VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

DATA SCIENCE STUDY PROGRAMME

Master's thesis

Large Language Models for Lithuanian Language

Didieji kalbos modeliai lietuvių kalbai

Dominykas Plesevičius

Supervisor : Prof. Aistis Raudys

Reviewer : Dr. Tomas Plankis

Vilnius
2025

Acknowledgements

The author is thankful the Information Technology Research Center, Faculty of Mathematics and Informatics, Vilnius University, for providing the High-Performance Computing (HPC) resources for this research.

Summary

Efforts to develop large language models for the Lithuanian language have been limited, primarily due to data and resource constraints. In this work, we aim to address this issue by training models specifically tailored for Lithuanian. We enhance existing multilingual large language models through additional training and develop a new Lithuanian-specific model with an optimized tokenizer. To evaluate their performance, we test these models across a diverse set of benchmarks. The results highlight both the strengths and weaknesses of Lithuanian LLMs while suggesting areas for improvement, including higher quality data collection, synthetic data generation, advanced training techniques, and more effective model design.

Keywords: Large language model, LLM, natural language processing, NLP, multilingual models, low-resource languages, model training

Santrauka

Bandymai kurti didžiuosius kalbos modelius lietuvių kalbai kol kas riboti, daugiausia dėl duomenų ir išteklių trūkumo. Šiame darbe siekiame spręsti šią problemą, kurdami modelius, specialiai pritaikytus lietuvių kalbai. Patobuliname esamus daugiakalbius modelius papildomai juos apmokydami ir sukuriame naują lietuvių kalbai skirtą modelį su optimizuotu tokenizatoriumi. Įvertiname modelių galimybes įvairiuose standartizuotose užduotyse. Rezultatai atskleidžia dabartinių lietuvių kalbos modelių stiprybes bei silpnybes ir sufleruoja galimas tobulinimo kryptis, tokias kaip aukštesnės kokybės duomenų surinkimas, sintetinių duomenų generavimas, pažangesnės apmokymo strategijos ir geresnis modelių dizainas.

Raktiniai žodžiai: Didieji kalbos modeliai, natūralios kalbos apdorojimas, daugiakalbiai modeliai, modelių apmokymas

Contents

List of abbreviations

LLM	Large language model
NLP	Natural Language Processing
GPU	Graphical Processing Unit
DDP	Distributed Data Parallel
LoRA	Low-Rank Adaptation
GPT	Generative Pre-trained Transformer

Introduction

Over the past decade, the field of natural language processing has been reshaped by the emergence of large language models. These models, have shown remarkable capabilities across tasks such as machine translation, question answering, and text summarization. However, development has largely centered on high-resource languages like English, leaving smaller languages like Lithuanian underrepresented. This discrepancy stems from limited access to high-quality Lithuanian text data, as well as insufficient financial and research incentives to build Lithuanian large language models.

Despite these challenges, recent advances in multilingual language modeling indicate that smaller languages can be effectively supported. Certain multilingual models have incorporated Lithuanian in their training. Yet, these models often rely on tokenizers optimized for larger, well-resourced languages, which can diminish performance on Lithuanian tasks.

We aim to address these gaps by improving existing and creating new large language models tailored to Lithuanian. Several studies are carried out:

- We train a new ”GPT-2 Lithuanian” model.
- We choose existing multilingual models best suited for Lithuanian.
- We improve the chosen models through additional training.
- We perform extensive evaluation of said models.

Through model evaluation on multiple benchmarks, this work reveals that the primary limitation for Lithuanian language models lies in the scarcity of large, high-quality datasets. However, the findings also highlight that careful training strategies and the development of Lithuanian specific tokenizers can enhance performance, even for smaller models.

Structurally, this thesis first examines the existing literature on low-resource language LLMs and the development of Lithuanian-specific models, providing context for the work already done in this domain. It then explains the methodology employed for training and improving the models. Following this, the data preparation pipeline is detailed, outlining the steps taken to curate and refine the datasets used in training. The thesis then focuses on the selection of candidate models, identifying existing multilingual models that are best suited for Lithuanian tasks. In addition to the existing models, it introduces the new ”GPT-2 Lithuanian” model, designed specifically to address the challenges of modeling Lithuanian. The training strategy is then described, and training is performed for the selected models. Finally, the evaluation of the models across various benchmarks is presented, along with an analysis of the results and possible improvements.

1 Literature review

Studies have consistently demonstrated that large language models underperform on low-resource languages compared to high-resource ones. Performance metrics across various benchmarks reveal a persistent gap between languages such as English, and those with limited resources. Research indicates that models like ChatGPT, which excel in tasks for well-resourced languages, struggle significantly with low-resource counterparts [1]. For example, South Asian languages have been shown to exhibit lower accuracy in natural language processing tasks compared to their high-resource counterparts [2]. Another study aimed at evaluating and comparing the performance of low-resource languages with English further reinforces this trend [3]. These findings highlight the major challenges low-resource languages face in LLM development today.

The challenges stem from several interconnected factors. One of the most significant issues is the lack of high-quality datasets, which limits the data available for training these models [4]. While languages like English benefit from extensive text corpora across various domains, low-resource languages often lack such breadth and diversity. Another significant issue arises from the tokenization strategies employed by large language models. Tokenizers, which break text into subword units for processing, often introduce biases against languages with complex morphology or infrequent token patterns [5]. This structural unfairness amplifies the challenges already present due to data scarcity, further compromising model performance.

To address these challenges, researchers have proposed and implemented several solutions. One approach is the use of synthetic datasets, which either augment existing corpora or generate entirely new data to enrich low-resource language datasets [6]. Another promising direction involves rethinking tokenizer design. Token-free models, which operate directly on raw text at the byte or character level, have been shown to reduce discrepancies introduced by traditional tokenization processes [7]. Additionally, the development of multilingual models has proven valuable for low-resource languages by leveraging shared linguistic features and enabling cross-lingual transfer learning [8].

In the case of Lithuanian, recent advancements have shown significant promise. The release of the first open Lithuanian-specific LLM, built on the Llama 2 model and additionally trained with the Lithuanian partition of the CulturaX dataset [9], represents a major milestone in Lithuanian LLM development. This effort not only provides the Lithuanian language model, but also introduces translated benchmarks critical for evaluating LLM performance in the language. Multilingual models like EuroLLM [8] and mGPT [10] have also incorporated Lithuanian. Together, these advancements highlight the opportunities for further Lithuanian LLM development. We aim to contribute by investigating additional pretraining techniques on multilingual models and improving tokenization to optimize outcomes for Lithuanian language modeling.

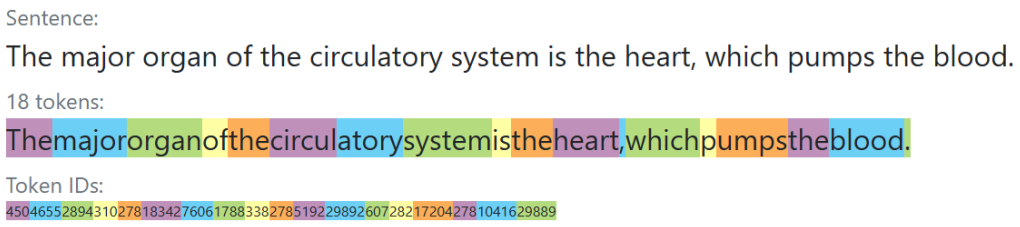
2 Methodology

2.1 Causal language models

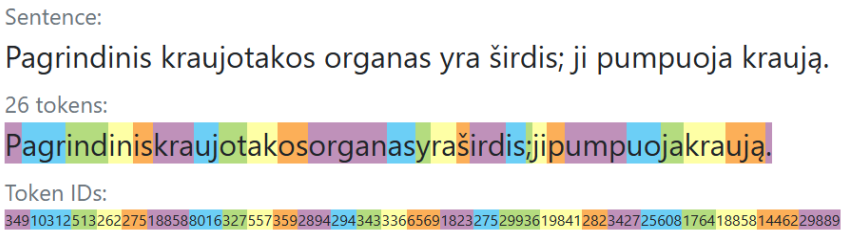
There are various types of large language models, each suited to different applications. We focus on causal language models, which adopt an autoregressive approach to text generation. In practical terms, a causal language model generates each successive token based only on the tokens that precede it. This unidirectional property makes causal models particularly intuitive for tasks where the model predicts the future of a text sequence, such as completing a sentence or conducting a conversation. Notably, many of today's most popular chatbot systems, such as ChatGPT, operate on this causal principle. Our goal is to create models specifically optimized for text generation, and as such, we focus exclusively on this type of model.

2.2 Tokenization

Tokenization is the process of converting raw text into a sequence of numerical tokens that a language model can understand. Essentially, each word or part of a word is assigned a unique integer, allowing the model to process textual input in the form of discrete tokens rather than raw characters. Good tokenization not only helps reduce the total number of tokens needed to represent a sentence, making models faster and more memory efficient, but also preserves linguistic patterns that are crucial for the model to learn and generate coherent text.



(a) English sentence



(b) Lithuanian sentence

Figure 1: *The same sentence in English and Lithuanian tokenized using the Llama tokenizer [11].*

2.3 Context length

For causal language models, context length defines the window of preceding tokens available for predicting the next token in a sequence. A larger context length allows the model to see a more

preceding tokens, enabling it to make better predictions for the next token. While increasing context length generally improves the performance of text generation, it also introduces greater computational and memory demands.

2.4 AdamW optimizer

Training large language models centers on gradient descent: each step attempts to reduce the model's loss by adjusting internal parameters in the direction indicated by the gradient. Pure gradient descent, however, applies the same learning rate to every parameter, relying solely on the immediate gradient to update the weights. This can be slow to converge and is often sensitive to poorly chosen learning rates.

By contrast, AdamW keeps track of more information than just the raw gradient. Instead of using a single learning rate across all weights, it maintains running averages of past gradients for each parameter. These averages allow the optimizer to adjust the learning rate adaptively at each update step, increasing it for some parameters and reducing it for others based on their gradient histories.

Because AdamW relies on accumulated statistics rather than gradients alone, it naturally filters out brief fluctuations. This effect helps guide the optimization process more smoothly toward lower loss regions. Overall, these mechanisms allow AdamW to converge more reliably than pure gradient descent, especially when training large, high-dimensional models.

2.5 Computation optimizations

Training large language models is a computationally intensive process that requires substantial time and resources. To make this process feasible and efficient, it is essential to employ certain computation optimizations that maximize hardware utilization and minimize training time. These optimizations ensure that we can effectively perform our research while working within the constraints of available computational resources.

