



School of Information Technology and
Engineering at the ADA University



School of Engineering and Applied Science
at the George Washington University

DEVELOPMENT OF GENERATIVE MODELS FOR AZERBAIJANI TEXT

A Thesis

Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Eljan Mahammadli

Supervisor Dr. Samir Rustamov

April 2024

THESIS ACCEPTANCE

This Thesis by: Eljan Mahammadli

Entitled: *Development of Generative Models for Azerbaijani Text*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

(Adviser)

(Date)

(Program Director)

(Date)

(Dean)

(Date)

ACADEMIC INTEGRITY STATEMENT

“I affirm that this is my own work, I attributed where I used the work of others, I did not facilitate academic dishonesty for myself or others, and I used only authorized resources for my Thesis, per the ADA University Academic Integrity requirements. If I failed to comply with this statement, I understand consequences will follow my actions. Consequences may range from failing the course to expulsion from the program/university and may include a transcript notation.”

Eljan Mahammadli

30.04.2024

(Full Name)

(Signature)

(Date: DD.MM.YY)

Acknowledgments

I would like to thank Professor Samir Rustamov for his excellent guidance and support throughout my research. His help has been essential to my success. I am thankful to ADA University and George Washington University for providing the facilities and opportunities that greatly aided my work. Special thanks to my dear friend Ayna Aslanova, and to my friends and colleagues Ali Asgarov, Yusif Mukhtarov, and Toghrul Tahirov for their friendship and encouragement. I am deeply grateful to my family for their endless love and support, which has always motivated me. Lastly, thank you to everyone who contributed to this thesis. Your help has made a significant difference.

Abstract

One of the significant problem is the lack of open-source, generative-based small language models available for the Azerbaijan language, an issue which remains despite the existence large-scale, privately funded offerings such as OpenAI GPT-4 or Claude Opus. Given the urgent demand for more serviceable models with the potential to both produce and comprehend Azerbaijani, this study marks the first-ever endeavor to pre-train a generative-based small language model in a targeted manner for this minority language.

While existing large models are proprietary and the trend is moving towards open-weight or open-source Large Language Models with Fine-tuned weights for specific tasks, the best-case scenario is given developed model. Unlike many other languages, it is virtually impossible to fine-tune a dedicated model in Azerbaijani in the absence of either comprehensive or clean datasets for pre-training powerful models to be measured in billions of tokens.

Considering the outlined issues, the research employed the collection of the largest Azerbaijani text corpus available to date, estimated at nearly 3 billion tokens, from a breadth of sources, including Wikipedia, OSCAR, mC4, books, news, and other scraped data. The dataset curation process was thorough, and many filtering steps were taken, including data sampling and preparation, prior to the model training. A 150-million-parameter decoder-only pre-trained language model was then created with the use of Llama2 architecture to facilitate autoregressive text generation in Azerbaijani. Afterward, the model was instruction-fine-tuned with the Alpaca instruction dataset, consequently turning it into a question-answering model in Azerbaijani.

Although the model is less powerful, as evidenced by its smaller size containing only 150 million parameters, compared to larger multi-billion parameter models, it is unique in its ability to generate text in Azerbaijani with few mistakes and answer simple questions. This is a significant outcome for researchers given that the other open-source models, including the largest multi-billion ones with billions of parameters trained on trillions of tokens, cannot demonstrate such linguistic proficiency in the Azerbaijani language. As such, the present study is a vital first step in the field of Azerbaijani language processing, allowing for potential improved performance.

The work presents a foundation for further research and applications of fine-tuned generative models in NLP for low-resource languages. It also confirms the possibility of creating effective language models and shows that the size of model training data and computational resources could be easily increased by the scaling laws of machine learning. Thus, the described project fills a notable gap in Azerbaijani language processing and gives a way to research and produce further applications for NLP. The work could benefit the general NLP area by showing the potential of fine-tuned generative models to be used as a tool for enhancing language understanding and multimodal generation and shedding light on prospects for their further improvement and applications for other low-resource languages.

The source code and supplementary materials for this project are available on GitHub at <https://github.com/eljanmahammadli/AzLlama> for replication and further research.

