	252
t	168,878	;	1,175
e	155,162	!	2,008
o	150,893	h	2,419
r	144,680	:	2,443
k	139,937	c	3,184
n	139,879	f	3,958

The target vocabulary contains 7 tokens, including 4 special symbols and 3 stress marks. The table below (Table 4) presents each token, its count in the dataset, and the weight assigned for cost-sensitive learning. Weights are calculated to address class imbalance during training, with the PAD token (*) assigned a weight of zero to ensure it does not contribute to the loss calculation.

Table 4 *Target vocabulary tokens: count and class weight.*

Token	Count	Weight
* (PAD)	0	0.0
# (UNK)	2,931,105	0.0185
< (SOS)	54,144	1.0
> (EOS)	54,144	1.0
~ (tilde)	210,133	0.2577
` (grave)	168,472	0.3214
´ (acute)	108,963	0.4969

The final dataset comprises 56,447 samples[2], which were partitioned into a training set of 54,144 samples (95.92%) and a validation set of 2,303 samples (4.08%). All validation set data is taken exclusively from the book "*Kirčiuotų tekstų chrestomatija*".

As shown in Table 15, Table 16, and Table 17 (present in Appendix 2), the word and character statistics are highly comparable between the training and validation sets, confirming that the two datasets share similar linguistic characteristics. This observation is further supported by the word and character count distributions presented in Figure 7, Figure 8, and Figure 9 (present in Appendix 2), although minor differences are more noticeable in the distribution plots.

Table 5 shows that the distribution of stress marks is also consistent across sets, with tilde

[2]In this context, a sample refers to a single sequence of text with a maximum length of 200 characters. Most samples correspond to full sentences, but some may be partial sentences resulting from splits at punctuation marks or whitespace during pre-processing.

Table 5 *Stress mark statistics for training and validation data.*

(a) *Training Data*

Type	Count	Percentage
Total	487,568	100%
Grave	168,472	34.55%
Acute	108,963	22.35%
Tilde	210,133	43.10%

(b) *Validation Data*

Type	Count	Percentage
Total	22,983	100%
Grave	7,010	30.50%
Acute	5,201	22.63%
Tilde	10,772	46.87%

and grave accents being the most frequent and the acute accent appearing less commonly (half as often as tilde). This is a pattern expected from the linguistic rules of Lithuanian (the acute accent is used under more specific conditions). While some class imbalance exists, it is not severe and can be effectively addressed using cost-sensitive learning, where token frequencies are accounted for through class weights during training.

2.1.4 Model Architecture

This thesis proposes a Transformer-based, sequence-to-sequence model for Lithuanian stress assignment. By leveraging the Transformer's attention mechanism and ability to model global context, this approach directly addresses the key limitations in current prominent stressing methods: it enables context-sensitive, sentence-level stress prediction, and can potentially achieve higher accuracy and generalization than both traditional rule-based and prior documented neural approaches (LSTM, BERT). Successful development of such a model would represent a significant step forward for Lithuanian TTS and other downstream language technologies.

The model architecture used is implemented in the PyTorch library [7], the implementation is based on the Transformer design introduced by Vaswani et al. in the "*Attention is All You Need*" paper [30]. The Transformer consists of an encoder and a decoder, each built from stacked layers that integrate multi-head self-attention and position-wise feedforward networks (see Figure 1). Because the Transformer is position-agnostic by design, positional encoding is needed to inject sequence order information in to the source and target embeddings.

The created model consists of the following components and a total of 22,102,535 trainable parameters:

- **Embedding layer:** Maps source and target tokens to high-dimensional vectors.
- **Positional encoding layer:** Adds position-dependent information to embeddings with dropout.
- **Transformer layer:** Stacked encoder and decoder blocks using multi-head self-attention and feed-forward layers.
- **Generator layer:** Linear layer that projects the output of the decoder to the target vocabulary size.

Model hyperparameter configuration:

- Dropout: 0.1
- Embedding dimension: 512
- Maximum sequence length: 300
- Number of attention heads: 8
- Encoder layers: 3
- Decoder layers: 3
- Feed-forward network dimension: 2048

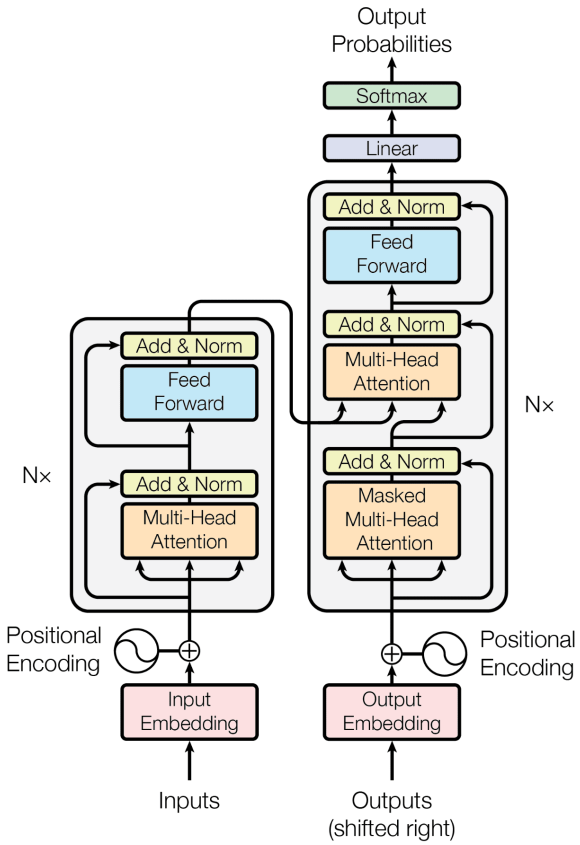


Figure 1 *Transformer model architecture, as proposed and illustrated in Vaswani et al. [30]*

2.1.5 Training and Inference

The training procedure begins with dataset preparation, starting from the point where all texts are already pre-processed and annotated with stress marks. To generate the input (source) sequences, stress marks are removed from the original text and the remaining characters are tokenized using the source vocabulary. The target sequences are created by tokenizing the original stressed text using the target vocabulary and further processed to ensure perfect alignment with the source sequences, so both sequences are of equal length (as shown in Table 1).

Optimization is performed using the Adam optimizer. The training objective is a weighted

cross-entropy loss (2), where class weights are derived from token frequencies in the target vocabulary to address class imbalance. The `PAD` token is excluded from loss computation. The loss is defined as:

$$\mathcal{L}_{\text{WCE}} = -\sum_{i=1}^{N} w_{y_i} \log p(y_i \mid x_i) \tag{2}$$

where N is the number of non-padding tokens, y_i is the ground truth label for the i-th token, $p(y_i \mid x_i)$ is the predicted probability, and w_{y_i} is the class weight, typically computed as the inverse frequency of y_i in the training set.

During each training step, the target sequences are shifted by one position to serve as input and output pairs for teacher forcing[3]. The Transformer model takes the embedded and positionally encoded source and shifted target sequences as input, along with the appropriate padding masks and a square subsequent mask to enforce autoregressive decoding. The model outputs a probability distribution over the target vocabulary at each timestep, and the loss is calculated against the true shifted target sequence (see 1 algorithm).

1 algorithm Transformer training step with teacher forcing

Input: source sequence S, target sequence T
$T_{\text{in}} \leftarrow T_{[:,:-1]}$ // Target input: all but last token
$T_{\text{out}} \leftarrow T_{[:,1:]}$ // Target output: all but first token
$M_s \leftarrow$ padding mask for S
$M_t \leftarrow$ padding mask for T_{in}
$A_t \leftarrow$ square subsequent mask for T_{in}
$O \leftarrow \text{Transformer}(S, T_{\text{in}}, M_s, M_t, A_t)$
Compute loss $\mathcal{L}$ between O and T_{out}
Update model parameters using $\mathcal{L}$

For inference a few methods were established:

- Greedy decoding;
- Beam search decoding;
- Greedy decoding with additional stressing rules.

Autoregressive greedy decoding is a straightforward method of inference in context of sequence-to-sequence prediction. It selects the highest-probability token at each step, without considering alternative paths or future possibilities. This makes greedy decoding efficient, but it does not allow for exploration of multiple hypotheses, as in beam search.

Beam search is a heuristic search technique that explores multiple candidate sequences at each decoding step, maintaining the top-k most probable hypotheses instead of greedily selecting only the best option at each stage. This approach enables the model to consider a wider set of possible outputs, reducing the risk of committing to suboptimal early choices and typically yielding higher-quality sequence predictions in sequence-to-sequence tasks [16].

[3]Teacher forcing is a training strategy in sequence models where the ground truth from the previous timestep is fed as input to the model, instead of using the model's own prediction. This helps the model learn more efficiently by preventing error accumulation during training.

Greedy decoding with rules is a custom implementation of a greedy decoding algorithm that incorporates linguistically motivated inference rules tailored to Lithuanian stress assignment (see 2 algorithm):

1. The algorithm predicts stress marks only for those letters in Lithuanian that can be stressed: `a, ą, e, ę, ė, i, į, y, l, m, n, o, r, u, ų, ū`. This restriction reduces computational overhead and prevents erroneous stress assignment to ineligible characters.

2. To enforce correct prosodic structure, the decoder assigns at most one stress mark per word. Once a stress mark is assigned within a word, the algorithm skips to the next word, reducing the risk of over-prediction.

2 algorithm Greedy decoding with additional stressing rules

Require: Source $S = (s_1, \ldots, s_n)$, model, stressable letters $\mathcal{L}$, tokens `SOS`, `EOS`, `UNK`

$T \leftarrow$ [`SOS`]
`is_word_stressed` $\leftarrow$ `False`
for $i = 1$ **to** n **do**
  if s_i is not a letter **then**
    `is_word_stressed` $\leftarrow$ `False`
    Append `UNK` to T
    continue
  end if
  if $s_i \in \mathcal{L}$ and not `is_word_stressed` **then**
    Predict stress for s_i using model and current T
    Append predicted token to T
    if predicted token is a stress mark **then**
      `is_word_stressed` $\leftarrow$ `True`
    end if
  else
    Append `UNK` to T
  end if
end for
Append `EOS` to T
return T

2.1.6 Evaluation, Baselines and Comparison Methods

To quantitatively evaluate model performance, predictions on the validation dataset are compared against ground truth stress-marked sequences. The stress assignment task is framed as a multi-class classification problem at the token level: for each character in the input sequence, the model predicts whether a stress mark should be applied and, if so, which type. This formulation allows the use of standard classification metrics to assess model accuracy both at the sequence and token level.

- **Sequence Accuracy (SA):** The proportion of samples in which the entire output sequence exactly matches the ground truth. This is a strict metric: a single token error in a sequence counts as a miss.
- **Token-Level Metrics:**
  - *Precision:* The proportion of predicted stress tokens that are correct:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3}$$

  - *Recall:* The proportion of ground truth stress tokens that are correctly predicted:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

  - *F1 Score:* The harmonic mean of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

- **Per-Class Token Metrics:** Precision (3), recall (4), and F1 (5) are also calculated separately for each stress mark type (grave, acute, tilde).