2.5.1 8-bit optimizer

To increase memory efficiency during model training, we use the AdamW 8-bit optimizer. Unlike the standard AdamW optimizer, which operates using 32-bit floating-point precision for storing parameters and performing calculations, the 8-bit variant reduces the precision of optimizer states to 8-bit floats. This significantly lowers memory usage while maintaining performance comparable to the full-precision optimizer.

For instance, to train a model of 1.7 billion parameters, the standard AdamW optimizer would require memory to store the model parameters, gradients, and two optimizer states in 32-bit precision. This setup would consume about 20GB of memory solely for the optimizer states. By using the AdamW 8-bit optimizer, the memory requirements of the optimizer are reduced, bringing the total to around 10GB. This optimization enables the efficient use of hardware, freeing up memory for larger batch sizes or longer context windows.

However, reducing precision to 8-bit does introduce some potential downsides. Lower numerical precision can lead to a slight loss of accuracy, which might affect convergence speed or final model quality. Despite these concerns, studies have shown that the AdamW 8-bit optimizer performs nearly as well as the standard version, with negligible impact on training outcomes in most cases [12].

2.5.2 Mixed precision training

To accelerate model training, we employ mixed precision training, which utilizes both 16-bit and 32-bit floating-point representations during computations. This approach leverages the speed advantages of lower precision arithmetic while maintaining the necessary accuracy for specific operations.

In this setup, certain parts of the model, such as weights and activations, are stored in 16-bit precision, allowing for faster computations. However, to preserve numerical stability, critical operations like gradient calculations and updates are performed in 32-bit precision. While mixed precision training increases memory usage by requiring both 16-bit and 32-bit floats to be stored on the GPU, the significant speed improvements make it a worthwhile trade-off.

2.5.3 Distributed data parallel

Distributed Data Parallel is a highly effective technique for accelerating the training of large language models by distributing the workload across multiple GPUs. Unlike some other forms of parallelism, where a single model is split across multiple GPUs, DDP keeps a complete copy of the model on each GPU and processes different subsets of the training data in parallel. This approach minimizes overhead and maximizes parallelization efficiency since each GPU operates independently on its assigned data, only synchronizing gradients at the end of each training step.

DDP is particularly advantageous because it avoids the complexities and slower training speeds often associated with model parallelism, where splitting the model across devices can introduce significant communication delays and dependencies between GPUs. By fitting the entire model on a single GPU, DDP allows for straightforward scalability and better hardware utilization.

3 Data

3.1 Training dataset

The Lithuanian partition of the Multilingual Colossal Clean Crawled Corpus (mC4) dataset [13] will be used to fine-tune the models for the Lithuanian language. mC4 is a large-scale multilingual dataset originally developed to train the mT5 model family. mC4 provides filtered and language partitioned text data collected from the Common Crawl repository [14], which is a massive publicly available archive of web content scraped monthly from the public internet.

mC4 is constructed by a filtering and partitioning process to ensure the quality and usability of its data. First, language detection is applied to entries in the Common Crawl archive, identifying text that belongs to specific languages, including Lithuanian. Once the language is determined, several filtering steps are applied to remove content that is irrelevant, noisy, or inappropriate for language modeling. These filters exclude placeholder text, program code, advertisements, explicit or offensive material, and excessively short entries.

The Lithuanian partition of mC4 provides a substantial amount of text data for training. However, some challenges associated with web-crawled data or issues specific to the Lithuanian language remain. They include entries with mixed languages, incomplete filtering of Lithuanian-specific offensive words, and variability in language quality across different web sources.

3.2 Data analysis

The Lithuanian partition of mC4 consists of approximately 11.2 million entries of Lithuanian webpage text, scraped from around 160,000 unique web pages. While the data is already cleaned and filtered during the original mC4 preprocessing, certain issues remain unaddressed. These include the presence of mixed-language entries, where Lithuanian text is combined with other languages, as well as Lithuanian-specific offensive terms that were not fully accounted for in the generic filtering process. Additionally, some low-quality content, such as text from informal forums or advertisement boards, and redundant entries from duplicate or highly similar web pages, persist in the dataset.

Taking a look at the source URL distribution graph (Figure 2) we see that the most frequent entries come from reputable sources such as news outlets, Wikipedia, and other high-quality domains.

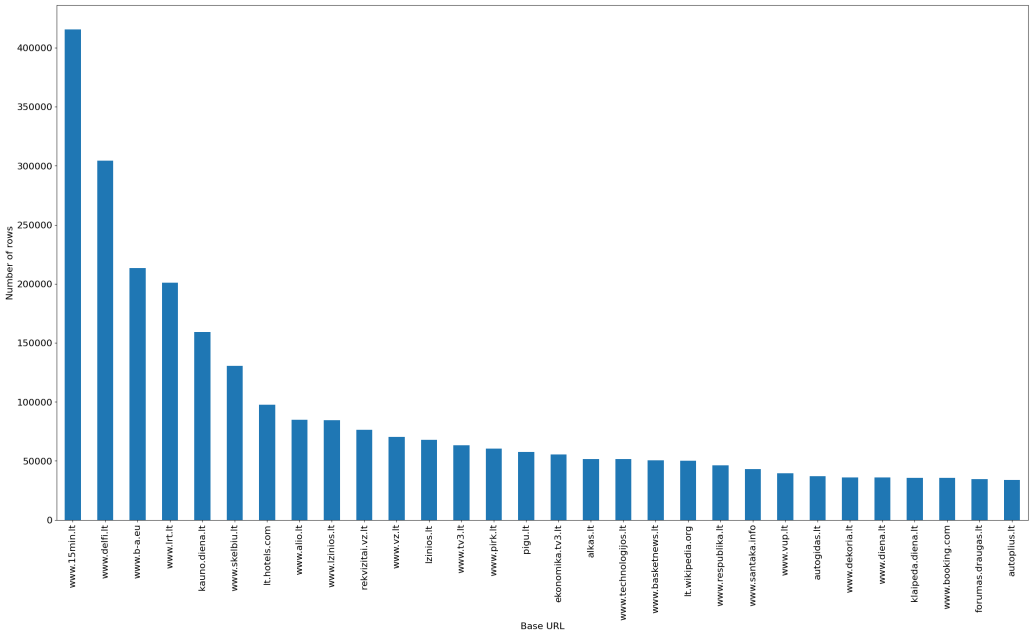


Figure 2: *Top 30 most common URL sources*

These sources typically contribute well-structured, grammatically accurate text, making them ideal for training language models. However, a substantial portion of the dataset originates from forums, advertisement boards, and other user-generated or social media platforms (`skelbiu.lt`, `draugas.lt`, `autoplius.lt`...). These sources often contain informal, poorly structured, or low-quality text, which can introduce noise into the dataset. These entries can affect model performance, as exposure to less curated text can lead to the generation of outputs that mirror the undesirable qualities of the data.

Examining the most common domain suffixes (Figure 3) provides further insight into the dataset's composition.

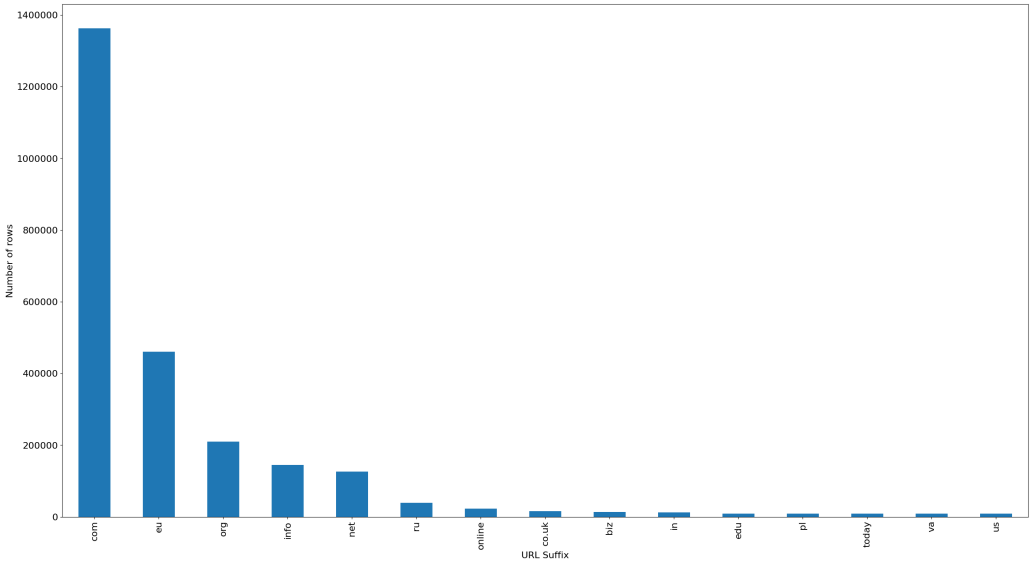


Figure 3: *Top 15 most common URL suffixes without `.lt`*

A significant number of entries come from domains with suffixes such as `.co.uk` (United Kingdom) and `.ru` (Russia), indicating non-Lithuanian sources. Closer inspection of these entries reveal

that many contain a mixture of Lithuanian and another language. This likely arises from the language detection tool `langdetect`[1] which was used during mC4's construction. Although `langdetect` is very efficient, it classifies entries based on the entire input text, which can result in misclassifications when entries include multiple languages. These mixed entries are problematic as they can lead the model to learn incorrect associations or incorporate undesirable patterns that could degrade its performance on Lithuanian tasks.

Additionally, while the mC4 dataset uses preprocessing methods to filter out explicit or offensive language, the filtering process relies on the "List of Dirty Naughty Obscene and Otherwise Bad Words"[2], which does not specifically account for Lithuanian swear words. As a result, entries containing Lithuanian-specific offensive language may remain in the dataset. These entries risk causing the model to generate inappropriate or obscene outputs.

Further investigation of the dataset also reveals duplication issues stemming from the inclusion of both standard and mobile versions of certain websites (`delfi.lt` and `m.delfi.lt`). While the majority of duplicate content was removed during the deduplication process applied to the original mC4 dataset, some overlapping or extremely similar text remains. This duplication has been shown to increase the model's tendency to memorize commonly occurring sentences in the training data [15] which can have negative impact on the models performance.

In summary, while the Lithuanian partition of mC4 offers generally good quality data, several challenges remain, including low quality content, mixed-language entries, and instances of Lithuanian offensive language. We will address each of these issues through targeted processing steps to increase the dataset's quality for large language model training.

3.3 Data processing

To address the problems identified in the section above, we will implement a data cleaning process with steps tailored for each of the data issues.

The process involves re-evaluating the dataset using improved language detection methods, filtering entries with low-quality content or offensive language and additional deduplication. Through this systematic approach, the dataset will be refined to better reflect high-quality Lithuanian language.