Contents

1	Introduction	11
1.1	Background	11
1.2	Problem Statement	11
1.3	Objectives of the Research	11
1.4	Significance of the Study	11
1.5	Scope and Limitations	11
1.6	Contributions	12
2	Literature Review	13
2.1	Generative Models in NLP	13
2.1.1	Early Methods in NLP	13
2.1.2	Current State of the Art in Generative Models	16
2.2	Challenges of Low-Resource Languages	17
2.2.1	Definition of Low-Resource Languages	17
2.2.2	Specific Challenges: Data Scarcity	17
2.2.3	Strategies for Overcoming Challenges	19
2.2.4	Integration with Broader Research Goals	21
2.3	Gap in Research for Azerbaijani Language Processing	21
3	Dataset Acquisition and Curation	23
3.1	Source of Data	23
3.2	Dataset Curation: Cleaning and Filtration Techniques	24
3.2.1	Document Modification	24
3.2.2	Heuristic-based Document Filtering	25
3.2.3	Machine Learning Document Filtering	26
3.2.4	Deduplication	27
3.2.5	Personally Identifiable Information Removal	27
3.3	Challenges and Solutions in Data Curation	27
4	Model Development	29
4.1	Overview of Model Architecture	29
4.2	Pre-training the Language Model	34
4.2.1	Training Data Preparation	34
4.2.2	Training Process and Parameters	36
4.3	Instruction Fine-tuning	39
4.3.1	Foundation Model vs. Supervised Fine-tuning	39
4.3.2	Instruction-following dataset	40
4.3.3	Training Details and Adjustments for Azerbaijani Language	40
4.3.4	Challenges with Translated Data and Domain-Specific Content	42
5	Model Evaluation	43
5.1	Evaluation Metrics	43
5.1.1	Contextual Alignment Score	43
5.1.2	Semantic Scores Using Sentence Transformers	44
5.1.3	Character n-gram F-score (CHRF)	45
5.2	Performance Comparison with Open-Weight and Proprietary Models	46
5.3	Strengths and Limitations of the Developed Model	46
6	Conclusion	48

6.1	Potential Use Cases in Azerbaijani	48
6.1.1	Implications for Low-Resource Language Processing	48
6.2	Summary of Findings	48
6.3	Concluding Remarks	49
6.4	Recommendations for Future Work	49
A	Appendix	56
A.1	Parameters for AzLlama-150M Foundation (Base) Model	56
A.2	Parameters for AzLlama-150M Instruction Following Model	59
A.3	Model Architecture	60
A.4	Question and Answer Interactions with AzLlama-150M Model	60
A.5	Pre-trained model (not instruction following)	63
A.6	Additional training metrics	64
A.7	GPU Usage	64
A.8	Parameter Counts	64
A.9	Chat-bot UI	64
A.10	Code snippets for Pre-Training	65

List of Figures

FIGURE 1.	Architectural Comparison of Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM) models (Dancker, 2022)	15
FIGURE 2.	Transformer Model Architecture: Overview of the key components, including attention mechanisms, feed-forward layers, and positional encodings Vaswani et al. (2017).	15
FIGURE 3.	Timeline of Large Language Models Over 10B: This figure highlights key release dates and evaluates publicly accessible models, marked in yellow (Zhao et al., 2023)	17
FIGURE 4.	Filtering pipeline for AzLlama training corpus of 3B tokens.	25
FIGURE 5.	Stages for each filtering pipeline. The diagram outlines a multi-stage data processing workflow with percentages indicating the proportion of data remaining after each step. Starting with 'mC4' at 79%, it proceeds to 'Language Identification' at 97%, followed by 'Heuristic Based Filtering' at 78%, 'ML Filtering' at 77%, and concludes with 'Deduplication' at 68%. Supplementary sources of data include 'OSCAR' at 14%, 'Additional' at 4%, and 'Wikipedia' at 3%.	26
FIGURE 6.	LLaMA Architecture Ibe (2024)	30
FIGURE 7.	Comparison of Multi-Head, Multi-Query, and Grouped Query Attention (GQA) mechanisms Ainslie et al. (2023).	31
FIGURE 8.	Training loss. In practice, bigger models tends to have lower loss when compared to lower counterparts. But here we observe the vice versa. The reason for this is that first the small models (AzLlama-14M) has been only trained only Wikipedia while the biggest model has been trained on all the available data including internet. And the second reason is that the 150M model is undertrained which means upon training additional epochs would end up in the convergence (See Figure 11).	37
FIGURE 9.	Validation loss	37
FIGURE 10.	Learning Rate Schedule for AzLlama: The model employs a learning rate schedule that includes a warmup phase for the initial 10,000 steps out of a total of 90,000 steps, followed by a cosine decay. The maximum learning rate is set at 6e-4, with a minimum learning rate of 6e-5. This approach helps in stabilizing the training process early on and gradually fine-tuning the learning rate to enhance model performance over time.	39
FIGURE 11.	Number of Global Training Tokens for AzLlama: Currently, AzLlama has been trained on 5 billion tokens, which suggests potential under-training. To optimize its performance, there is scope to extend training for one or two additional epochs, potentially increasing the training volume to between 9 and 12 billion tokens. This expansion could significantly enhance the model's understanding and generation capabilities in Azerbaijani.	39
FIGURE 12.	Training loss on the instruction following dataset.	41
FIGURE 13.	General Knowledge Question 1	61
FIGURE 14.	General Knowledge Question 2	61
FIGURE 15.	General Knowledge Question 3	62
FIGURE 16.	General Knowledge Question 4	62
FIGURE 17.	Factual Question 2	62
FIGURE 18.	Factual Question 2	63