In addition to quantitative evaluation, attention map analysis is performed. Attention maps serve as a visualization of the weight distributions produced by the model's attention layers, providing insight into the model's internal decision-making process during stress assignment. In the context of Transformer models, several types of attention maps can be distinguished:

- **Encoder Self-Attention:** These maps depict the dependencies established among input tokens within the encoder.
- **Decoder Self-Attention:** These maps represent the dependencies among generated output tokens within the decoder.
- **Multi-head (Cross) Attention:** These maps describe how each output token attends to the encoded representations of the input sequence.

For analytical purposes, attention weights can be visualized at the level of individual attention heads or aggregated across heads and layers. Such visualizations can assist in diagnosing the model's behavior, revealing whether the network is focusing on linguistically and contextually appropriate positions when predicting stress marks.

To evaluate the effectiveness of the proposed model, several baseline and comparison methods are employed. The primary baseline consists of applying stress to the evaluation dataset using existing rule-based tools: the "Kirčiuoklis" system developed by VDU and the "Phonology Engine" library (0.2.8 version) by A. Pielikis. In line with the central hypothesis of this thesis, the trained neural model is expected to outperform these rule-based methods in terms of stress assignment accuracy. Furthermore, multiple inference strategies

are investigated to assess their impact on output quality, including greedy decoding (with and without the application of post-processing stress rules) and beam search decoding.

Furthermore, a contemporary benchmark is established using GPT-5.2 (Generative Pre-trained Transformer), the large language model released by OpenAI in December 2025. GPT-5.2 has demonstrated state-of-the-art performance across a wide range of natural language processing benchmarks [17]. In this experiment, the evaluation dataset sequences are submitted to GPT-5.2 with explicit prompting for Lithuanian stress annotation. The objective is to assess whether a cutting-edge, general-purpose language model can perform accurate stress assignment without any task-specific fine-tuning. Given the exceptional performance of large language models in tasks such as summarization, translation, and question answering, it is worthwhile to evaluate their applicability to low-resource linguistic tasks like Lithuanian stress marking. The complete system prompt used for this evaluation is provided in Prompt 1.

Prompt 1: System prompt used for GPT-5.2 stressing evaluation.

```
You will be given a Lithuanian text.
Your role is to stress the text using Lithuanian text stressing rules.
Return only the stressed text and nothing else.
Stressed text is a combination of the original text with stress marks added in the correct places.
Each word MUST have only a single stress mark added.
To add a stress mark, place the appropriate character after the letter that should be stressed.
Stress marks are these 3 characters: {GRAVE_ACCENT}{ACUTE_ACCENT}{TILDE_ACCENT}
They can be placed after one of these letters: {STRESS_LETTERS}
Here's an example:
* Original text: {original_example_text}
* Stressed text: {stressed_example_text}
```

2.1.7 Experimental Environment

The experiments were conducted using the hardware and software configurations detailed in Table 6, and Table 7. The model was trained for 1000 epochs, checkpoints were saved every 100 epochs, and whenever a new highest sequence accuracy was achieved, the best model state was saved as well. Evaluation on the validation dataset was performed every 10 epochs to monitor accuracy progression. All training parameters are summarized in Table 8.

The training dataset consisted of 56,447 samples provided by AAI-Labs, while the validation/testing set included 2,303 samples sourced from the book *"Kirčiuotų tekstų chrestomatija"*. The same dataset was used for both validation during model training and testing of various inference algorithms and stressing tools. All data used in this work is provided in Appendix 1.

Table 6 *Hardware Specifications.*

Component	**Specification**
CPU Model	11th Gen Intel(R) Core(TM) i9-11900KF @3.50GHz
Physical/Logical Cores	8/16
GPU	NVIDIA GeForce RTX 3060 (12 GB)
RAM	32 GB
Disk	953.26 GB (94.08 GB free)

Table 7 *Software Specifications.*

Software	**Version/Details**
Operating System	Windows 10 (10.0.19045)
Python	3.12.8 (Anaconda)
PyTorch	2.6.0+cu124
CUDA	12.4
NumPy	2.2.4

Table 8 *Training Configuration.*

Parameter	**Value**
Random Seed	42
Batch Size	32
Epochs	1000
Evaluation Interval	Every 10 epochs
Model Save Interval	Every 100 epochs
Learning Rate	1×10^{-4}

2.2 Speech Synthesis Experiment

This section outlines the methods developed and employed for evaluating the effect of stress-annotated input on Lithuanian TTS synthesis. The central aim is to investigate whether the inclusion of stress information can lead to perceptibly improved stress realization in synthesized speech compared to conventional plain-text or phoneme-based input. This directly contributes to testing the second part of the research hypothesis: that stress-marked text improves TTS output in terms of correct stress realization in audio. The methodology covers the complete experimental pipeline, beginning with the description and preparation of the data sources, followed by model architecture choices, training procedures, and evaluation methods. Special focus is given to the contextual awareness evaluation protocol developed for this work, which assesses whether synthesized stress aligns with human interpretation in stress-sensitive linguistic contexts. The final subsections document the experimental environment and configuration settings to ensure reproducibility.

2.2.1 Data Sources

The experiments conducted in this work rely on speech and text data obtained from multiple sources, serving distinct roles in the training pipeline.

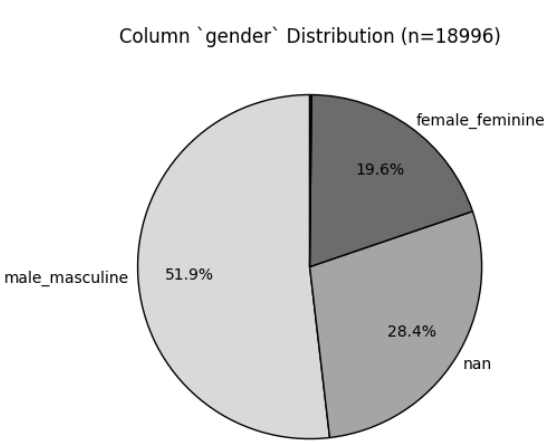
For pre-training the TTS model, the Lithuanian subset of the Mozilla Common Voice corpus (version 22.0) was used. Common Voice is a large-scale, multilingual, open-source speech dataset created by Mozilla through a crowdsourcing initiative, in which contributors donate and validate spoken recordings under a public-domain (CC0) license [4]. Although the dataset is primarily intended for training and evaluating automatic speech recognition (ASR) systems, its size make it suitable for initializing neural TTS models as well.

The Lithuanian Common Voice release (version 22.0), published on June 25, 2025, contains approximately 28 hours of validated speech recordings from 322 distinct speakers. The recordings consist of scripted Lithuanian sentences read by speakers of varying age, gender, and accent. In this work, Common Voice is used to train a base model using plain-text transcriptions, allowing the model to learn general Lithuanian phonetic and prosodic patterns.

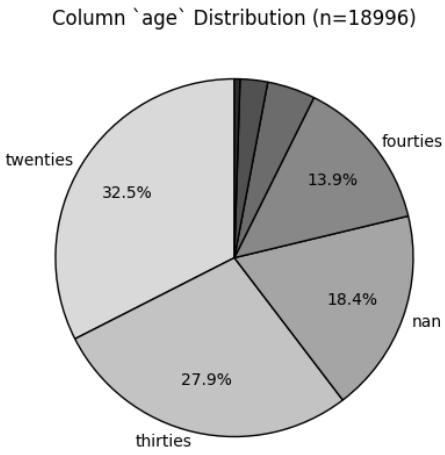
For fine-tuning, high-quality studio speech data was provided by AAI-Labs. This dataset consists of single-speaker Lithuanian speech studio recordings, previously used in speech synthesis research and production tasks. The recordings feature a male speaker narrating the Lithuanian translation of the novel *"Vakarų fronte nieko naujo"* (*All Quiet on the Western Front*) in a controlled studio environment. The total duration of the dataset is approximately 5.8 hours. Importantly, the corresponding transcripts are already annotated with Lithuanian stress marks, making the data suitable for experiments involving stress-aware speech synthesis.

2.2.2 Data Preparation and Analysis

The Lithuanian subset of the Mozilla Common Voice corpus was provided with metadata stored in tab-separated value (`.tsv`) files and audio recordings in MP3 format. An initial analysis of the metadata was performed to assess speaker diversity. The dataset exhibits moderate demographic diversity, with approximately 51.9% of recordings attributed to male speakers, 19.6% to female speakers, and the remainder labeled as unknown (Figure 2a). The age distribution is also relatively balanced, with the largest groups being speakers in their twenties (32.5%) and thirties (27.9%), followed by speakers in their forties (13.9%), and smaller proportions from other age groups (Figure 2b). Despite this diversity, speaker information was not planned to be utilized during training. For the purposes of this experiment, the Common Voice data was treated as a single-speaker dataset. This decision was motivated by the intended role of the Common Voice corpus as a pre-training resource rather than as a source for high-quality voice modeling. Most speakers in the dataset contribute only a small number of recordings, making it unsuitable for robust multi-speaker modeling. Instead, the goal of pre-training was to allow the model to learn general Lithuanian phonetic and prosodic patterns, which would later be refined through fine-tuning on studio-quality single-speaker data.



(a) Gender distribution of speakers in the Common Voice Lithuanian corpus.

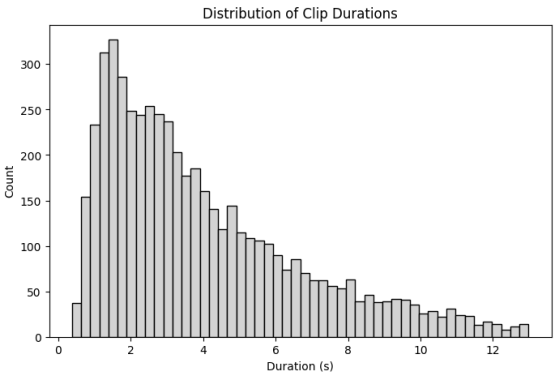


(b) Age distribution of speakers in the Common Voice Lithuanian corpus.

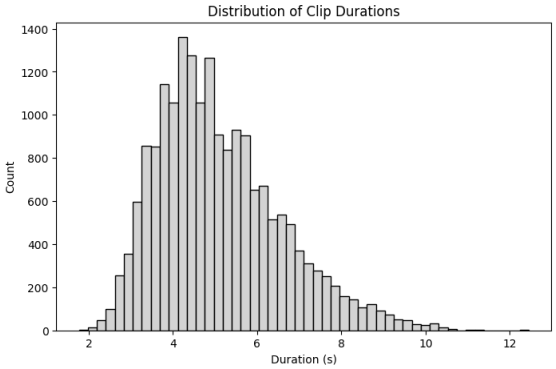
Figure 2 *Demographic distribution of speakers in the Common Voice Lithuanian dataset.*

The Common Voice dataset consisted of 18,996 utterances, totaling approximately 26.9 hours of audio. The minimum clip duration was 1.76 seconds, the mean duration was 5.09 seconds, and the maximum duration was 12.45 seconds (Duration distribution can be seen in Figure 3b). The dataset was randomly split by samples into training and validation sets using a 90–10 ratio, resulting in approximately 24.23 hours of training data and 2.67 hours of validation data.

For model training, the dataset was converted into a filelist format. Training and validation entries were stored in separate `.txt` files, where each line contains the relative path to an audio file and its corresponding transcript, separated by a vertical line (|) delimiter. Audio recordings were resampled from the original 48 kHz sampling rate to 16 kHz and converted to the WAV format to ensure consistency with the training pipeline and to provide lossless audio input.