3.3.1 Filtering by perplexity

To tackle the challenge of low-quality Lithuanian text in the dataset, we use a perplexity-based filtering method, following the approach described in previous research [16]. This technique involves training an n-gram model on a trusted high-quality corpus and using it to calculate perplexity scores for every entry in the dataset. Perplexity measures how well the model predicts a piece of text, with higher scores indicating that the text is inconsistent with the training data. For this project, a 5-gram model was trained on Lithuanian Wikipedia articles, chosen because they provide reliable, well-edited, and grammatically accurate content.

[1] https://pypi.org/project/langdetect/

[2] https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words

The computed perplexity scores enable us to distinguish between high- and low-quality text. As seen in Figure 4, entries from credible sources like `lrt.lt` and `vz.lt` consistently result in low perplexity scores, confirming their alignment with the high quality text patterns found in the Wikipedia training data. On the other hand, text from forums such as `draugas.lt` or ad sites like `autoplius.lt` score significantly higher. These higher perplexity values reflect content that deviates from the high-quality standard. To improve the dataset, we filter out the top 25% of entries with the highest perplexity scores, ensuring that the remaining text is of better quality.

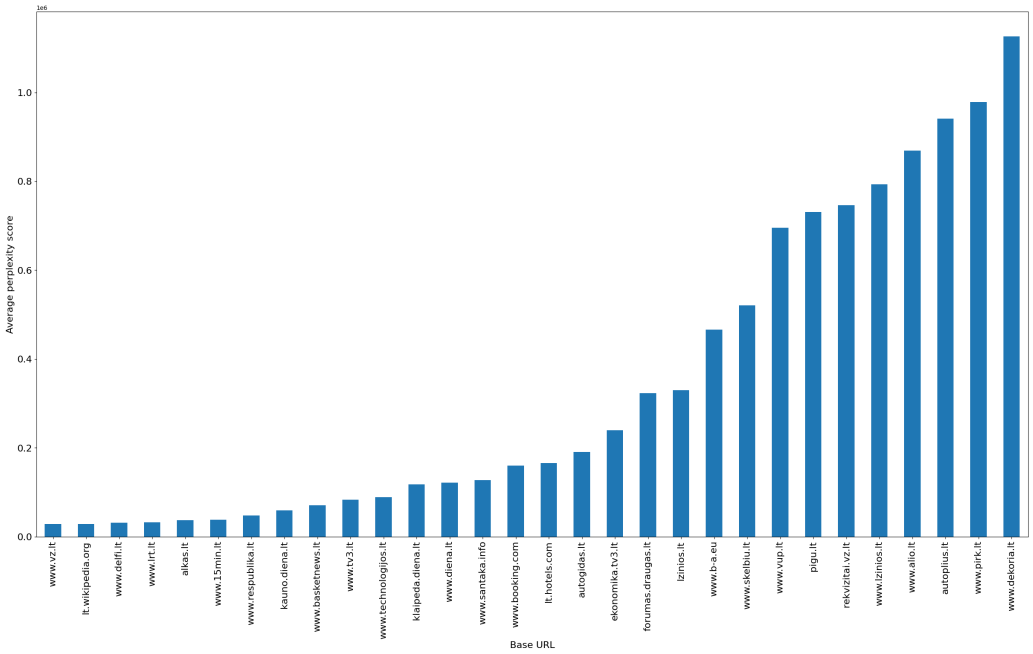


Figure 4: *Average perplexities of the top 30 most common URLs*

3.3.2 Filtering by language

Another issue in the dataset is the presence of mixed-language entries, which occurred because the initial mC4 language detection was performed at the entry level. This means that an entry could be classified as Lithuanian even if it included segments in other languages. To address this, we re-analyzed the dataset using the `py-lingua`[3] library, which applies a combination of n-gram models and can detect multiple languages within a single text entry.

Using this tool, we identified and removed non-Lithuanian segments while retaining the Lithuanian portions of these entries. This process ensured that entries classified as Lithuanian were genuinely representative of the language.

3.3.3 Filtering by bad words

While the mC4 dataset includes preprocessing to remove inappropriate content, it relies on a general list of bad words which does not include Lithuanian-specific explicit language. To fix this, we created a custom list of Lithuanian bad words. Using this list, we filtered out entries containing offensive or inappropriate language. This step ensured that explicit content was mostly removed, making

[3] https://github.com/pemistahl/lingua-py

the dataset more suitable for training. Filtering out such content also helps reduce the likelihood of the model generating offensive and undesirable outputs, which is particularly important for large language models.

3.3.4 Dataset deduplication

Duplication of text in the dataset is another challenge that needed attention. Repeated or highly similar entries can lead to overfitting on frequently occurring phrases or patterns. To address this issue, we applied the tools and methodology described in Lee et al. [15], which has been shown to make the model generate memorized text up to ten times less often.

The deduplication approach iterates over the text of the dataset and identifies substrings that are exact or near-exact duplicates of eachother. It systematically removes these redundant substrings to ensure that only unique content remains. This method is efficient and effective in reducing duplication, helping to improve the quality of the dataset by retaining only non-redundant data.

3.3.5 Filtering short entries

Finally, after filtering for quality, language issues and removing duplicates, we addressed the presence of short entries in the dataset. These entries, often resulting from earlier filtering steps, typically lacked substantial content, such as entries consisting of only a few words or incomplete sentences. To resolve this, we implemented a length-based filter, removing entries with fewer than 100 characters.

This step helped to eliminate noise, as the retained entries offered richer, more contextually complete examples for the model to learn from.

Table 1 shows the summary of how many entries and characters were removed after each training step.

Table 1: *Summary of how many entries and characters were filtered*

	Original	Perplexity filtering	Language filtering	Bad words filtering	Deduplication	Short entries filtering	Filtering rate (%)
Entries (count)	11274295	8455721	8449676	8440990	8437780	8391515	25.6%
Entries (%)	100%	75.0%	74.9%	74.9%	74.8%	74.4%	
Characters (count)	36017009225	30309623778	30163573418	30072147904	27362635039	27359506060	24.0%
Characters (%)	100%	84.2%	83.7%	83.5%	76.0%	76.0%	

4 Models

The development of large language models for Lithuanian has been minimal compared to English and other major languages commonly prioritized in the field. This is primarily due to the challenges we will discuss in the following chapter. To date, there has been only one notable effort to create a dedicated language model for Lithuanian. However, the emergence of multilingual models, including European focused LLMs, opens up new possibilities for supporting smaller languages like Lithuanian alongside their larger counterparts.

4.1 Challenges in developing large language models for Lithuanian

Developing large language models for less commonly spoken or ”unpopular” languages involves significant technical and economic challenges. These challenges arise from the limited availability of high-quality data, lower demand, fewer applications for such models, and the lack of financial incentives or rewards, making investment in these languages less attractive compared to more widely spoken ones.

One of the most significant challenges is the scarcity of large-scale datasets, which is primarily due to the fact that less popular and less widely spoken languages naturally have less written content available on the internet. Unlike widely spoken languages like English, where vast amounts of text data are easily accessible, these languages do not have as much material to work with. This scarcity makes it difficult to train language models effectively, limiting their ability to capture the knowledge needed for good performance in these languages.

Adding to this, resource allocation within the field of artificial intelligence heavily skews toward languages with larger speaker populations or greater economic influence. This prioritization is driven by market dynamics, as the development of LLMs for major languages often promises higher returns on investment due to their broader applicability. Consequently, smaller languages are sidelined, receiving minimal funding and research attention. For instance, languages like Lithuanian, despite their cultural and historical significance, struggle to garner the necessary resources for comprehensive LLM development. This systemic neglect perpetuates the technological gap between well-represented and underrepresented languages.

Another critical barrier is the absence of pre-existing computational tools that are often taken for granted when developing LLMs for major languages. Basic tools like tokenizers may not exist for unpopular languages. This forces researchers to start from scratch, significantly increasing the time, effort, and expertise required to create language models. The lack of standardized benchmarks and evaluation datasets further complicates the process, making it difficult to measure the performance of models or compare them with others.

While multilingual language models, have demonstrated potential for addressing underrepresented languages, they often fall short. These models are typically trained on datasets that include multiple languages, but the representation of smaller languages within these datasets is minimal, resulting in worse performance. The dominance of larger languages in such models creates an imbalance, where the smaller languages are overshadowed.

Developing large language models for less widely spoken languages faces many challenges, including limited funding, small datasets, and the complexity of the languages themselves. These issues make the process harder and require more effort.

4.2 Existing Lithuanian large language models

Currently, only two open Large Language Models specifically trained for Lithuanian exist: `Lt-Llama-2-7B` and `Lt-Llama-2-13B`. These models were introduced in the "Open Llama2 Model for the Lithuanian Language" [9] article and are based on the Llama 2 architecture. They were trained on the Lithuanian subset of the CulturaX [17] dataset.

Evaluation of these models on the translated language understanding benchmarks, show mixed results. While the models demonstrated consistent improvements on tasks like ARC, HellaSwag, and WinoGrande, many other tasks showed limited or no gains over the base Llama 2 models. The authors of the paper hypothesize that this discrepancy may be due to the nature of the pretraining dataset, which primarily consists of web-crawled data and lacks the domain-specific richness required for some of the evaluated tasks.

While these models show promising results on selected benchmarks, their architecture still relies on the original Llama 2 tokenizer, which was designed primarily for English. Although this approach is practical, it may limit the models performance on Lithuanian-specific tasks. Developing or adapting a tokenizer tailored to Lithuanian could further enhance the models performance.

4.3 Existing multilingual models

While multilingual models have been gaining popularity, the priorities in their development often exclude languages with smaller speaker bases, such as Lithuanian.

Among the most advanced multilingual LLMs, such as LLama 3.2, Microsoft Phi-3.5, and BLOOMZ, there is a shared tendency to emphasize high-resource languages. These models benefit from vast amounts of training data for widely spoken languages, however, less popular languages usually remain absent from their training corpora.

At the same time, there are some less popular multilingual models that offer more promise for underrepresented languages. For instance, mGPT [10] incorporates Lithuanian data into its training set, enabling the model to provide more meaningful outputs for Lithuanian tasks. Similarly, EuroLLM [8], designed with a focus on European languages, also includes Lithuanian.

While leading multilingual LLMs excel at supporting high-resource languages, their performance for Lithuanian remains inconsistent due to its absence from many training datasets. However, smaller, region-specific models like mGPT and EuroLLM show promise in representing smaller languages more effectively.

4.4 Criteria for model selection

We select the models for fine-tuning according to four criteria:

1. Model must be a causal language model.
2. Model size less than 2 billion parameters.
3. How well the model tokenizes Lithuanian text.
4. Has the model seen Lithuanian data.

These considerations ensured the choice of models that were computationally feasible to train and capable of effectively handling the generation of Lithuanian text.

The first criterion was that the model must be a causal language model. As described in the methodology, causal language models are well-suited for tasks such as text generation and completion, which is the goal of this research.