FIGURE 19. Few-shot prompting the base model (not instruction following). Since the base model is not able to answer questions as an assistant it is possible to use few-shot prompts to get answers to the question but it is not as stable as the instruction following model. As mentioned before base model is essentially a document completer 63

FIGURE 20. Base model also does not know when to stop and can generate duplicated sentences if repetition penalty is not applied. It is given a sentence and then the model continues to generate more tokens. It is not possible to ask questions and get answers contrary to the instructions following the model. 63

FIGURE 21. Validation perplexity 64

FIGURE 22. GPU Utilization 64

FIGURE 23. Chat-bot User Interface for AzLlama-150M 65

FIGURE 24. Configuring and Executing Distributed Training Sessions: This Python code is for setting up and executing a distributed training session of a GPT model. The script incorporates initial configuration, data loader preparation, model instantiation, and optimizer setup. It also handles conditional resumption from saved checkpoints, providing detailed output about the training process and resource utilization. 66

FIGURE 25. Optimizing Training Workflows: Monitoring, Efficiency, and Dynamic Adaptation: Code snippet highlights optimizing training workflows for a GPT model. The script involves validation checks, performance monitoring through TFLOP calculations, and dynamic training iteration management based on data distribution across computing nodes. Additionally, it handles efficient resume operations from previous training states and dynamic learning rate adjustments, enhancing both training speed and model performance under distributed computing environments. 67

FIGURE 26. Gradient Management and Performance Monitoring: This code is for fine-tuning the training process of an AI model. The script uses advanced gradient accumulation techniques to optimize resource utilization and reduce training time. It includes detailed performance logging and monitoring, dynamically adjusting the learning rate, and efficiently managing checkpoints. The code also handles validation at specific intervals to ensure model accuracy. 68

FIGURE 27. Streamlining Model Validation and Optimizing Learning Rates: Those utility functions are for validation and learning rate adjustment during AI training. The script includes a validate function for efficiently assessing model performance on validation data without affecting gradients, and a get learning rate function that dynamically adjusts the learning rate based on training progress, ensuring optimal convergence. Additionally, an initialization of weights function sets initial weights of neural network components to support stable training. 69

List of Tables

TABLE 1.	Statistics of Wikipedia Articles by Language (Sizes in Parquet Files) . . .	17
TABLE 2.	Document and token counts for selected languages from the CulturaX dataset, which includes data from mC4 and OSCAR corpora. This dataset encompasses 16TB of data in parquet format, expanding to 27TB when unpacked, and dedicates over half of its resources to non-English languages to support multilingual model training (Nguyen et al., 2023). . .	18
TABLE 3.	AzLlama-150M Model Architecture Parameters: Hidden Size refers to the dimensionality of the hidden layers in the model, affecting the capacity to learn complex patterns. Intermediate Size specifies the size of the feed-forward layer within each transformer block, which is crucial for processing information between attention layers. Max Position Embeddings defines the maximum number of tokens the model can process in a single sequence, determining the scope of context the model can consider. The number of Attention Heads indicates how many parts the model divides its attention mechanism into, allowing it to simultaneously attend to different parts of the input sequence from different representational perspectives. The number of Hidden Layers represents the depth of the model, where each layer can learn a different level of abstraction. The number of Key-Value Heads specifies how many sets of key and value pairs are used in the attention mechanism, which can affect the model’s ability to differentiate and prioritize different aspects of the input data. . .	34
TABLE 4.	Training Parameters and Performance Metrics of the AzLlama-150M Model	38
TABLE 5.	Configuration Parameters of the Instruction Following Model	41
TABLE 6.	(Large) Language Models (LLMs) Information	43
TABLE 7.	Contextual Alignment Metrics	44
TABLE 8.	Semantic Scores	45
TABLE 9.	Character n-gram F-score (CHRF)	46
TABLE 10.	Configuration of the Training Model	56
TABLE 11.	Additional Configuration Parameters of the AzLlama-150M Foundation (Base)	57
TABLE 12.	Summary Metrics of the Model Training	58
TABLE 13.	Key Configuration Parameters of the Model	59
TABLE 14.	Training Statistics of the Model	59
TABLE 15.	Grouped parameter counts by layer type	64