(a) Clip duration distribution of the studio dataset.



(b) Clip duration distribution of the Common Voice Lithuanian dataset.

Figure 3 *Distribution of audio clip durations in the studio and Common Voice datasets.*

The studio-quality dataset consists of professionally recorded single-speaker Lithuanian

speech. The recordings (large WAV files) are paired with subtitle (`.srt`) files containing stress-annotated transcripts. Each recording was processed by aligning subtitle timestamps with the corresponding WAV files. Individual utterances were extracted by segmenting the audio files according to the timing information in the subtitle entries, producing independent audio clips for each sentence. During this process, audio was resampled from the original 44.1 kHz sampling rate to 16 kHz to match the Common Voice data pre-processing pipeline.

After segmentation, the dataset was analyzed to verify its statistical properties. The final studio dataset contains 5,265 utterances with a total duration of approximately 5.86 hours. The minimum utterance length is 0.38 seconds, the mean length is approximately 4 seconds, and the maximum length is 12.98 seconds (Duration distribution can be seen in Figure 3a). The data was randomly split into training and validation sets by samples using a 90–10 ratio, yielding approximately 5.26 hours of training data and 0.59 hours of validation data. Filelists were constructed using the same format as for the Common Voice dataset to ensure compatibility across experiments.

From the studio dataset, three distinct text representations were constructed to enable a controlled comparison of speech synthesis approaches: stressed text, plain text, and phoneme-based text. The stressed-text dataset directly corresponds to the original manually validated annotations provided with the studio recordings. The plain-text dataset was derived by systematically removing all stress marks from the stressed transcripts while retaining identical audio paths, resulting in a separate but fully aligned filelist. For the phoneme-based dataset, stress marks were first removed from the original transcripts, after which the resulting plain Lithuanian text was converted into phonemic representations using the Bootphon phonemizer Python library. This library provides a unified interface for grapheme-to-phoneme conversion across multiple languages and backends, enabling consistent phonetic transcription without manual rule specification [5]. In this work, phonemization was performed using the eSpeak[4] backend for the Lithuanian language, producing transcriptions in the International Phonetic Alphabet (IPA). The phonemizer configuration preserved punctuation, separated words using whitespace, and did not introduce explicit separators between individual phonemes, ensuring compatibility with the speech synthesis training pipeline. An illustrative example of the three text representations is shown below:

- **Plain text:** Pagaliau prasiskverbia žinia, jog pas mus į apžiūrą atvyksta kaizeris.
- **Stressed text:** Pagaliaũ prasìskverbia žinià, jóg pàs mùs į̃ ãpžiūrą atvỹksta káizeris.
- **Phoneme (IPA):** pagaʟeu praʂiʂkvʲerbʲe ʒɪnʲe, joːk pas mus iː apʒʲuːraː atʲvʲiːksta kaizerʲis.

2.2.3 Model Architecture and Training

The primary objective of this thesis is not to develop or benchmark state-of-the-art speech synthesis architectures, but to evaluate how different text input representations—plain text,

[4]eSpeak is an open-source speech synthesizer and phoneme generator that supports multiple languages. It provides rule-based grapheme-to-phoneme conversion and is widely used for lightweight TTS systems and linguistic pre-processing tasks.

stress-annotated text, and phoneme-based input—affect stress realization quality in Lithuanian TTS. Consequently, the choice of synthesis model is guided by reproducibility and prior empirical evidence rather than novelty of model selection.

Among neural TTS architectures, Tacotron 2 and VITS are the most commonly adopted in recent Lithuanian TTS research. In this work, the VITS model was selected based on prior experimental results reported by Radzevičius, where VITS consistently outperformed Tacotron 2 in subjective MOS evaluations for Lithuanian speech synthesis using both plain text and stress-annotated input [22].

VITS (Variational Inference with adversarial learning for end-to-end Text-to-Speech) is an end-to-end neural speech synthesis model that jointly learns text-to-acoustic alignment, acoustic modeling, and waveform generation within a single framework [15]. It combines a variational autoencoder (VAE) with normalizing flows and a generative adversarial network (GAN), eliminating the need for explicit duration prediction or external vocoder components. The model consists of a Transformer-based text encoder, a posterior encoder that maps acoustic features into a latent space, a flow-based alignment mechanism that enables monotonic text–speech alignment, and a convolutional generator trained adversarially against a multi-scale discriminator. This design allows VITS to produce high-quality speech while maintaining stable training and efficient inference.

The overall training procedure follows a two-stage approach. First, a base VITS model is trained on the Common Voice Lithuanian dataset using plain-text transcriptions to learn general prosodic and phonetic patterns. This base model is subsequently fine-tuned on studio-quality single-speaker recordings to produce three distinct synthesis models: (1) a model trained on plain-text transcriptions, (2) a model trained on stress-annotated text, (3) a model trained on phoneme-based representations. In all three fine-tuning cases the models are trained using the same training data (only different pre-processing routines) and for the same amount of time. This design enables a controlled comparison in which all factors except the textual input representation are held constant.

All four experiments, base model pre-training and three fine-tuning variants, share the same core VITS architecture, optimization strategy, and acoustic feature extraction pipeline. Training was performed using the Adam optimizer with a learning rate of 2×10^{-4}, $\beta_1 = 0.8$, $\beta_2 = 0.99$, and $\epsilon = 10^{-9}$. Mixed-precision training was enabled to improve computational efficiency, and the learning rate followed an exponential decay schedule with a decay factor of 0.999875. The mel-spectrogram and KL-divergence loss components were weighted with $c_{\text{mel}} = 45$ and $c_{\text{kl}} = 1.0$.

All datasets were processed at a sampling rate of 16 kHz using identical text normalization procedures. Acoustic features were extracted using short-time Fourier transform parameters with a filter length of 1024, hop length of 256, and window length of 1024. Mel-spectrograms consisted of 80 mel channels, with frequency bounds mel_fmin = 0.0 and mel_fmax = null. All experiments targeted the Lithuanian language and operated at the character or phoneme level depending on the specific configuration.

The generator and discriminator architectures were kept identical across experiments. The text encoder and posterior encoder used 192 hidden channels with a feedforward dimension of 768. The Transformer-based text encoder consisted of 6 layers with 2 attention heads per layer, kernel size 3, and dropout probability 0.1, while the posterior encoder employed 3 layers.

For all fine-tuning experiments, both the generator and discriminator were initialized from the same pre-trained VITS model trained on Common Voice Lithuanian subset, ensuring a consistent starting point. When fine-tuning on datasets with varying input representations—plain text, stress-annotated text, or phoneme-based input—the effective vocabulary size may change, affecting the shape of embedding layers and other vocabulary-dependent components. As a result, not all parameters from the pre-trained checkpoint are directly compatible. To address this, a checkpoint loading procedure was implemented. During model initialization, parameters with matching shape and data type are loaded directly; incompatible parameters are left randomly initialized. For vocabulary-sensitive layers such as embeddings, an optional partial loading strategy is applied: if the feature dimension matches but the vocabulary dimension differs, the shared portion (rows corresponding to overlapping symbols) is copied from the checkpoint, while rows for new symbols remain randomly initialized. This approach preserves useful learned representations while allowing the model to adapt to new input vocabularies.

The resulting generator models have the following parameter counts. Minor variations are due to differences in the size of the input embedding layer, which depend on the specific input vocabulary used by each model. However, these differences are negligible and do not affect model capacity in any meaningful way, allowing all models to be evaluated and compared on an equal basis.

- **Base model:** 36,296,496 parameters
- **Fine-tuned (plain text):** 36,296,496 parameters
- **Fine-tuned (stressed text):** 36,297,072 parameters
- **Fine-tuned (phoneme-based):** 36,317,424 parameters

2.2.4 Contextual Awareness Evaluation

In this thesis, contextual awareness refers to the model's ability to infer the correct stress placement for a word based on its context within a sentence. This capability is particularly important for Lithuanian, which contains numerous homographs—words that are written identically but differ in meaning and stress depending on their morphological form. A canonical example is the word "*pastato*", which may denote either a noun (*building*) or a verb form (*he/she places something*), with the correct stress pattern determined implicitly by surrounding words. For instance, the sentences "Jonas *pastãto* mašiną" and "Jonas eina iki *pãstato* galo" require different stress assignments for the same orthographic form.

In natural written text, stress information is typically absent and must be inferred from contextual cues. As a result, the responsibility falls on the text processing and speech

synthesis pipeline to correctly interpret the intended meaning and apply appropriate stress during generation. Despite its significance, contextual stress disambiguation is rarely evaluated explicitly. A review of existing research revealed no prior studies addressing this issue within the scope of Lithuanian TTS systems. The only comparable methodology found in the broader literature was a study from 2006, in which binary stress correctness tests were employed to evaluate English TTS output. In that work, listeners judged whether stress was correctly placed, with results reported as the percentage of words perceived to have correct stress [3]. To fill this gap in Lithuanian TTS evaluation, a custom contextual awareness evaluation procedure, functionally similar to binary stress correctness tests, was developed and is presented in this section.

It is important to distinguish the roles of individual components within the TTS pipeline. When stress-annotated text is used as input, contextual stress assignment is performed prior to synthesis by an external stressing model or tool. In this case, the TTS model acts solely as a rendering mechanism that follows explicit pronunciation instructions encoded in stress marks. On the other hand, when plain-text or phoneme-based inputs are used, no explicit stress information is provided, and the synthesizer must implicitly learn stress patterns from the training data. This increases the difficulty of the task and makes contextual awareness an emergent rather than prescribed property of the model.

The proposed evaluation method is based on synthesizing a curated set of sentences containing target homographs whose stress placement depends on sentence-level context. The synthesized outputs are then assessed through a listening test to determine whether the target word was stressed correctly in each instance.
Four TTS pipelines were evaluated:

- Plain-text TTS synthesis without additional preprocessing;
- Phoneme-based TTS synthesis, where text is phonemized prior to input;
- Transformer-stressed TTS synthesis, where text is stressed using the proposed Transformer-based stressing model before synthesis;
- Rule-based tool-stressed TTS synthesis, where text is stressed using either the VDU "Kirčiuoklis" tool or "Phonology Engine" prior to synthesis, whichever one is determined to perform better in terms of stressing accuracy.

A listening test was conducted for all pipelines. While stress correctness can be verified directly at the text level for systems that rely on explicit stressing tools, the listening test was nevertheless applied uniformly to ensure that stress marks were faithfully realized during synthesis. For plain-text and phoneme-based pipelines, perceptual evaluation is unavoidable, as stress realization cannot be determined reliably without listening to the audio output.

The evaluation serves two primary objectives. First, it assesses whether synthesized speech exhibits perceptually improved stress realization when a dedicated stressing component is introduced into the TTS pipeline, compared to plain-text and phoneme-based approaches. Second, it compares the contextual awareness of the two stressing methods, the rule-based approach and the proposed Transformer-based model.