The second criterion was model size. Due to technical limitations, models with fewer than 2 billion parameters were prioritized. This decision was based on the need to ensure compatibility with the available computational resources, including memory and processing power, while maintaining a manageable training time. Larger models often provide better performance, but their computational demands make them impractical within the scope of this research. Therefore, we choose to focus on smaller models.

The third criterion was how well the model could tokenize Lithuanian text. Tokenization is a critical process for large language models, and its quality directly impacts the model's ability to understand and generate text [18]. Models that perform poorly in tokenizing Lithuanian text are likely to have worse performance during fine-tuning and downstream tasks. Thus, we prioritize models that demonstrated better tokenization performance.

The fourth criterion focused on whether the model had been trained on Lithuanian data. Models that had seen Lithuanian text during pre-training would not need to "learn" the language entirely from scratch. This would give them an advantage during training, as they would already have some familiarity with the language. For this reason, we prioritized models that included Lithuanian data in their pre-training.

4.5 Selecting the models

To select the specific models for fine-tuning, we compile a list of candidate models that meet the first two criteria. The list is provided in Table 11. To evaluate the tokenization ability of these models, we conducted an experiment using the OPUS-100 dataset [19].

The OPUS-100 dataset is a multilingual parallel corpus containing sentence pairs in 100 languages, including Lithuanian and English. These sentence pairs are aligned translations, meaning that each pair conveys the same meaning. For example, an English sentence like "I would like a cup of coffee" is paired with its Lithuanian translation, "Norėčiau puodelio kavos". This alignment makes the dataset suitable for comparing tokenization performance across languages.

In the experiment, we selected 1,000 Lithuanian-English sentence pairs from the dataset. For each model, we tokenized the sentences and recorded the average number of tokens generated for both Lithuanian and English sentences. We chose to include English in the experiment as a baseline,

since English is a widely supported language in pre-trained models. While we do not necessarily care about the tokenization performance in English, including it helps to illustrate the difference in how the tokenizers handle Lithuanian compared to a more common and well-supported language.

The results of this experiment are presented in Figure 5, which highlights the tokenization performance of each candidate model. Models with better tokenization of Lithuanian were prioritized for fine-tuning.

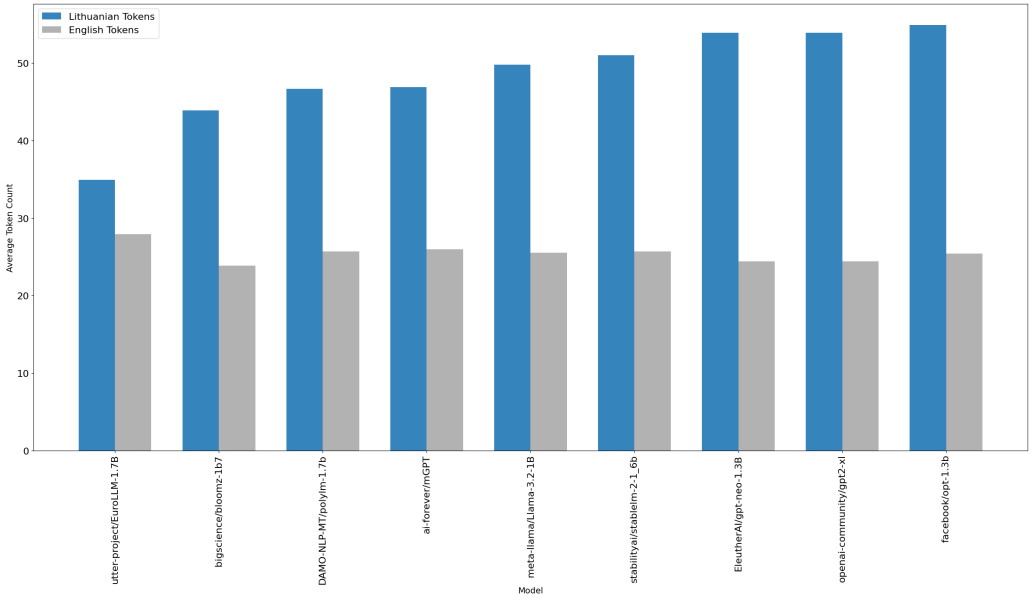


Figure 5: *Average token counts of 1000 Lithuanian and English sentence pairs for each candidate model*

The performance in tokenizing English text across the models is fairly consistent, averaging around 25 tokens per sentence. However, there is more variation in Lithuanian tokenization. The EuroLLM model performs the best for Lithuanian text with an average of approximately 35 tokens per sentence.

Models like BloomZ, PolyLM, and mGPT follow behind, requiring around 45 tokens per sentence for Lithuanian. In contrast, the remaining models tokenize Lithuanian sentences with an average of 50 or more tokens. This is more than double the token count needed for English sentences. This means that to generate equivalent text in Lithuanian the model would have to predict twice the amount of tokens than in English.

Lastly, we investigate whether the training data of the models included Lithuanian text and compile Table 2 to compare the candidate models.

Table 2: *Candidate models with parameter and token counts*

Model	Parameters (Billion)	Avg. Lithuanian Tokens	Inc. Lithuanian in training data
ai-forever/mGPT	1.3	46.895	Yes
EleutherAI/gpt-neo-1.3B	1.3	53.907	Not mentioned
meta-llama/Llama-3.2-1B	1.2	49.799	Not mentioned
bigscience/bloomz-1b7	1.7	43.900	No
DAMO-NLP-MT/polylm-1.7b	1.7	46.659	No
utter-project/EuroLLM-1.7B	1.7	34.922	Yes
openai-community/gpt2-xl	1.6	53.907	No
facebook/opt-1.3b	1.3	54.907	Not mentioned
stabilityai/stablelm-2-1_6b	1.6	51.003	No

Based on this table, two existing models were selected for fine-tuning: mGPT and EuroLLM. Both models met the requirement of having fewer than 2 billion parameters and included Lithuanian in their training data. EuroLLM demonstrated strong tokenization performance for Lithuanian, making it an ideal candidate for fine-tuning. In contrast, mGPT showed weaker tokenization performance but was still included due to its compact size and the presence of Lithuanian data in its training corpus.

4.6 GPT-2 Lithuanian model

As part of this work, we also developed a new model tailored specifically for Lithuanian. The model architecture mirrors the mGPT model, which itself is based on the GPT-2 architecture with certain layer sizes increased. This choice was deliberate, to ensure stability and reduce the likelihood of encountering unexpected issues. By mirroring the mGPT architecture, we use an established design that was practical given our time and computational constraints.

The new model features a Lithuanian-specific tokenizer trained on the Lithuanian split of the mC4 dataset. The tokenizer's parameters, such as vocabulary size and padding tokens, were also replicated from the mGPT tokenizer. This decision allowed seamless integration with the model architecture and avoided the risks associated with developing and fine-tuning an entirely new configuration.

This approach was motivated by the need to address the shortcomings of existing multilingual tokenizers, which struggle with Lithuanian's unique characteristics. By using this tokenizer, the model should be better equipped to handle tasks in Lithuanian, providing a stronger foundation for training and downstream applications.

Running the same tokenization experiment as for the candidate models we see that the resulting tokenizer is able to tokenize Lithuanian text with an average of 21 tokens, which is about 1.6 better than the best performing EuroLLM model. A comparison of the performance of this Lithuanian-

specific tokenizer against those used in existing the Lt-Llama models and the other candidate models are detailed in Figure 15 of the appendix.

5 Training

5.1 Hardware considerations

The training process is conducted on the VU supercomputer, equipped with an NVIDIA DGX-1 system. This system includes 8 Tesla V100-SXM2 GPUs, each with 32GB of RAM. To maximize training speed, we utilize all 8 GPUs during training. We aim to fit the entire model and its training parameters onto a single GPU, allowing for the most efficient parallel training using the Distributed Data Parallel technique.

5.2 Training strategy

The training strategy for the models was inspired by the Llama 3 training recipe, where they progressively increased the context length from 8,000 tokens to 128,000 tokens while simultaneously reducing the amount of data used at each stage [20]. We mimic this approach with shorter context lengths as shown in Table 3.

Table 3: *Context length and percentage of dataset for each training step*

Training step	Context Length	% of dataset
1	512	94%
2	1024	5%
3	2048	1%

Initially, the model is trained with a context length of 512 tokens, on 94% of the dataset. In the second phase, the context length was increased to 1024 tokens, and the model was trained using an additional 5% of the dataset. The final phase of training pushed the context length to 2048 tokens, utilizing the remaining 1% of the dataset and matching the maximum context length of the mGPT and GPT-2 Lithuanian models. While EuroLLM supports context lengths of up to 4096 tokens, training was limited to 2048 tokens due to computational constraints.

This phased approach to context length ensures that the model gradually adapts to longer text sequences without using overwhelming computational resources. By allocating the majority of the dataset to shorter context lengths and progressively increasing them for smaller portions, we aim to balance between resource efficiency and model performance.

5.3 Training parameters

For training, we select the largest batch size that can fit on the available GPU memory, which is 2. Due to memory limitations, directly using larger batch sizes is impossible. To overcome this, we use gradient accumulation with a factor of 8, simulating a batch size of 16. This approach allows the model to benefit from the stability properties of larger batch sizes while staying within the hardware constraints.

We set the learning rate to 0.0002, a value derived from the original pretraining parameters of the mGPT and EuroLLM models. By reusing this established value, we aim to avoid the additional computational cost and complexity of performing a search for an alternative learning rate. As seen from the training loss graphs, this value seems to work well, keeping the training stable.

Table 4: *Summary of used training parameters*

Parameter	Value
Learning rate	0.0002
Batch size	2
Gradient accumulation	8
Optmizer	AdamW 8-bit
Mixed precision	Yes

5.4 Training mGPT

We begin by training the mGPT model with a context window of 512. The loss curve of the first training step is shown in Figure 6.