LIST OF ABBREVIATIONS

Abbreviation **Explanation**

GPT	Generative Pre-trained Transformer
LLM	Large Language Model
API	Application Programming Interface
RNN	Recurrent Neural Network
BPE	Byte Pair Encoding
NLP	Natural Language Processing
LLaMA	Large Language Model Meta AI
OSCAR	Open Super-large Crawled ALMAnaCH coRpus
mC4	Multilingual Common Crawl
MMLU	Massive Multitask Language Understanding
ARC	AI2 Reasoning Challenge
SFT	Supervised Fine-Tuning
GPU	Graphics Processing Unit
VRAM	Video Random Access Memory
Wandb	Weights & Biases
KV	Key-Value
RoPE	Rotary Positional Embedding
SwiGLU	Sigmoid-Weighted Linear Unit
GQA	Grouped Query Attention
MHA	Multi-Head Attention
PII	Personally Identifiable Information
ML	Machine Learning
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
RMSNorm	Root Mean Square Layer Normalization
AdamW	Adam Optimizer with Weight Decay
BERT	Bidirectional Encoder Representations from Transformers
T5	Text-to-Text Transfer Transformer

1 Introduction

1.1 Background

The development of Language Large Models benefits mostly well-resourced languages which barely leaves any technological support for low-resource ones, such as the Azerbaijani language. Nowadays, LLMs for such languages are typically developed either as a translation from English, which effectively depends on the quality of the translation, or through the creation or adaptation of large proprietary methods, like any currently available GPT-4 alternatives. However, the latter have reduced performance for languages other than English and require substantial overhead due to their model size, and the necessity of using a paid API. Moreover, even in the case of large-scale availability of open-source models that incorporate multi-language setup, such models do not process Azerbaijani effectively, as their tokenizers process Azerbaijani text as individual characters without halting them to identify any words or even subwords.

1.2 Problem Statement

There currently are no effective language models specifically trained for Azerbaijani language. There are massive multilingual models as well as English-centric models but there is no strong foundational generative model capable of understanding and generating text in Azerbaijani. This seriously stifles the development of any NLP applications specifically aimed at Azerbaijani-speaking population.

1.3 Objectives of the Research

The aim of the research is to pre-train a small, decoder-only language model from scratch on a basis of 3 billion of Azerbaijani tokens based in LLaMA architecture and to create a SentencePiece BPE tokenizer designed for use with Azerbaijani language, which would ensure that the model can understand and generate semantically meaningful Azerbaijani text, have a good feel of grammatical peculiarities of Azerbaijani language, and be able to answer simple questions with appropriate prompt fine-tuning.

1.4 Significance of the Study

To sum it up, the study is the initial stage to develop a pre-trained base model and a special tokenizer for Azerbaijani and can be the basis for further research. Theoretical and practical implications of the study may be interesting for the field of NLP, as it shows that the creation of NLP models for low-resource languages can be scalable and similar in the majority of cases. At the same time, the case study offers additional points regarding the issue of privacy and the cost that can serve as more arguments for the need to develop and use open-source models.

1.5 Scope and Limitations

This research's focus is limited to a maximum context length of 512 tokens, produced due to data and computational restraints we have. While this confines the scale of incomprehensible behaviors of larger hallmark models, it also narrows the model's depth of context and applicability. Furthermore, even though the model appears to have potential for generating Azerbaijani texts and answering simple questions, it is not yet production ready, as it needs to be scaled in terms of data size and model parameters.

Although the developed model showcases the ability to generate fluent Azerbaijani text and gives accurate responses to other queries, it still has a significant drawback of being prone to

irreal output. Generating hallucinated content (Liu et al., 2024) is one of the most common issues connected with the language large models and can be observed to a greater extent in smaller models than in the ones that contain more parameters. Hallucinations imply that the model is feeding the user with the information that is not true or does not make sense, although it is represented as if it was a fact. The reason for such inaccuracy is that the model fails to understand it, which goes back to its incapability to structure its learning and outcomes correctly. Smaller models do not have sufficient memory and advanced training data to clearly understand what is a reliable pattern or a sort of a noise, as opposed to large models. Reduced attention span and a limited number of tokens shortened to a 512 token capacity are the aggravating factors why the model cannot maintain the coherence of texts. In such a way, the developed model can generate fluent text and give plausible answers to simple questions, practically, its application would be severely restricted for the reason of hallucinatory outputs, and users should be attentive while utilizing the model.