The construction of the evaluation dataset is a critical step in this process. A list of Lithuanian homographs whose meanings change depending on stress placement was compiled with the assistance of students from the Faculty of Philology. Initially, 45 word pairs were collected (e.g., *pastãto–pãstato*, *šaũk–šáuk*, *pláukus–plaũkus*, *laĩdo–láido*). To reduce ambiguity and ensure interpretability, a subset of 27 word pairs with clearer contextual distinctions was selected. For each word pair, four sentences were generated: two sentences (one with short context and one with extended context) for each meaning. Sentence generation was assisted by a large language model (GPT-5), which was prompted with the target word pairs and their intended meanings. All generated sentences were manually reviewed and corrected to eliminate grammatical and semantic inconsistencies. This process yielded a final set of 108 sentences, which are available in Appendix 1 along with all other data. An example sentence set for the word pair *pastãto–pãstato* is provided below:

- "Jis *pastãto* kėdę prie stalo."
- "Kas rytą jis *pastãto* kavos aparatą ant palangės, kad galėtų gerti kavą žiūrėdamas į kiemą."
- "Jonas eina iki *pãstato* galo ir suka į kairę."
- "*Pãstato* stogas buvo šiek tiek įlinkęs, todėl meistrai nusprendė jį sutvirtinti."

Each sentence was synthesized using all four pipelines, producing a set of audio samples. A listening test was then conducted to determine whether the target word in each sentence was stressed correctly. Only the stress realization of the target word was evaluated; overall speech quality and unrelated pronunciation errors were not considered, except in cases where they clearly interfered with stress perception.

The binary correctness labels obtained from the listening test were aggregated into a *contextual awareness score (CA)*. This score is defined as a linear normalization of the proportion of correctly stressed cases, yielding values in the range $[-1, 1]$:

$$\mathrm{CA} = 2 \cdot \frac{\sum \mathrm{correct}}{N} - 1 \tag{6}$$

where $\sum \mathrm{correct}$ denotes the number of sentences in which the target word was stressed correctly, and N is the total number of evaluated sentences. The interpretation of this score is as follows:

- 0 indicates no contextual awareness, corresponding to random or rigid behavior;
- values > 0 indicate positive contextual awareness;
- values < 0 indicate systematic misinterpretation of context.

This interpretation is essential for distinguishing genuinely context-aware behavior from trivial strategies. A model that consistently applies the same stress pattern regardless of context may still achieve approximately 50% correctness in such a binary homograph stressing test, yet this does not reflect meaningful contextual understanding. In this setting, simple

accuracy is not a reliable metric, that's why linear normalization in the range of $[-1, 1]$ is applied, to make the output more interpretable.

According to the hypothesis, TTS systems that incorporate stress information, particularly those using automatically stressed input, are expected to achieve higher CA scores compared to models relying solely on plain text or phonemes. Furthermore, the Transformer-based stressing model developed in this work is anticipated to outperform the rule-based approach in terms of contextual stress accuracy.

2.2.5 Experimental Environment

The experiments were conducted using the hardware and software configurations detailed in Table 6, and Table 9. Logging and evaluation were performed every 500 and 10 000 steps, respectively, using a fixed random seed (1234) to ensure reproducibility.

Pre-training of the VITS model was performed using the Lithuanian subset of the Common Voice dataset (version 22.0), which is publicly available for download. The dataset was divided into 24.23 hours of training data and 2.67 hours of validation data. For the fine-tuning experiments, professionally recorded studio data provided by AAI-Labs was used, comprising 5.26 hours of training audio and 0.59 hours of validation audio. To evaluate contextual awareness, a dedicated set of 108 curated sentences was used for testing. Both the contextual evaluation set and the studio recording data used for fine-tuning are available in Appendix 1.

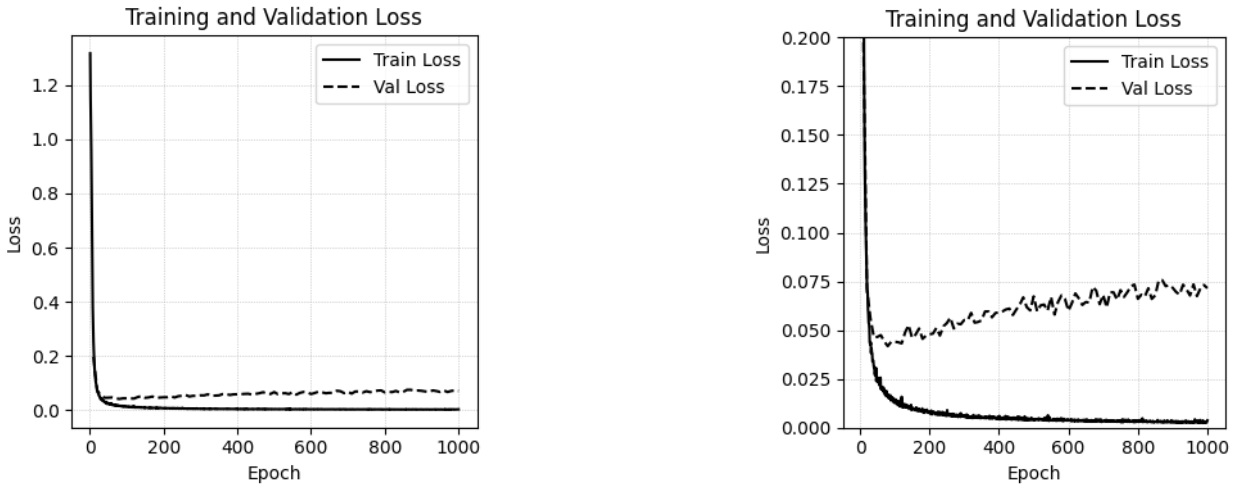
Table 9 *Software Specifications.*

Software	**Version/Details**
Operating System	Windows 10 (10.0.19045)
Python	3.10.18 (Anaconda)
PyTorch	2.6.0+cu124
CUDA	12.4
NumPy	2.2.6

3 Experimental Results

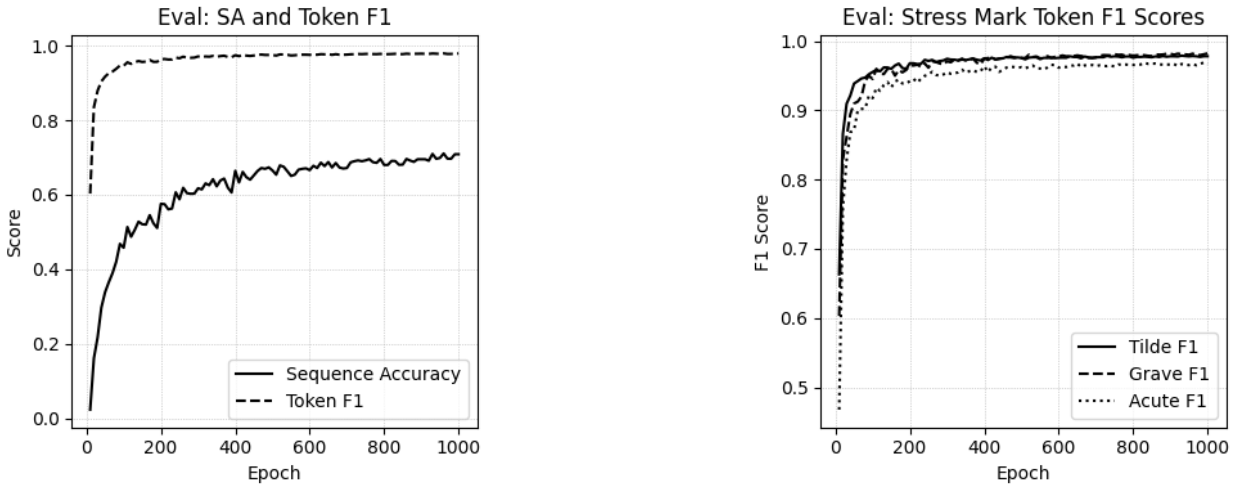
3.1 Text Stressing Results

- Training epochs: 1000
- Training duration: 1.48 days
- Number of parameters: 22,102,535
- Model size on disk: 86,976 KB
- Best weights: achieved at epoch 960 with SA of 0.711



(a) Loss *(b) Loss (removed outliers)*

Figure 4 *Training and validation loss. Full plot (left), zoomed-in (right).*



(a) Sequence Accuracy and F1 *(b) F1 for all stress tokens*

Figure 5 *Validation metrics: SA and token F1 (left), Token F1 score of each stress token (right).*

Figure 4 illustrates the loss curves during training. The validation loss began to gradually increase after approximately 100 epochs, but this upward trend was minimal, with the total increase amounting to just 0.025 by the end of training. Despite this slight rise in loss, all performance metrics continued to improve throughout the entire training period (see Figure 5), with SA reaching its peak at epoch 960. This indicates that the model continued to learn and improve across the full duration of training. Meanwhile, the individual stress metrics plateaued after around 200 epochs, with little change observed thereafter.

Table 10 presents the results of various decoding strategies used to generate stress marks during inference on the validation set. The evaluated methods include standard greedy decoding, beam search with $k = 5$, and a custom greedy decoding variant that incorporates rule-based stress constraints (see 2 algorithm). Standard greedy decoding achieved the highest token-level performance across all metrics. While the rule-based greedy decoding yielded the highest sequence accuracy, its improvement over the standard approach was minimal, only 0.001 higher. Beam search performed similarly to rule-based greedy decoding across all metrics, showing no clear advantage. Overall, the inclusion of stress rules or use of beam search did not provide substantial gains over the default greedy decoding strategy.

Table 10 *Comparison of Transformer inference methods.*

Metric	**Greedy (default)**	**Greedy (rules)**	**Beam Search** ($k = 5$)
Sequence Accuracy (SA)	0.711	**0.712**	0.711
Token Precision	**0.978**	0.963	0.963
Token Recall	**0.980**	0.959	0.959
Token F1	**0.979**	0.961	0.961

Table 11 presents the evaluation results of various stress assignment methods (VDU "Kirčiuoklis", the "Phonology Engine" (PE), GPT-5.2, and the proposed Transformer model with standard greedy decoding) on the validation set. Among all evaluated methods, GPT-5.2 performed the worst, achieving a sequence accuracy of only 0.027. This result highlights the model's inability to reliably assign stress marks in Lithuanian, despite its strong general-purpose language capabilities.

A comparison of the two rule-based systems reveals a major gap in recall. The VDU tool left only 312 out of 22,983 words unstressed, primarily due to out-of-vocabulary issues. These cases included non-standard or rare words such as *"jūratė"*, *"gdansko"*, and *"mamatė"*. Incorrect stress placement occurred in 580 instances, mostly affecting words like *yrà* → *ỹra*, *báltos* → *baltõs*, and *bañgos* → *bangõs*, where the correct stress depends on morphological form or contextual cues that the rule-based system lacks. In contrast, the PE algorithm produced 1,137 incorrectly stressed words, nearly double that of VDU, and left 4,150 words unstressed. This higher count of unstressed words is largely attributed to the tool omitting stress on high-frequency, unambiguous function words such as "ir", "kad", "iš", "bet" and others. While the omission of stress in these cases may not severely impact comprehension, it does lead to reduced consistency and lower recall compared to VDU. Despite similar levels

of precision, VDU outperforms PE in recall due to its broader coverage and more consistent output.

The Transformer-based model demonstrated the best overall performance, surpassing the VDU tool in both sequence accuracy and token-level metrics. Although the improvement in sequence accuracy was modest (from 0.702 to 0.711), it represents a consistent lead over all other approaches, suggesting that the model's ability to leverage contextual information contributes meaningfully to its performance.