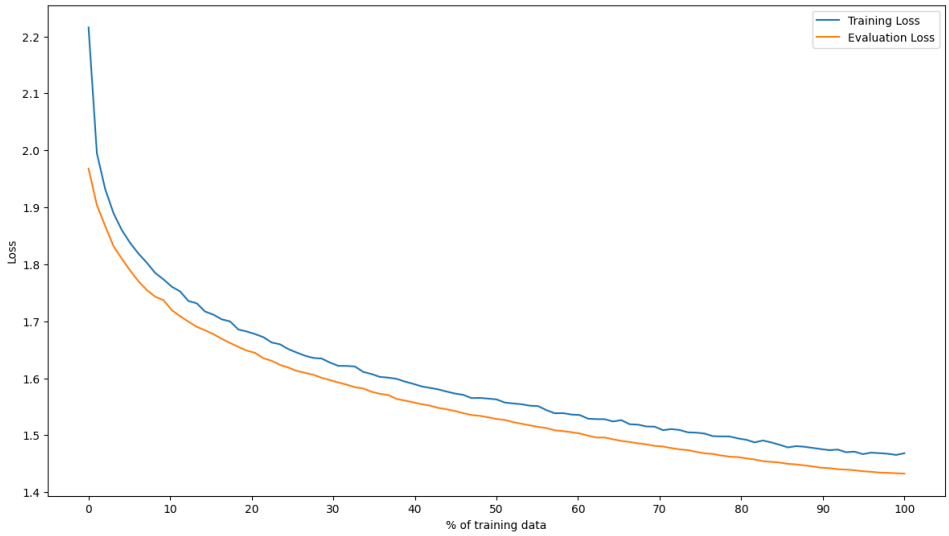


Figure 6: *mGPT model step 1 training and validation losses*

The loss curve shows a fairly standard pattern for a well-behaved training process. There is a sharper decrease in the curve at the beggining of training as the model is quickly learning the patterns in the data. As training progresses both the training and evaluation loss curves begin to flatten indicating that the model is nearing convergence and is learning slower. There is a small and failry constant gap between the training and validation loss, with the validation loss being smaller. This is likely because of data augmentations in the training set, particularly deduplication and the slicing out of non lithuanian langauge.

We then increase the context window and perform the second training step.

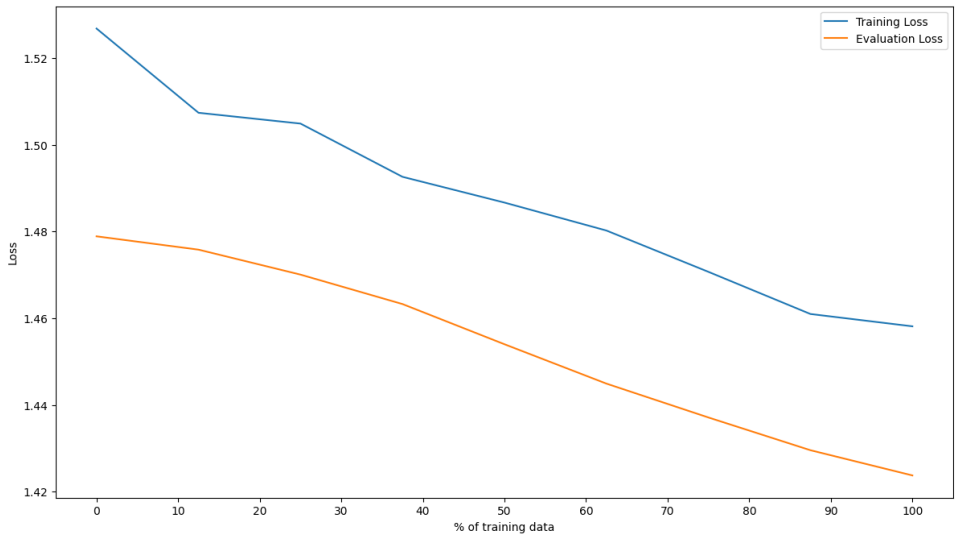


Figure 7: *mGPT model step 2 training and validation losses*

We see that both the training and evaluation loss curves start at similar values compared to the end of first step, which indicates that the model has retained knowledge from the first training step and is building upon that. While the training speed is much slower, we still see a steady decline in the losses.

Lastly, we perform the third and final training step with a context window of 2048.

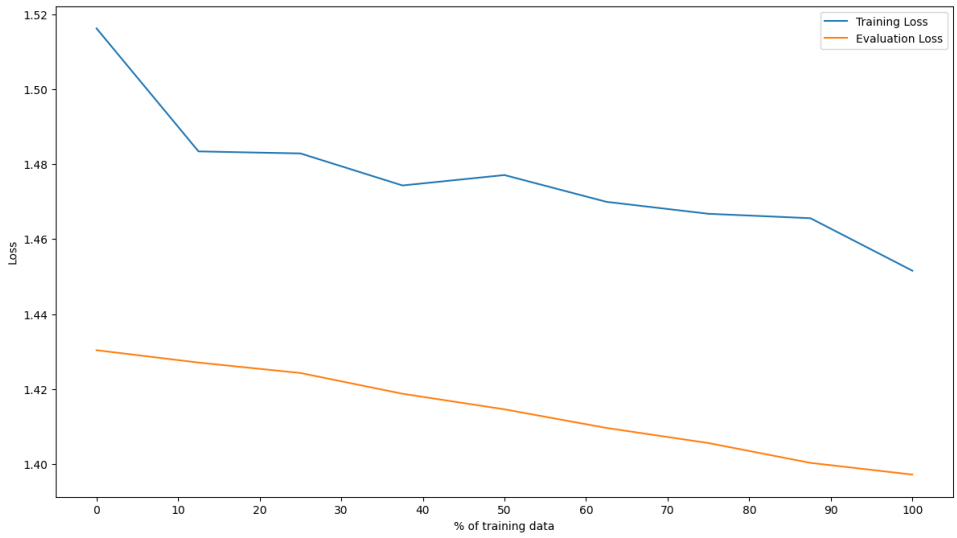


Figure 8: *mGPT model step 3 training and validation losses*

In Figure 8, we observe that the loss curves follow a similar pattern as in the previous step. Both the training and evaluation losses continue to decrease even toward the very end of the training

process, which indicates that the model is still learning and suggests the potential to extend training further.

5.5 Training EuroLLM

We perform the first training step for the EuroLLM model.

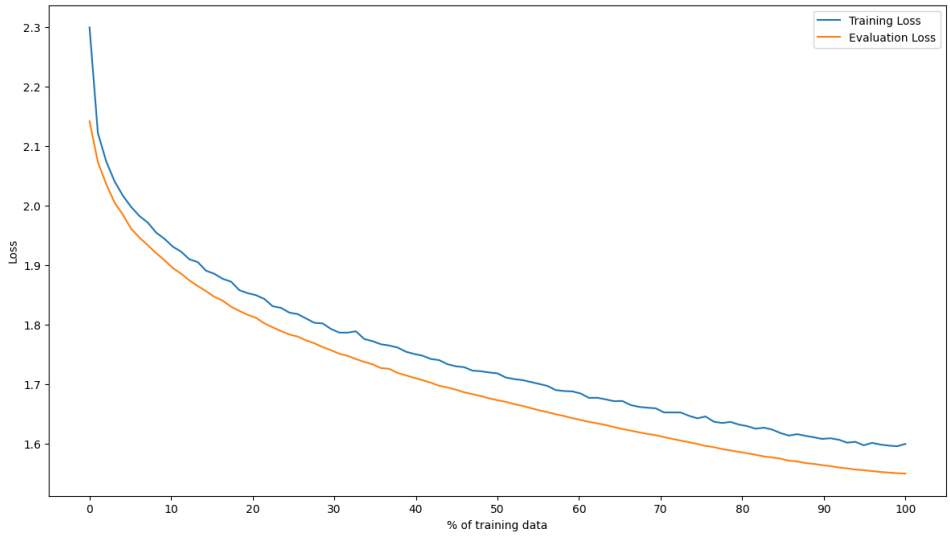


Figure 9: *EuroLLM model step 1 training and validation losses*

The curve looks similar when compared to the first training step of the mGPT model. It also shows a fairly standard pattern, with a steeper decline at the start and a slow flattening at the end.

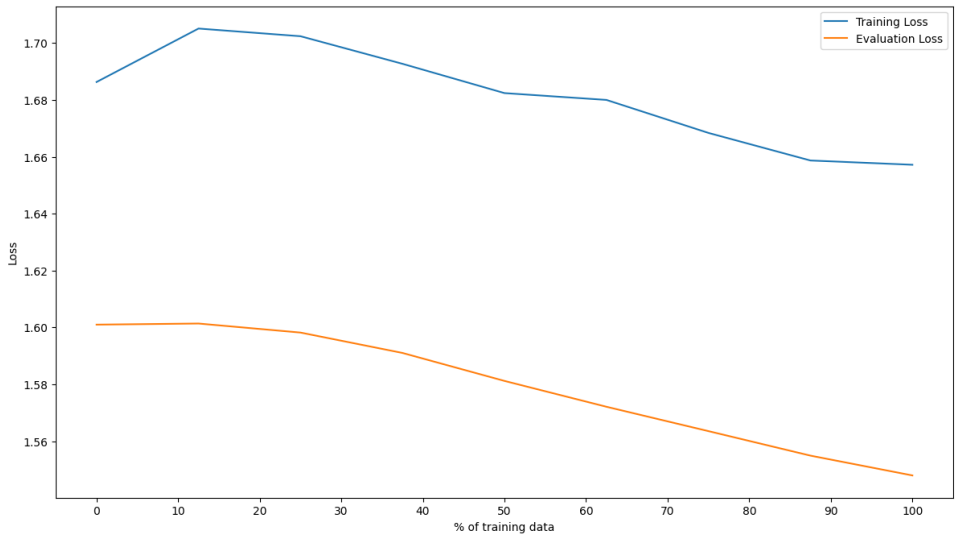


Figure 10: *EuroLLM model step 2 training and validation losses*

In Figure 10, we observe that the training loss at the start of the second step is slightly higher than where it ended after the first step. This could be due to the model adapting to the increased context length. However, the graph still shows a gradual decrease in both the training and evaluation losses.

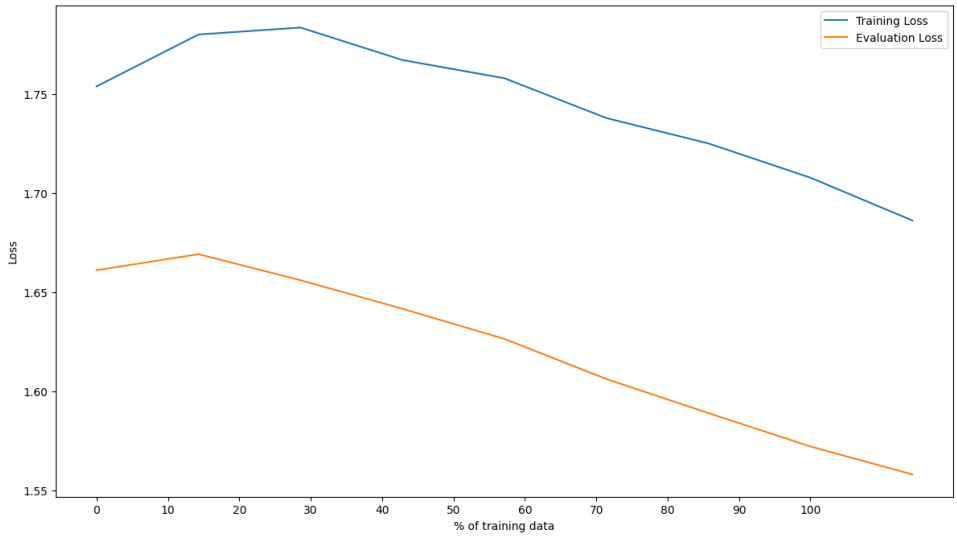


Figure 11: *EuroLLM model step 3 training and validation losses*

For the third training step of the EuroLLM model, the loss curves exhibit a pattern similar to the previous step as well as the mGPT loss curves. Both the training and evaluation loss curves continue the downward trend with a steady decline.