Table 11 *Comparison of stressing methods.*

Metric	Transformer	VDU	PE	GPT-5.2
Sequence Accuracy (SA)	**0.711**	0.702	0.195	0.027
Token Precision	**0.978**	0.977	0.963	0.413
Token Recall	**0.980**	0.961	0.771	0.349
Token F1	**0.979**	0.969	0.856	0.378

Table 12 summarizes the performance metrics for each class of stress tokens assigned by the Transformer with greedy decoding. The acute token consistently yields the lowest scores, while the grave token achieves the highest. Nonetheless, the performance differences among the token classes are marginal.

Table 12 *Token-level precision, recall, and F1 scores for grave, acute, and tilde stress marks.*

Token	Precision	Recall	F1 Score
Grave	0.977	0.987	0.982
Acute	0.963	0.970	0.967
Tilde	0.977	0.980	0.979

For attention analysis, a phrase from the validation set, "*antanukas dabar sekė jos kiekvieną judėjimą*" was used. The model correctly produced the stressed output: "*antanùkas dabar̃ sẽkė jõs kiekvíeną judéjimą.*" Both self-attention and multi-head attention were examined for the encoder and decoder, with attention maps averaged and min-max scaled across each layer.

Encoder self-attention (Figure 10, Appendix 2) reveals a notable pattern in the first layer, where the model focuses its attention on the spaces between words. This behavior is logical, as spaces indicate word boundaries, which are crucial for the model to recognize in order to avoid assigning multiple stress marks within a single word. The second and third encoder layers show predominantly diagonal attention, which is typical for sequence-to-sequence models, although the attention map of the third layer is somewhat noisier than the second.

Decoder self-attention (Figure 11, Appendix 2) exhibits a distinct pattern in the first layer, with attention concentrated on a particular character near the beginning of the sequence and on stress marks close to the diagonal. The second decoder layer displays no clear pattern,

suggesting minimal or no activation, while the third returns to a typical diagonal attention structure.

Decoder multi-head attention (Figure 12, Appendix 2) is characterized by a noisy and intricate pattern in the first layer, which appears to form a faint grid: stress mark tokens along the output axis and space characters on the input axis. This further supports the observation that the model leverages space tokens to guide the placement of stress marks, ensuring that each word is stressed appropriately. The second and third layers show broader diagonal attention patterns, consistent with standard sequence-to-sequence behavior. Figure 13 in Appendix 2 provides a detailed visualization of the attention patterns for each head in the multi-head attention layers. It can be observed that many of the heads tend to capture similar information on the diagonal.

3.2 Speech Synthesis Results

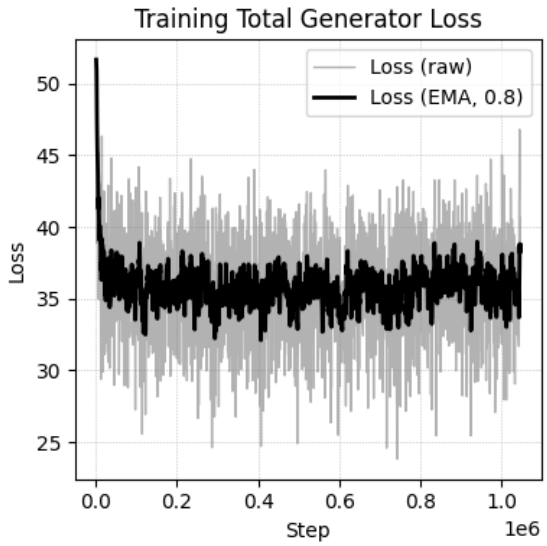
The base model was trained until coherent speech output was achieved on validation data, reaching 1,047,000 steps (514 epochs). The checkpoint selected for all fine-tuning experiments was saved after 1,000,000 steps (491 epochs). All fine-tuned models were trained for 300 epochs, with final checkpoints saved at step 80,000 (epoch 298).

As shown in Figure 6, the base model's training loss (Figure 6a) plateaued around a value of 35 but remained highly volatile throughout training. A similar level of fluctuation is observed across all fine-tuned models, though their training runs were significantly shorter. While the loss values during fine-tuning hovered near 34, no clear stabilization was observed, and the loss continued to vary across epochs with only a small downward trend. Full TensorBoard logs are available in Appendix 1.

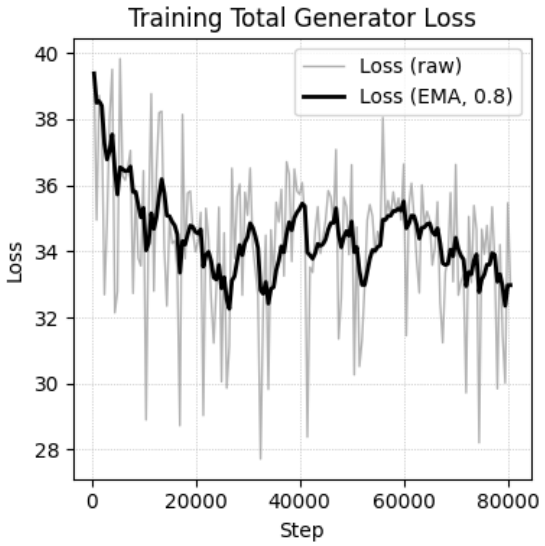
Table 13 *Summary of training durations and model sizes for base and fine-tuned configurations.*

Model	Epochs	Duration	Parameters	Disk Size (KB)
Base (Plain)	491	7.13 days	36,296,496	426,376
Fine-tuned (Plain)	298	15.41 h	36,296,496	426,368
Fine-tuned (Stressed)	298	15.55 h	36,297,072	426,375
Fine-tuned (Phoneme)	298	15.58 h	36,317,424	426,614

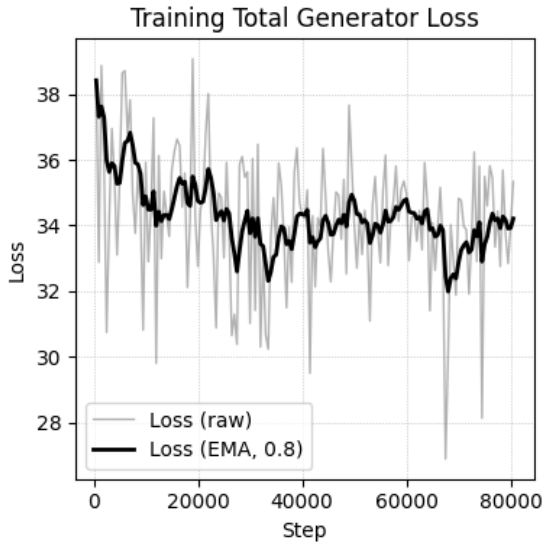
After training, each generator model was used to synthesize sentences from the contextual awareness evaluation set. A listening test was then conducted to assess whether the synthesized speech correctly applied stress to the target word, depending on its sentence context. Only the stress realization of the target word was evaluated, overall speech quality or unrelated pronunciation errors were disregarded unless they directly affected stress perception. The resulting binary judgments (correct/incorrect) were used to calculate the CA score, as defined in equation 6. For the rule-based stressing pipeline, the VDU tool was used instead of the "Phonology Engine"+, as it achieved higher accuracy in prior evaluations and produced fewer unstressed words overall (see Table 11).



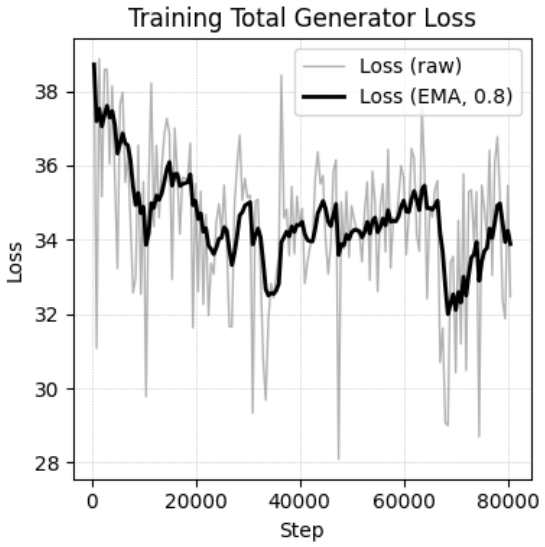
(a) Base model (Common Voice)



(b) Fine-tuned (plain text)



(c) Fine-tuned (stressed text)



(d) Fine-tuned (phoneme-based)

Figure 6 *Training loss curves for the base model and all three fine-tuning variants. Each plot shows raw total generator loss (gray) and its exponentially smoothed version (black, EMA with factor 0.8).*

Table 14 *Contextual awareness scores for each synthesis model.*

Model	Plain	Phonemized	Stressed (TFM)	Stressed (VDU)
Score	0.019	0.056	0.296	0.463

The results in Table 14 show that both the plain text and phoneme-based models exhibit scores close to zero. This suggests that they either apply stress in a rigid, context-invariant manner or behave nearly randomly, both indicating a lack of contextual understanding. Although the phoneme-based model slightly outperforms the plain model (0.056 vs. 0.019), the difference is minor and likely reflects incidental variation rather than systematic improve-

ment. This similarity is not surprising, as both models rely on implicit stress learning during training, any context sensitivity that emerges is incidental rather than explicitly modeled.

In contrast, both stress-annotated variants, where an external model is responsible for assigning stress prior to synthesis, achieve significantly higher CA scores. Notably, the rule-based VDU "Kirčiuoklis" outperforms the neural Transformer-based stress model (0.463 vs. 0.296). This outcome was unexpected, as it suggests that the deterministic morphological disambiguation used in VDU is more effective at resolving stress ambiguity in context than the Transformer model trained for stressing purpose. Nevertheless, both scores fall short of the ideal maximum of 1.0, highlighting the substantial room for improvement in modeling contextual stress disambiguation. On the other hand, a larger and more diverse evaluation set may also show a different outcome.

Despite outperforming the baseline methods (plain text and phonemes), neither stressing approach currently achieves truly human-level contextual sensitivity. A consistent failure case was observed for the ambiguous form "plaukus" (accusative plural of "hair" or "while swimming"), where neither the VDU tool nor the Transformer-based model adjusted stress based on syntactic context. In the following sentences:

- *Ji kerpa plaukus.*
- *Ji švelniai šukavo jos plaukus, kol jie pradėjo blizgėti saulėje kaip šilkinės sruogos.*
- *Jis priartėjo prie kranto plaukus.*
- *Plaukus tolyn nuo tilto, jis pajuto, kaip vanduo tampa gilesnis ir šaltesnis.*

Despite clear syntactic cues indicating whether "plaukus" is a direct object (noun) or verb form, both models consistently realized it with the same stress pattern (*pláukus*, meaning "while swimming"), failing to adapt to context. For human listeners, these distinctions are immediately apparent.

4 Discussion

The results of the conducted experiments present a mix of promising developments and unexpected limitations. According to the primary hypothesis of this thesis, a neural network-based model should be capable of leveraging sentence-level context to accurately assign stress marks in Lithuanian text, thereby outperforming existing rule-based tools such as VDU's "Kirčiuoklis". In part, this hypothesis is supported by the results. In the stress prediction task, the Transformer model achieved a higher sequence-level accuracy than the VDU tool (0.711 vs. 0.702) and demonstrated superior performance on per-token metrics (see Table 11). These results suggest that the Transformer is capable of learning generalizable stressing patterns from data, including unseen words, which gives it a distinct advantage over vocabulary-dependent rule-based systems.