5.6 Training GPT-2 Lithuanian

Lastly, we perform the first training step for the GPT-2 Lithuanian model.

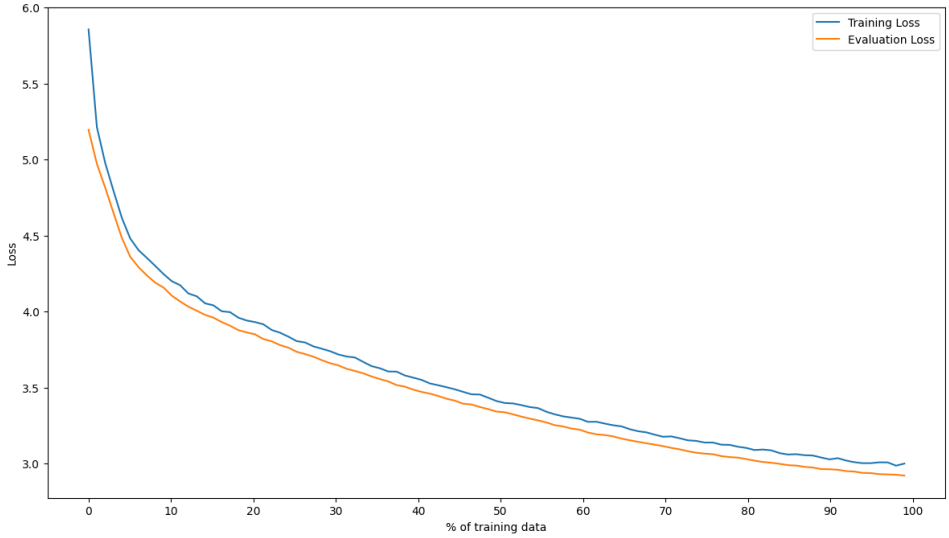


Figure 12: *GPT-2 Lithuanian model step 1 training and validation losses*

From the graph we also see the intial rapid and steep decline, and the flattening towards the end.

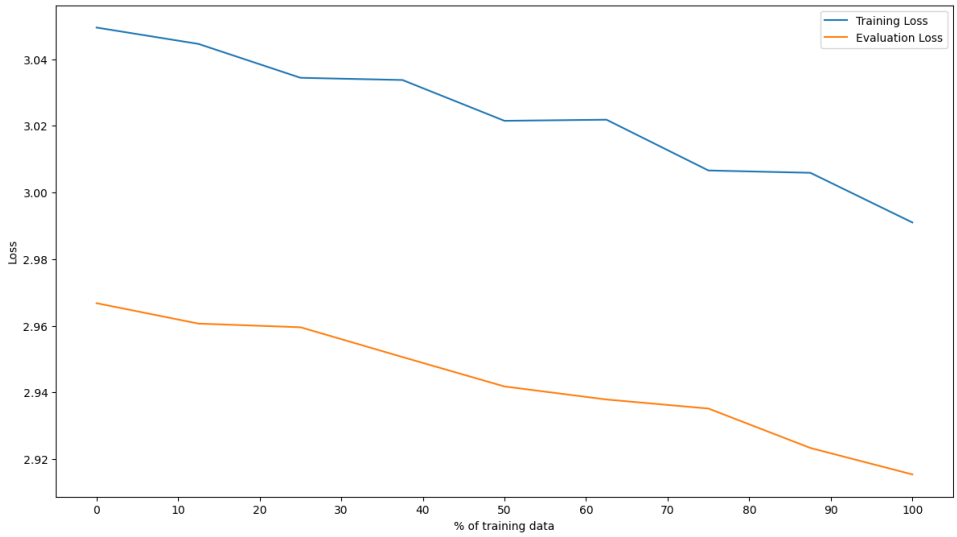


Figure 13: *GPT-2 Lithuanian model step 2 training and validation losses*

Similarly to the other models the second training step demonstrates steady progress, with both training and evaluation losses decreasing smoothly and consistently.

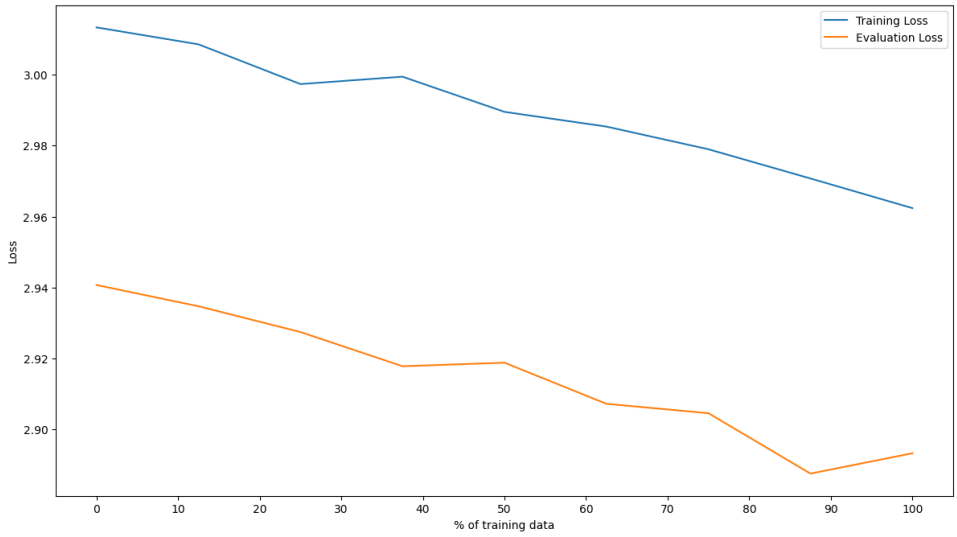


Figure 14: *GPT-2 Lithuanian model step 3 training and validation losses*

Finally Figure 14 shows results alike to the previous training step, with the losses gradually decreasing also indicating potential to extend training further.

Summary of the final losses and the total training times for a single GPU can be seen in Table 5.

Table 5: *Final losses and training times for each model*

Model	Training loss	Validation loss	Training time (hours)
ai-forever/mGPT	1.451	1.397	58.674
utter-project/EuroLLM-1.7B	1.658	1.557	56.486
GPT-2 Lithuanian	2.970	2.888	54.179

6 Evaluation

6.1 Evaluation criteria

To assess the performance of the Lithuanian language models, a diverse set of evaluation benchmarks is used. These benchmarks, which include popular language understanding metrics such as HellaSwag, GSM8K, and MMLU, are widely recognized as standard tools for evaluating large language models. They are designed to measure various aspects of language understanding and generation, offering a comprehensive view of the model's capabilities across multiple domains and tasks.

Because the chosen evaluation datasets were not originally available in Lithuanian, we utilize translated versions of these datasets, as provided in the "Open Llama 2 Model for Lithuanian Language" article [9]. The translated datasets make it possible to evaluate the model in its target language while maintaining the meaning and integrity of the original tasks.

6.1.1 Truthful QA

TruthfulQA [21] is a benchmark designed to assess the truthfulness and factual accuracy of language models. It consists of questions aimed at testing the model's ability to avoid generating plausible-sounding but incorrect information. The benchmark has two subsets: `truthfulqa_mc1`, which involves multiple-choice questions with a single correct answer, and `truthfulqa_mc2`, which involves multiple-choice questions with multiple correct answers. This dataset will help evaluate the model's ability to generate accurate and truthful content.

6.1.2 ARC

The AI2 Reasoning Challenge (ARC) [22] is a benchmark designed to evaluate a model's question-answering abilities on science-based questions, similar to those found in standardized exams. It requires not only factual knowledge but also reasoning and the application of that knowledge. The benchmark is divided into two subsets: `arc_easy`, which includes simpler questions, and `arc_challenge` which contains only questions answered incorrectly by both a retrieval-based algorithm and a word co-occurrence algorithm. This dataset will help to evaluate the model's scientific reasoning ability.

6.1.3 WinoGrande

WinoGrande [23] is a large scale commonsense reasoning dataset built to tackle the Winograd Schema Challenge [24]. It presents questions that require the model to resolve ambiguous words based on contextual understanding. This dataset will help to evaluate the models contextual and linguistic understanding.

6.1.4 MMLU

The Massive Multitask Language Understanding (MMLU) [25] dataset is a benchmark that spans multiple domains, including mathematics, history, technology and more. MMLU is useful for testing the model's general knowledge and domain-specific understanding across a variety of topics. By using this dataset, we can evaluate the model's ability to handle diverse subject matter.

6.1.5 HellaSwag

HellaSwag [26] is a dataset for commonsense reasoning that requires the model to complete a sentence or scenario in a way that aligns with human expectations. This dataset will be used to test the model's robustness and ability to understand contextually appropriate completions.

6.1.6 GSM8K

GSM8K [27] is a dataset focused on mathematical problem solving. It contains thousands of grade school level math problems. This dataset will assess the model's numerical reasoning skills and its ability to interpret and solve mathematical problems.

6.2 Evaluation results

To evaluate the models we use the Language Model Evaluation Harness[4] library, which provides a unified way to test language models. For the mGPT and EuroLLM models we perform evaluation on the initial and final model. For the GPT-2 Lithunian model we only perform evaluation after training is completed, as the initial model is untrained. The evaluation results for each benchmark can be seen in Table 6.

Table 6: *Evaluation of the original and trained models*

Benchmark	Original			Trained		
	mGPT	EuroLLM	GPT2-LT	mGPT	EuroLLM	GPT2-LT
arc_challenge	0.1962	0.2619	-	0.2125	0.1928	**0.2190**
arc_easy	0.3392	0.4937	-	0.3994	0.2719	**0.3996**
gsm8k	0.0053	0.0091	-	0.0076	0.0136	**0.0142**
hellaswag	0.2883	0.2755	-	0.3062	0.2785	**0.3070**
mmlu	0.2344	0.2482	-	0.2314	**0.2567**	0.2306
truthful_mc1	0.2509	0.2546	-	0.2558	**0.2913**	0.2705
truthful_mc2	0.4222	0.4094	-	0.4417	**0.4666**	0.4556
winogrande	0.4972	0.5022	-	**0.5257**	0.5028	0.5146

Overall, the results showcase how each model reacts differently to additional training, with some interesting patterns emerging across the various benchmarks.