The architectural differences between the two approaches help explain these results. The VDU tool relies heavily on a large dictionary, most likely operating at the word or subword level. As a result, it fails to assign stress marks to words not present in its lexicon. The Transformer, by contrast, uses a character-level vocabulary and learns stressing patterns from training data, enabling it to generalize to novel words, which is an important property for downstream TTS systems where consistent stressing coverage is important. However, this performance gap should be interpreted cautiously. The evaluation dataset—*"Kirčiuotų tekstų chrestomatija"*—consists of literary and poetic texts, many of which contain rare words which are absent in contemporary Lithuanian writing. These characteristics may have disproportionately affected the performance of the VDU tool. With a more contemporary or standard corpus, the observed advantage of the Transformer model may diminish.

To investigate context awareness, the attention weights of the Transformer were analyzed. While attention maps offer qualitative insights, they are inherently subjective and insufficient for rigorous conclusions. For this reason, a dedicated context awareness evaluation was conducted during the speech synthesis experiment. This additional experiment provided more critical and quantitative insight. When comparing contextual stress disambiguation capabilities using the contextual awareness score (Equation 6), the VDU tool significantly outperformed the Transformer model (0.463 vs. 0.296). This result reveals a key limitation: despite having access to full sentence information, the current implementation of the Transformer model does not consistently or effectively exploit contextual cues for stress assignment. Interestingly, the VDU tool, although not context-aware in a neural sense, uses rule-based morphology analysis, which can incorporate limited context through the grammatical structure of nearby words. For example, in the sentence *"Jõnas pastãto mašìną ùž pãstato iř eĩna namõ"*, the VDU tool correctly disambiguates *"pastato"* based on the presence of surrounding words. However, when the word *"Jõnas"* is removed, the tool's default stressing becomes incorrect in that case: *"Pãstato mašìną ùž pãstato iř eĩna namõ"* (first word should be *"Pastãto"*). The Transformer was intended to overcome this limitation by modeling long-range dependencies. While it does exhibit this capability in some cases, the

model's performance remains inconsistent. This shortcoming underscores the need for architectural refinement, additional training data, or better objective functions to more robustly capture contextual semantics in stress assignment.

The second part of the thesis hypothesis proposed that incorporating automatic text stressing into the synthesis pipeline would improve the accuracy of stress realization in generated speech. This assumption is clearly supported by the results of the context awareness evaluation, where both plain-text and phoneme-based models performed near chance level (scores close to 0), indicating a lack of meaningful stress disambiguation. In contrast, models utilizing stressing tools achieved positive scores, demonstrating more contextually appropriate stress realization.

This raises an important question: is it worthwhile to pursue this modular approach of incorporating a separate stress assignment model in the Lithuanian TTS pipeline? At first glance, this may seem like a step backwards from the popular end-to-end paradigm in speech synthesis. In recent years, the field has increasingly moved toward unified architectures where a single model learns to handle all aspects of synthesis—including prosody, phonetics, and stress—implicitly from data. From this perspective, delegating stress prediction to a separate model may seem less elegant or future-proof. However, it is important to consider the low-resource nature of the Lithuanian language. End-to-end models that perform well on high-resource languages like English often require massive amounts of data, extensive training time, and powerful computational infrastructure. In practice, these conditions are difficult to meet for Lithuanian. While promising initiatives such as LIEPA-3 are helping to grow available resources [31], Lithuanian TTS remains constrained by limited open-source datasets.

This is precisely where a modular architecture shows its strength. By training a dedicated stress prediction model on a curated dataset, one can develop a reusable component that can be integrated into various Lithuanian TTS systems. Once trained, this model can consistently provide stress annotations for arbitrary text inputs, offloading this responsibility from the TTS model itself. This allows speech synthesis models to focus entirely on acoustic modeling and prosody realization, tasks that are already complex, without being additionally burdened by the need to learn stress placement implicitly. In principle, this division of labor could enable higher quality synthesis with smaller TTS models and less training data.

That said, incorporating stress into TTS training is not trivial. While a stress assignment model can be used during inference to annotate new input text, using stressed inputs during TTS training introduces a new challenge: alignment between text and audio. The stress annotation must reflect how words are actually pronounced in the speech recordings. However, natural spoken language includes variation, such as dialectal stress shifts, speaker idiosyncrasies, and acceptable alternatives (e.g., "agũrkinis" vs. "agurkìnis") [28]. Mistakes, colloquialisms, or ambiguous forms may also occur. Therefore, creating a high-quality dataset for stress-aware TTS training requires a careful review of transcriptions to ensure consistency between written stress marks and spoken pronunciation, which is a significant

undertaking. In this thesis, the studio dataset was already pre-stressed, which enabled stress-aware synthesis experiments without this additional annotation effort. However, scaling this approach would require more extensive preparation and possibly even semi-automated verification methods to ensure label accuracy.

To advance the current research, a key step would be the development of a significantly larger and more robust dataset for stress prediction. The dataset used in this thesis consisted of approximately 56,000 samples, which, while sufficient for a proof-of-concept, is relatively small for training models that are expected to learn nuanced contextual patterns. A dataset comprising several million annotated examples would be more appropriate, particularly if it includes a greater density of homographs, which require the model to resolve stress ambiguity based on surrounding context. Such improvements in data quality and quantity are likely to yield significant gains in both accuracy and contextual awareness.

In addition to expanding the dataset, the stress prediction model itself can be enhanced. The current Transformer-based model contains approximately 22 million parameters, a modest size by contemporary standards. Increasing the depth of the encoder and decoder stacks, enlarging the feedforward and embedding layers, or incorporating pre-trained Lithuanian language embeddings are all viable strategies for improving model performance.

Another valuable line of inquiry would involve exploring the extent to which TTS models can learn stress implicitly. Specifically, it would be beneficial to determine how much and what kind of data (e.g., single vs. multi-speaker corpora) is required for a TTS model to acquire context-aware stressing capabilities without explicit supervision. This could be compared directly to the data requirements for training a separate stress prediction model to achieve the same goal. Such comparative research could help clarify whether a modular or end-to-end approach is more efficient and effective for low-resource language synthesis.

Conclusion

This thesis presented a novel, modular approach to Lithuanian TTS synthesis that integrates a Transformer–based stress prediction model as a dedicated pre-processing component. Motivated by the challenges of stress assignment in Lithuanian, the research hypothesized that (1) a Transformer-based model can outperform rule-based tools by leveraging sentence-level context for stress placement, and (2) stress-annotated text as input to TTS improves stress realization in generated speech.

The experimental results support these hypotheses. In the standalone text stressing task, the proposed Transformer model achieved higher sequence accuracy and per-token F1 scores than the prominent VDU rule-based stressing tool "Kirčiuoklis" tool (SA of 0.711 and F1 of 0.979 vs SA of 0.702 and F1 of 0.969). Importantly, the neural model was able to assign stress even to uncommon words in Lithuanian vocabulary, demonstrating its capacity to generalize beyond rigid dictionary lookups. These findings confirm that neural architectures can learn and apply linguistic patterns in ways that rule-based systems cannot.

However, the Transformer model's ability to exploit sentence context remains limited. In a contextual awareness evaluation that tested whether the model could assign stress correctly based on surrounding words, the rule-based VDU tool outperformed the Transformer (CA score 0.463 vs. 0.296). Although neither system approached human-level performance, this result highlights that morphological heuristics, when combined with partial contextual cues, remain competitive. These findings suggest that while neural models offer flexibility and broader coverage, their contextual inference capabilities still require improvement.

The second part of the hypothesis was also supported: incorporating stress information significantly improved the quality of stress realization in synthesized speech. Plain-text and phoneme-based TTS models, which relied on implicit stress learning, scored near zero in the contextual awareness test, indicating little to no understanding of stress ambiguity. Meanwhile, both stress-informed pipelines, regardless of whether they used rule-based or neural stress assignment, achieved substantially better results, confirming the value of explicitly modeling stress in Lithuanian TTS systems.

Nonetheless, building robust stress-aware synthesis pipelines requires significant investment in data and infrastructure. A major limitation in this work was the size of the stress-annotated dataset (56k examples). Expanding this corpus to several million curated examples with careful attention to homographs would likely yield substantial improvements. Enhancing the model architecture (e.g., more layers, larger embeddings, pre-trained language models, or different architecture) could also help improve context modeling and overall stressing accuracy.

Finally, future work should explore more deeply whether it is more efficient to teach stress implicitly through massive TTS training or explicitly through modular preprocessing. Understanding the trade-offs between these approaches, particularly in low-resource settings, could be crucial for advancing speech synthesis technology in underrepresented languages.

In summary, this thesis demonstrated that stress-based input improves Lithuanian TTS synthesis in terms of stress realization and that a Transformer-based model can serve as a viable alternative to rule-based stress tools. While modular stress prediction diverges from current end-to-end trends, it remains a compelling approach for low-resource languages like Lithuanian. With targeted investments in dataset creation and model scaling, this architecture could lead to substantial improvements in the linguistic fidelity of synthesized speech. Furthermore, the methodology presented in this thesis may be applicable to other low-resource languages that face similar challenges with stress-sensitive pronunciation in TTS systems.

References

[1] T. Anbinderis. "Automatic stressing of Lithuanian text using decision trees." In: *Information Technology and Control* 39.1 (2010).

[2] T. Anbinderis. "Mathematical modelling of some aspects of stressing a Lithuanian text." In: (2010).

[3] J. Arciuli, J. Thompson. "Improving the assignment of lexical stress in text-to-speech systems." In: *Proceedings of the 11th Australian International Conference on Speech Science & Technology*. 2006, pages 296–300.

[4] R. Ardila, M. Branson, K. Davis, M. Kohler, et al. "Common Voice: A Massively-Multilingual Speech Corpus." eng. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Edited by N. Calzolari, F. Béchet, P. Blache, K. Choukri, et al. Marseille, France: European Language Resources Association, 2020, pages 4218–4222. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.520/.

[5] M. Bernard, H. Titeux. "Phonemizer: Text to Phones Transcription for Multiple Languages in Python." In: *Journal of Open Source Software* 6.68 (2021), page 3958. https://doi.org/10.21105/joss.03958. URL: https://doi.org/10.21105/joss.03958.

[6] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, et al. "Unsupervised cross-lingual representation learning at scale." In: *arXiv preprint arXiv:1911.02116* (2019).

[7] P. Developers. *torch.nn.Transformer — PyTorch Documentation*. 2025. URL: https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html (viewed 2025-06-03).

[8] L. Dong, Z.-Q. Guo, C.-H. Tan, Y.-J. Hu, Y. Jiang, Z.-H. Ling. "Neural grapheme-to-phoneme conversion with pre-trained grapheme models." In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pages 6202–6206.

[9] D. Geneva, G. Shopov, K. Garov, M. Todorova, S. Gerdjikov, S. Mihov. "Accentor: An Explicit Lexical Stress Model for TTS Systems." In: *Proc. Interspeech 2023*. 2023, pages 4848–4852.