[4] https://github.com/EleutherAI/lm-evaluation-harness

The mGPT model sees a slight increase in most of its benchmark scores following training but experiences a small drop on the MMLU metric. Despite that dip, it still performs relatively well on both ARC (Arc Challenge improves from 0.1962 to 0.2125; Arc Easy from 0.3392 to 0.3994) and HellaSwag (from 0.2883 to 0.3062). Notably, mGPT also achieves the highest final score in Winogrande (0.5257), edging out the other models by a narrow margin.

The most striking observation for EuroLLM is the significant drop in the ARC metrics. After training, EuroLLM's performance on ARC Challenge declines from 0.2619 to 0.1928, and on ARC Easy from 0.4937 to 0.2719. We hypothesize that this is due to how the original EuroLLM model was pretrained, where the last 10% of training involved high-quality educational resources that likely boosted the reasoning capabilities needed for ARC. After we perform additional training on our dataset, the model may have "forgotten" or acquired undesirable behavior with respect to this benchmark, causing the large decrease in performance. What is even more interesting is that EuroLLM's performance falls even below that of the mGPT and GPT2-LT models, both of which were trained on the same dataset. This suggests that additional factors, possibly related to the training process or the model's specific architecture, may be at play. However, we cannot pinpoint the exact cause. Despite this, EuroLLM remains competitive on other benchmarks, showing moderate gains in the TruthfulQA and MMLU metrics, where it manages to outperform the other models.

The GPT-2 Lithuanian model outperforms both mGPT and EuroLLM on several benchmarks by a narrow marigin: it achieves the highest scores on ARC Challenge (0.2190), ARC Easy (0.3996), GSM8K (0.0142), and HellaSwag (0.3070), though these gains over other models are only fractions of a percent. It also posts strong numbers on TruthfulQA, coming in close behind EuroLLM.

Looking across all final numbers, each model tends to perform within a few percentage points of the others on most tasks with the exception of EuroLLM on the ARC metrics. This shows that the current limiting factor of Lithuanian LLM development might not be the model architecture, but rather the quantity and quality of the training data. Another consistent observation is that all models score very low on GSM8K (staying around 1%), which is not surprising given the size of the models, quality of the data and knowing how LLMs often struggle with mathematical tasks. EuroLLM's slight advantage on MMLU and TruthfulQA likely stems from its exposure to more high-quality educational data during its pretraining stage, whereas GPT2-LT makes up ground on several other benchmarks thanks to the Lithuanian tokenizer.

We also compare the final results of the trained models to the existing Lt-Llama models.

Table 7: *Evaluation of the final and Lt-Llama models*

Metric	mGPT	EuroLLM	GPT2-LT	Lt-Llama-7B	Lt-Llama-13B
arc_challenge	0.2125	0.1928	0.2190	0.2176	**0.2491**
arc_easy	0.3994	0.2719	0.3996	0.4158	**0.5072**
gsm8k	0.0076	0.0136	0.0142	**0.0197**	0.0144
hellaswag	0.3062	0.2785	0.3070	0.3316	**0.4054**
mmlu	0.2314	**0.2567**	0.2306	0.2314	0.2303
truthful_mc1	0.2558	**0.2913**	0.2705	0.2717	0.2681
truthful_mc2	0.4417	**0.4666**	0.4556	0.4378	0.4222
winogrande	0.5257	0.5028	0.5146	0.5359	**0.6164**

Looking at the comparison in Table 7, we observe that the larger Lt-Llama models outperform the 1.3B parameter models in several benchmarks, including ARC, HellaSwag, and Winogrande. However, it is worth noting that Lt-Llama-13B is nearly ten times bigger than mGPT, EuroLLM, and GPT2-LT, yet its best performance increase over the smaller models hovers only at about 10% on the selected benchmarks.

Interestingly, GSM8K remains a challenge for all models, showing no clear improvement at the 13B scale, highlighting the intrinsic difficulty of mathematical reasoning tasks. Meanwhile, EuroLLM continues to hold the top spot on MMLU and TruthfulQA, despite having fewer parameters. This suggests that raw model size is not the only driving factor for better results. The type and quality of data, as well as the training strategy employed, can be equally important, if not more.

Overall, this table illustrates how scaling up model size helps with certain kinds of benchmarks. However, it also shows that there are scenarios like MMLU and TruthfulQA where specialized training and data selection allow more compact models to achieve strong or even superior performance.

Lastly, we provide some examples of the specific prompts from the ARC benchmark, to illustrate what knowledge the models acquired or lost during training. For instance, the mGPT model fails to answer the question in Table 8 before training, however after the additional training it answers the question correctly.

Table 8: *ARC-Easy example 1 with model answers*

Question:		
Augalai ir gyvūnai susideda iš organinių junginių. Kokie iš šių elementų dažniausiai randami organiniuose junginiuose? A: Geležis, deguonis, nikelis, varis. B: Natris, kaliumas, auksas, vandenilis. C: Helis, neonas, argonas, kriptonas. D: Anglis, vandenilis, deguonis, azotas.		
Correct answer	D	
Model answer	Original	Trained
mGPT	A	D
EuroLLM	D	D
GPT2-LT	-	D

Another example shows similar a result for EuroLLM. In Table 9 we see how the model answered this question incorrectly before training, however acquired the knowledge after.

Table 9: *ARC-Easy example 2 with model answers*

Question:		
Greitis, kuriuo garso bangos keliauja, priklauso nuo: A: Atstumo tarp virpesių šaltinio ir imtuvą. B: Medžiagos, per kurią keliauja garsas, tipo. C: Garsą skleidžiančio objekto dydžio. D: Garsą skleidžiančio įrenginio tipo.		
Correct answer	B	
Model answer	Original	Trained
mGPT	B	B
EuroLLM	D	B
GPT2-LT	-	C

Lastly, Table 10 shows the how the EuroLLM and mGPT models, contrary to the the previous examples, lose knowledge, and are not able to answer the specific question after training.

Table 10: *ARC-Easy example 3 with model answers*

Question:		
Kuri savybė apibūdina kačiuko kailio tekstūrą? A: Pilkas. B: Šiltas. C: Ilgas. D: Minkštas.		
Correct answer	D	
Model answer	Original	Trained
mGPT	D	A
EuroLLM	D	A
GPT2-LT	-	A

This example is especially interesting, because it indicates a potential issue with the translated evaluation datasets themselves. After training, when asked about the texture of a kitten's fur, all the models respond with ”pilkas”, which is incorrect. This may be due to the fact that the word ”pilkas” is more frequently associated with cats in Lithuanian text compared to ”minkštas”. Examining the original question in English, the correct answer is ”soft,” which, in the context of a kitten's fur, would be more accurately translated as ”švelnus”. When the answer is replaced with ”švelnus” and the evaluation is rerun, the models provide the correct response. This issue arises because the evaluation datasets were translated using ChatGPT, without human oversight to ensure accuracy and contextual alignment.

In conclusion, the evaluation results demonstrate the effects of additional training and model design on performance across various benchmarks. While mGPT model generally shows incremental improvements after training, EuroLLM highlights the potential risks of losing task specific capabilities due to data or architectural factors. The comparison with larger models like Lt-Llama shows the impact of model scale in achieving high performance. Furthermore, the challenges with translated evaluation datasets, particularly when relying on automated tools like ChatGPT without human oversight, emphasize the need for accurate translations of benchmarks to ensure fair and meaningful assessments of model capabilities.

7 Possible improvements

While model size and architecture do play a key role in performance, our findings suggest that the main limiting factor for Lithuanian language models is the availability of high-quality data. Here, we outline several potential avenues for improvement:

One of the most straightforward ways to enhance performance is to gather or generate more high-quality training data. A promising strategy, demonstrated in other work, involves leveraging larger language models such as ChatGPT to produce synthetic data [28]. Smaller models can then be trained or fine-tuned on these generated datasets, often resulting in significant performance gains. Another possibility is to adopt a strategy similar to EuroLLM, where the final stages of pretraining are dedicated solely to high-quality educational resources. This approach may help retain or boost the reasoning capabilities necessary for tasks such as ARC.

Improving the translation quality of evaluation datasets could be another step toward enhancing Lithuanian language models. Current evaluation datasets are translated using ChatGPT, which lacks the contextual depth that human oversight can provide. This limitation can lead to inconsistencies, inaccuracies and incorrect translations, which may affect the evaluation outcomes. To address this, future efforts could involve humans to review and refine the translated datasets, ensuring that linguistic subtleties and contexts are accurately represented.

Our comparisons highlight that larger models, such as Lt-Llama-13B, tend to outperform smaller ones on several benchmarks. Therefore, one potential path for improvement is to use even larger base models. By combining these large models with modern fine-tuning techniques such as LoRA [29], researchers could mitigate the increased computational costs while still tailoring the models to Lithuanian data or to specific tasks.

An alternative approach is to build a smaller, dedicated Lithuanian model. Currently, our GPT-2 Lithuanian model inherits the tokenizer parameters, including the 100,000 token vocabulary, directly from the multilingual mGPT model. While this larger vocabulary suits multilingual contexts, it may be oversized for purely Lithuanian text. A more carefully designed tokenizer, together with an improved dataset, could lead to further performance gains. This smaller model strategy is especially appealing where computational resources or inference speed are constrained, as smaller models are typically easier to deploy and train.

All of our models are base models, so a more task oriented fine-tuning regimen, particularly for question answering, could yield significant gains on metrics like ARC, TruthfulQA, or other reasoning-based tasks. Additionally, extending the training to leverage longer context windows might help the models handle more complex or context heavy prompts. From the training loss graphs, it appears that the models have not completely plateaued, suggesting that training for more epochs could further improve results.

Overall, each of these strategies, whether increasing model scale, refining data quality and tokenizer design, or performing more targeted fine-tuning offers clear pathways to enhance Lithuanian language models. By combining multiple improvements, future work can aim for even greater performance across a range of benchmarks.

Conclusions

In this work, we explored the challenges of training Lithuanian language models, highlighting the scarcity of high-quality data and the difficulty of adapting existing multilingual models to a low-resource language. Our evaluation results, using a variety of benchmarks ranging from commonsense tasks to reasoning intensive tasks, underscore several key observations:

- **Impact of additional Training.** The mGPT and EuroLLM models both benefited from further fine-tuning, demonstrating small but consistent performance improvements in tasks like HellaSwag and ARC. However, EuroLLM experienced a notable drop on ARC after training, suggesting that specialized high-quality data like the educational resources used at the end of its original training plays a critical role in strong reasoning performance.
- **Results of a dedicated Lithuanian model.** Our GPT-2 Lithuanian model, trained from scratch with a Lithuanian tokenizer, performed competitively across most tasks, even marginally surpassing the other models in benchmarks such as ARC and HellaSwag. Though the gains were small, this highlights the potential for specialized, language-specific designs.
- **Role of model size.** Larger models tend to outperform smaller ones on certain benchmarks like ARC, HellaSwag, and Winogrande. However, these gains are not as pronounced as one might expect for a model that is nearly ten times larger.
- **Importance of high quality Lithuanian data.** Across all models, the results repeatedly point toward the need for bigger, higher quality Lithuanian datasets. Whether through carefully curated resources, synthetic data generation from large teacher models, or extended fine-tuning on domain-specific texts, improvements in data quality appear more pivotal than simply increasing model size or changing the architecture.

These results suggest several potential directions for future work. For instance, developing more specialized tokenizers for Lithuanian, collecting or generating higher-quality datasets, and trying advanced fine-tuning methods like LoRA on larger models. By combining better data, improved tokenization, and modern training strategies, we can continue to improve Lithuanian language models.

References and sources

[1] N. R. Robinson, P. Ogayo, D. R. Mortensen, G. Neubig. *ChatGPT MT: Competitive for High- (but not Low-) Resource Languages*. 2023. URL: https://arxiv.org/abs/2309.07423.

[2] M. A. Hasan, P. Tarannum, K. Dey, I. Razzak, U. Naseem. *Do Large Language Models Speak All Languages Equally? A Comparative Study in Low-Resource Settings*. 2024. URL: https://arxiv.org/abs/2408.02237.

[3] K. Dey, P. Tarannum, M. A. Hasan, I. Razzak, U. Naseem. *Better to Ask in English: Evaluation of Large Language Models on English, Low-resource and Cross-Lingual Settings*. 2024. URL: https://arxiv.org/abs/2410.13153.

[4] A. Nag, S. Chakrabarti, A. Mukherjee, N. Ganguly. *Efficient Continual Pre-training of LLMs for Low-resource Languages*. 2024. URL: https://arxiv.org/abs/2412.10244.

[5] A. Petrov, E. L. Malfa, P. H. S. Torr, A. Bibi. *Language Model Tokenizers Introduce Unfairness Between Languages*. 2023. URL: https://arxiv.org/abs/2305.15425.

[6] F. Cassano, J. Gouwar, F. Lucchetti, C. Schlesinger, et al. *Knowledge Transfer from High-Resource to Low-Resource Programming Languages for Code LLMs*. 2024. URL: https://arxiv.org/abs/2308.09895.

[7] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, C. Raffel. *ByT5: Towards a token-free future with pre-trained byte-to-byte models*. 2022. URL: https://arxiv.org/abs/2105.13626.

[8] P. H. Martins, P. Fernandes, J. Alves, N. M. Guerreiro, et al. *EuroLLM: Multilingual Language Models for Europe*. 2024. URL: https://arxiv.org/abs/2409.16235.

[9] A. Nakvosas, P. Daniušis, V. Mulevičius. *Open Llama2 Model for the Lithuanian Language*. 2024. URL: https://arxiv.org/abs/2408.12963.

[10] O. Shliazhko, A. Fenogenova, M. Tikhonova, V. Mikhailov, A. Kozlova, T. Shavrina. *mGPT: Few-Shot Learners Go Multilingual*. 2023. URL: https://arxiv.org/abs/2204.07580.

[11] A. Petrov. *Tokenization fairness*. URL: https://aleksandarpetrov.github.io/tokenization-fairness/ (viewed 2025-01-02).

[12] T. Dettmers, M. Lewis, S. Shleifer, L. Zettlemoyer. *8-bit Optimizers via Block-wise Quantization*. 2022. URL: https://arxiv.org/abs/2110.02861.

[13] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, C. Raffel. *mT5: A massively multilingual pre-trained text-to-text transformer*. 2021. URL: https://arxiv.org/abs/2010.11934.

[14] C. C. Organization. *Common Crawl - Open Repository of Web Crawl Data*. 2024. URL: https://commoncrawl.org/ (viewed 2024-10-10).

[15] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, N. Carlini. *Deduplicating Training Data Makes Language Models Better*. 2022. URL: https://arxiv.org/abs/2107.06499.

[16] Z. Ankner, C. Blakeney, K. Sreenivasan, M. Marion, M. L. Leavitt, M. Paul. *Perplexed by Perplexity: Perplexity-Based Data Pruning With Small Reference Models*. 2024. URL: https://arxiv.org/abs/2405.20541.

[17] T. Nguyen, C. V. Nguyen, V. D. Lai, H. Man, N. T. Ngo, F. Dernoncourt, R. A. Rossi, T. H. Nguyen. *CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages*. 2023. URL: https://arxiv.org/abs/2309.09400.

[18] M. Ali, M. Fromm, K. Thellmann, R. Rutmann, et al. *Tokenizer Choice For LLM Training: Negligible or Crucial?* 2024. URL: https://arxiv.org/abs/2310.08754.

[19] B. Zhang, P. Williams, I. Titov, R. Sennrich. "Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Edited by D. Jurafsky, J. Chai, N. Schluter, J. Tetreault. Online: Association for Computational Linguistics, 2020, pages 1628–1639. https://doi.org/10.18653/v1/2020.acl-main.148. URL: https://aclanthology.org/2020.acl-main.148.

[20] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, et al. *The Llama 3 Herd of Models*. 2024. URL: https://arxiv.org/abs/2407.21783.

[21] S. Lin, J. Hilton, O. Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2021.

[22] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, O. Tafjord. "Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge." In: *ArXiv* abs/1803.05457 (2018).

[23] K. Sakaguchi, R. L. Bras, C. Bhagavatula, Y. Choi. "WinoGrande: An Adversarial Winograd Schema Challenge at Scale." In: (2019). URL: https://arxiv.org/abs/1907.10641.

[24] H. J. Levesque, E. Davis, L. Morgenstern. "The Winograd Schema Challenge." In: *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. 2011. URL: https://api.semanticscholar.org/CorpusID:15710851.

[25] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, J. Steinhardt. "Measuring Massive Multitask Language Understanding." In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).

[26] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, Y. Choi. "HellaSwag: Can a Machine Really Finish Your Sentence?" In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[27] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, et al. "Training Verifiers to Solve Math Word Problems." In: (2021). URL: https://arxiv.org/abs/2110.14168.

[28] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, H. Hajishirzi. *Self-Instruct: Aligning Language Models with Self-Generated Instructions*. 2023. URL: https://arxiv.org/abs/2212.10560.

[29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. URL: https://arxiv.org/abs/2106.09685.

[30] OpenAI. *ChatGPT*. 2024. URL: https://www.openai.com/ (viewed 2025-01-04).

[31] D. Science. *Writefull*. 2024. URL: https://www.writefull.com/ (viewed 2025-01-04).

[32] Grammarly. *Grammarly*. 2024. URL: https://www.grammarly.com/ (viewed 2025-01-04).

Appendix 1. Code and models

The code used for data processing, training, and evaluating the models in this study is publicly available at `https://github.com/pledominykas/Magistras`.

The modified datasets and trained models are available on huggingface via the following links:

- Dataset: `https://huggingface.co/datasets/domce20/c4-lithuanian-enhanced`
- mGPT Lithuanian: `https://huggingface.co/domce20/mGPT-Lithuanian`
- EuroLLM Lithuanian: `https://huggingface.co/domce20/EuroLLM-1.7B-Lithuanian`
- GPT-2 Lithuanian: `https://huggingface.co/domce20/GPT2-Lithuanian`

Appendix 2. List of candidate models

Table 11: *List of open causal language models with less than 2B parameters*

Model	Parameters (Billion)
ai-forever/mGPT	1.3
EleutherAI/gpt-neo-1.3B	1.3
meta-llama/Llama-3.2-1B	1.2
bigscience/bloomz-1b7	1.7
DAMO-NLP-MT/polylm-1.7b	1.7
utter-project/EuroLLM-1.7B	1.7
openai-community/gpt2-xl	1.6
facebook/opt-1.3b	1.3
stabilityai/stablelm-2-1_6b	1.6

Appendix 3. Extended tokenization experiment

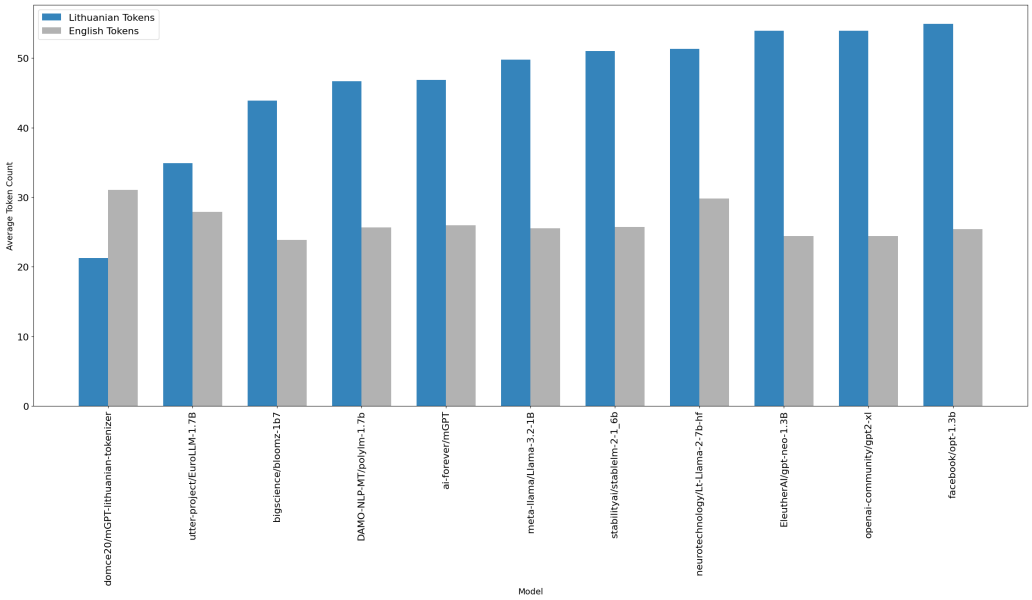


Figure 15: *Average token counts of 1000 Lithuanian and English sentence pairs for candidate, GPT-2 Lithuanian and Lt-Llama models*

Appendix 4. Use of artificial intelligence

During the writing of this thesis, artificial intelligence tools were used to improve text quality, readability, and correct mistakes. The following tools were utilized:

- ChatGPT [30]: For refining phrasing, rewriting content, and fixing stylistic issues.
- Writefull [31]: Inbuilt Overleaf tool used to improve writing by providing real-time language suggestions and ensuring linguistic accuracy.
- Grammarly [32]: For grammar correction, spelling checks, and enhancing text clarity.