[10] P. Kasparaitis. "Automatic Stressing of the Lithuanian Text on the Basis of a Dictionary." In: *Informatica* 11.1 (2000), pages 19–40.

[11] P. Kasparaitis. "Automatic stressing of the Lithuanian nouns and adjectives on the basis of rules." In: *Informatica* 12.2 (2001), pages 315–336.

[12] P. Kasparaitis, D. Antanavičius. "Investigation of input alphabets of end-to-end lithuanian text-to-speech synthesizer." In: *Baltic journal of modern computing.* 11.2 (2023), pages 285–296.

[13] V. Kavaliauskas. *Kirčiuotų tekstų chrestomatija: Lietuvių poezija ir proza*. Lietuvos edukologijos universitetas, 2014. ISBN: 9786094170881. URL: https://books.google.lt/books?id=SezizgEACAAJ.

[14] A. Kazlauskienė, G. Raškinis, A. Vaičiūnas. "Automatinis lietuvių kalbos žodžių skiemenavimas, kirčiavimas, transkribavimas." In: (2010).

[15] J. Kim, J. Kong, J. Son. *Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech*. 2021. URL: https://arxiv.org/abs/2106.06103.

[16] R. Leblond, J.-B. Alayrac, L. Sifre, M. Pislar, J.-B. Lespiau, I. Antonoglou, K. Simonyan, O. Vinyals. "Machine translation decoding beyond beam search." In: *arXiv preprint arXiv:2104.05336* (2021).

[17] OpenAI. *Introducing GPT-5.2*. 2025. URL: https://openai.com/index/introducing-gpt-5-2/ (viewed 2025-12-27).

[18] M. Ott, S. Edunov, D. Grangier, M. Auli. "Scaling neural machine translation." In: *arXiv preprint arXiv:1806.00187* (2018).

[19] A. Pielikis. *lt-norm-stress-dataset-gen: tensor2tensor for stressing text in Lithuanian*. GitHub repository. 2018. URL: https://github.com/aleksas/lt-norm-stress-dataset-gen (viewed 2025-12-27).

[20] A. Pielikis. *Transformer Normalizer and Stressor for Lithuanian*. GitHub repository. 2019. URL: https://github.com/aleksas/transformer-normalizer-and-stressor-lt (viewed 2025-12-27).

[21] A. Pielikis. *phonology_engine: Word stress and text normalization library for Lithuanian*. GitHub repository. 2020. URL: https://github.com/aleksas/phonology_engine (viewed 2025-12-27).

[22] A. Radzevičius. "Lithuanian speech synthesis using neural networks." Master's thesis. Vilniaus universitetas., 2022.

[23] A. Radzevičius, A. Raudys, P. Kasparaitis. "Speech synthesis using stressed sample labels for languages with higher degree of phonemic orthography." In: *International Conference on Information and Software Technologies*. Springer. 2021, pages 378–387.

[24] B. H. Story. *History of Speech Synthesis*. English. Edited by W. F. Katz, P. F. Assmann. Publisher Copyright: © 2019. Taylor and Francis, 2019, pages 9–33. ISBN: 9781138648333. https://doi.org/10.4324/9780429056253-2.

[25] I. Sutskever, O. Vinyals, Q. V. Le. "Sequence to Sequence Learning with Neural Networks." In: *arXiv preprint arXiv:1409.3215* (2014).

[26] V. Tumasonis. "Lietuviškos kirčiuotos raidės: kodavimas ir įvedimas iš klaviatūros." In: *Informacijos mokslai* 42-43 (2007), pages 135–140.

[27] UAB Sistemium. *kirtis.info: Web-based Lithuanian stress marking tool*. Open-source project, source code at https://github.com/Sistemium/krc-angular. 2024. URL: https://kirtis.info (viewed 2025-12-27).

[28] Valstybinė lietuvių kalbos komisija. *Rekomenduojamų kirčiavimo variantų sąrašas*. 2025. URL: https://www.vlkk.lt/aktualiausios-temos/tartis-ir-kirciavimas/rekomenduojamu-kirciavimo-variantu-sarasas (viewed 2025-12-18).

[29] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, et al. "Tensor2Tensor for Neural Machine Translation." In: *CoRR* abs/1803.07416 (2018). URL: http://arxiv.org/abs/1803.07416.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017).

[31] Vilniaus Universitetas, Matematikos ir Informatikos Fakultetas. *Projektas „Didžiojo lietuvių kalbos garsyno sukūrimas (LIEPA-3)"*. 2025. URL: https://mif.vu.lt/lt3/mokslas/projektai/nacionaliniai-projektai?view=article&id=4796 (viewed 2025-12-18).

[32] Vytauto Didžiojo universitetas. *Kirčiuoklis: Automatinis lietuvių kalbos kirčiavimas*. 2021. URL: https://kalbu.vdu.lt/mokymosi-priemones/kirciuoklis/ (viewed 2025-06-03).

Appendix 1. AI usage, Code and Data

Acknowledgment of AI Assistance

Throughout the research and writing process of this thesis, AI tools, primarily OpenAI's ChatGPT (versions 4o, 4.1, 5, 5.1, and 5.2), were employed to support various tasks. These included:

- **Coding assistance:** Generating functions based on natural language descriptions, refactoring existing code for clarity and efficiency, and producing code for plotting figures using consistent styles.
- **Writing support:** Refining draft text, correcting grammar, improving phrasing, and ensuring academic tone and fluency across sections of the thesis.

AI assistance was used strictly as a tool to enhance productivity and quality, with all final decisions, analyses, and formulations made by the author.

Code Repositories

The source code developed and used during this thesis is publicly available on GitHub:

- **Text Stressing:** `https://github.com/robertmackevic/ma-lt-stressing`
- **Speech Synthesis:** `https://github.com/robertmackevic/ma-lt-tts`

Data and Experiment Artifacts

All non-public data, including datasets, trained models, and experiment outputs, are stored in a secure institutional OneDrive folder. Access is restricted to members of Vilnius University with valid organizational credentials:

`https://vult-my.sharepoint.com/:f:/g/personal/robert_mackevic_mif_stud_vu_lt/IgC3bIBUqS5ITbEg-YdD0Ku7AfYtD9ktd3cSxNsjLPu9OmM?e=GbZDi8`

Appendix 2. Tables and Figures

Table 15 *Word statistics (per sequence) for training and validation data.*

(a) *Training Data*

Statistic	Value
Total words	487,568
Unique words	72,346
Mean	9.01
Std. deviation	5.85
Median	8.00
Minimum	1
Maximum	31

(b) *Validation Data*

Statistic	Value
Total words	22,983
Unique words	9,277
Mean	9.98
Std. deviation	6.37
Median	9.00
Minimum	1
Maximum	29

Table 16 *Character statistics (per sequence) for training and validation data.*

(a) *Training Data*

Statistic	Value
Total characters	3,906,241
Mean	72.15
Std. deviation	47.37
Median	60.00
Minimum	5
Maximum	200

(b) *Validation Data*

Statistic	Value
Total characters	179,830
Mean	78.09
Std. deviation	50.07
Median	68.00
Minimum	5
Maximum	200

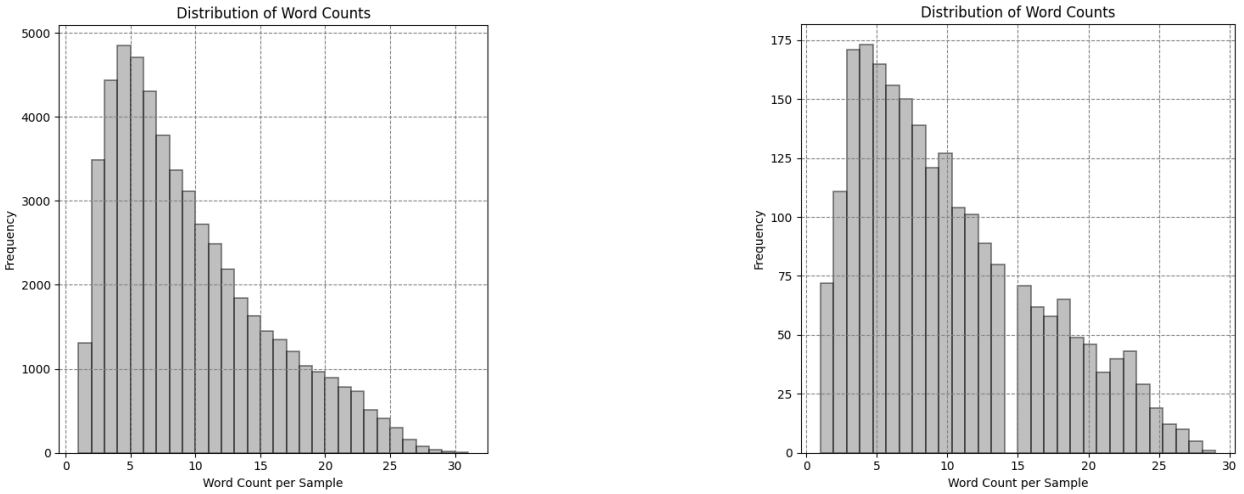
Table 17 *Character statistics (per word, excluding punctuation) for training and validation data.*

(a) *Training Data*

Statistic	Value
Total characters	3,303,254
Mean	6.77
Std. deviation	2.82
Median	7.00
Minimum	2
Maximum	23

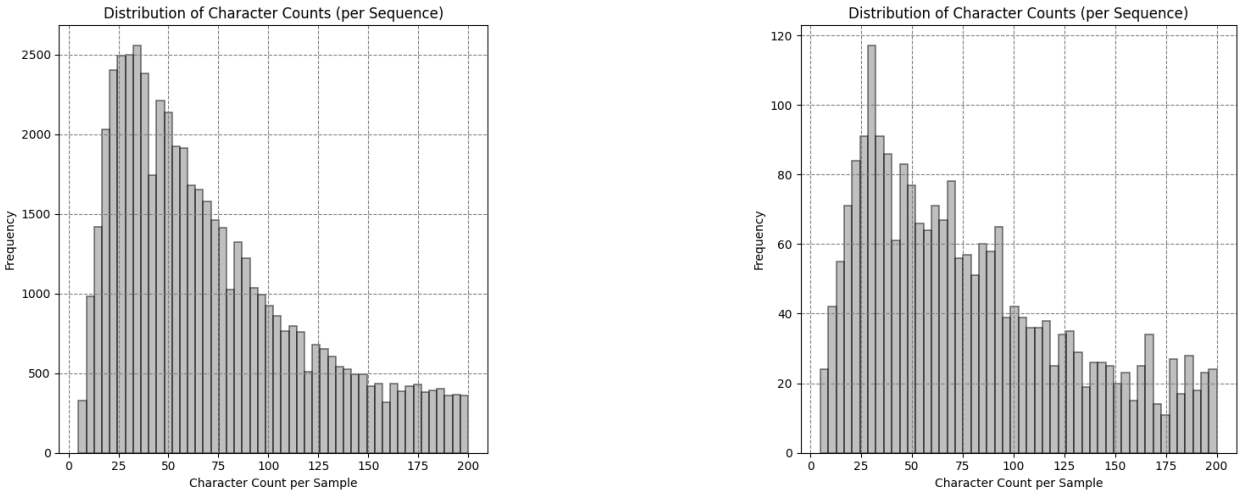
(b) *Validation Data*

Statistic	Value
Total characters	151,526
Mean	6.59
Std. deviation	2.61
Median	6.00
Minimum	2
Maximum	18



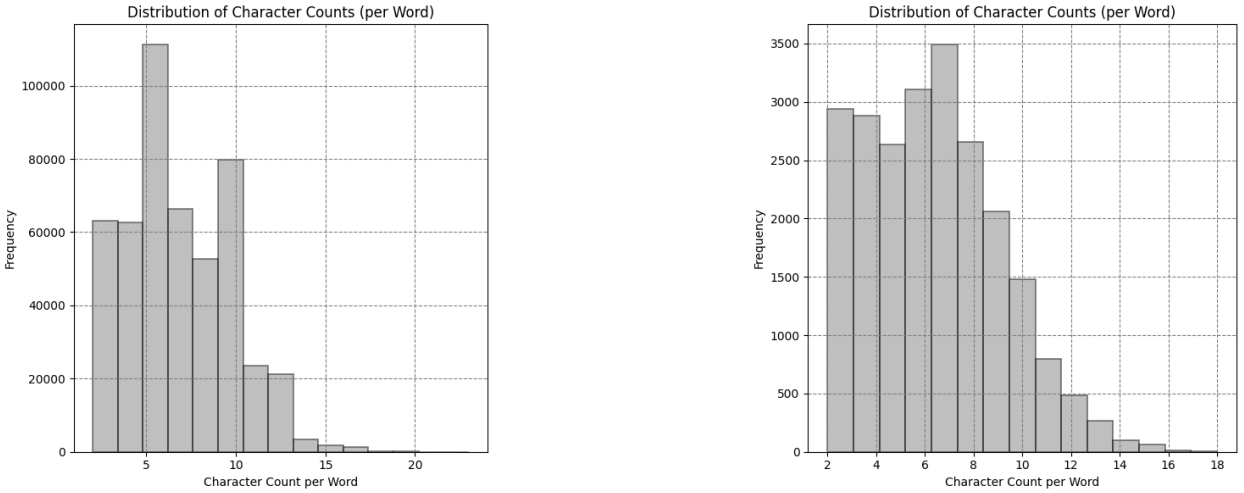
(a) Training Data *(b) Validation Data*

Figure 7 *Word count per sequence distribution: training (left) and validation (right) data.*



(a) Training Data *(b) Validation Data*

Figure 8 *Character count sequence distribution: training (left) and validation (right) data.*



(a) Training Data *(b) Validation Data*

Figure 9 *Character count per word distribution: training (left) and validation (right) data.*

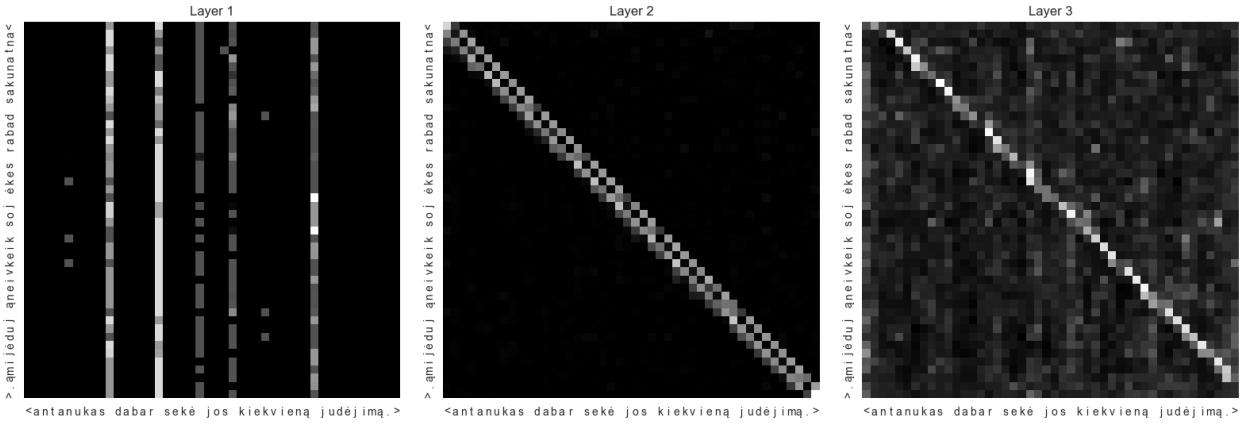


Figure 10 *Encoder self attention for the phrase "antanukas dabar sekė jos kiekvieną judėjimą"*

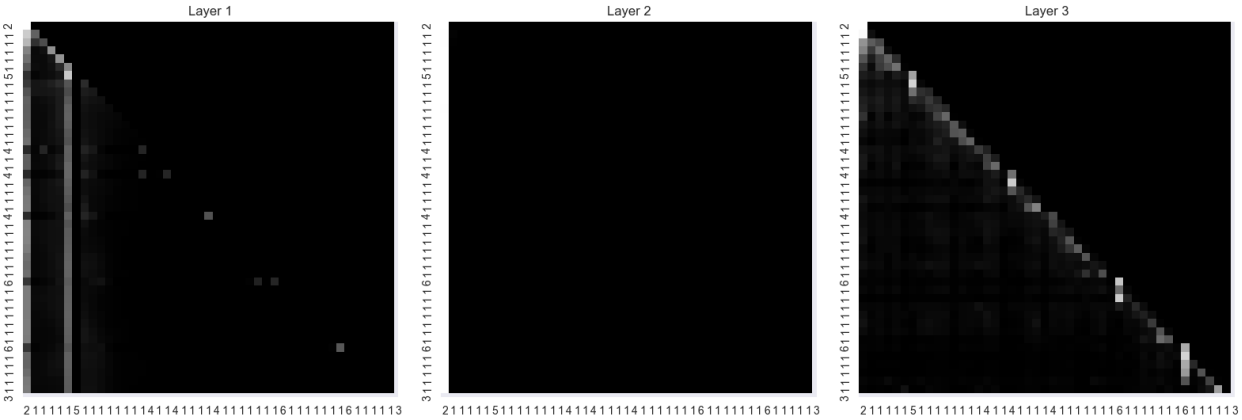


Figure 11 *Decoder self attention for the phrase "antanukas dabar sekė jos kiekvieną judėjimą"*

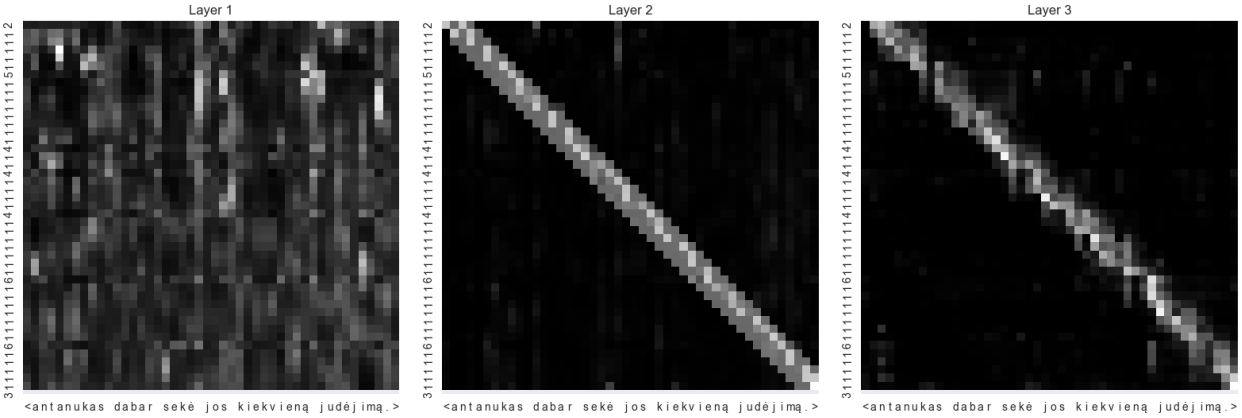


Figure 12 *Decoder multi-head attention for the phrase "antanukas dabar sekė jos kiekvieną judėjimą"*

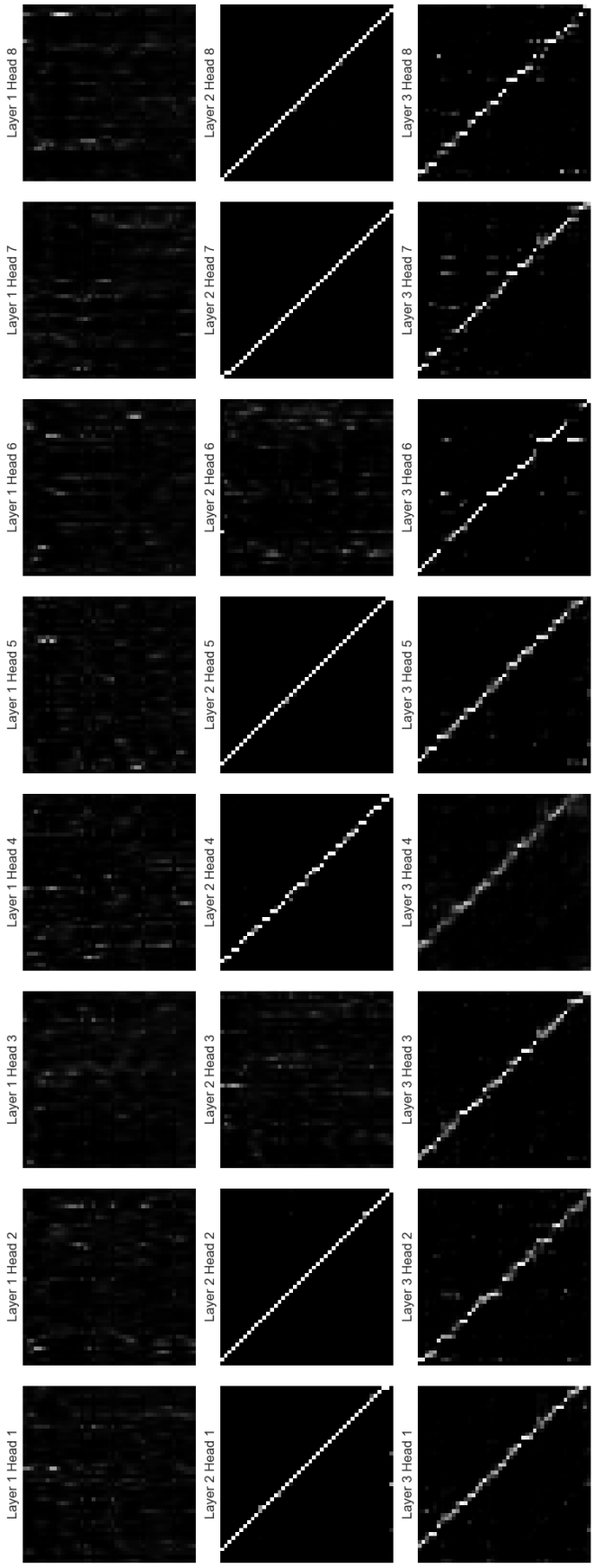


Figure 13 *Decoder multi-head attention of each head in each layer for the phrase "antanukas dabar sekė jos kiekvieną judėjimą" (Rotated 90 degrees)*