1.6 Contributions

This thesis represents a significant advancement in the field of language technologies, focusing on the Azerbaijani language within the broader context of natural language processing. By developing and utilizing a set of sophisticated tools and methodologies, this research not only enhances the capability of language models for Azerbaijani but also sets a foundation for further academic and practical applications in low-resource language processing.

1. **Development of a Large Corpus:** Creation of a 3 billion token corpus, the largest of its kind, designed specifically for language modeling and text generation tasks in Azerbaijani.
2. **Pre-training of a Foundation Model:** Development of a 150 million parameter LLaMA-based model, pre-trained from scratch, establishing a robust foundation for further language-specific applications.
3. **Instruction Following Fine-Tuning:** Fine-tuning of the pre-trained model on an instruction-following dataset, enhancing its ability to follow simple commands in Azerbaijani.
4. **Evaluation Dataset Compilation:** Assembly of a comprehensive evaluation dataset encompassing general knowledge, derived from 11 different LLMs, including 8 open-source models (like LLaMA 2, LLaMA 3, Mistral, Mixtral, Gemma) and 3 proprietary models (such as Claude Haiku, GPT-4, and the custom-developed AzLlama).
5. **Advanced Data Filtering Techniques:** Implementation of a comprehensive data filtering pipeline, designed specifically to the Azerbaijani language. This includes the development of a KenLM Kneser-Ney trigram model for rigorous text filtering, trained on Azerbaijani Wikipedia content.

2 Literature Review

2.1 Generative Models in NLP

Generative models are a class of statistical models in natural language processing created to generate new data instances that would correspond to the existing distribution. In NLP, such models are used to produce purposive and communicative pieces of text that correlate with a given context and clearly express the point. Generative models are important as they allow to understand and work with language and are used in numerous applications related to natural language processing.

The first example of the model's application is text generation where generative algorithms produce not only short replies given by a chat-bot but also whole articles that can be almost indistinguishable from human-written ones. Another important application is machine translation where models are used to convert the meaning of a word, a phrase, or a whole text from one language to another avoiding the necessity of creating a separate model for every possible language combination. Finally, generative models are used in conversational AI to formulate a reply that would not only be accurate but appropriate in the given context, thus making possible more naturally looking and thoughtful conversations.

Generative Adversarial Networks or GANs is another type of generative model that combines two models controlling their training. In a GAN model, generator network creates random samples of data while the discriminator network evaluates their performance, thus forcing both models to improve their skills. It creates the need for generator to enhance the quality of samples produced.

2.1.1 Early Methods in NLP

The earliest NLP systems were mainly rule-based, and they used handcrafted rules. Developed in the 1950s and 1960s, the systems were usually informed by formal linguistics, with a substantial part of the rules being directly encoded into the computer in terms of syntax and grammar. An example of such a system included the SHRDLU developed by Winograd (1972). The system was created in the early 1970s and showed how a computer could generally understand natural language within a controlled environment of "blocks world." The secret was that the system had to be developed in a world programmed to understand it. This was achieved using rules about English and the model world to create a program that elaborately explained how it perceived the world.

There was a substantial change from rule-based to statistical methods since the late 1980s and 1990s. One of the significant changes in NLP processing involved statistical machine translation (SMT), the first of which were developed in the 1990s by IBM. Statistical translation models developed by IBM are a series, the first of which included the IBM Model 1 that was developed in 1990. These models learned to translate from considering analyzing thousands of sentences from bilingual text corpora for translation compared to rule-based models, which extensively demanded grammatical rules to operate (Brown et al., 1990).

The systems also used probability models such as Hidden Markov models and N-gram models, the use of which was particularly rampant in speech recognition and part of speech tagging. HMMs in particular model the sequence and other time series model that NLP task concerned. N-gram model was particularly useful at the time and formed the basis for the first speech recognition and machine translation systems (Rabiner, 1989). The introduction of machine learning concepts such as decision trees, support vector machines and random forests in the late 90s and 2000s saw the modeling of NLP tasks that could be done such as elementary tasks in classification, e.g. sentiment analysis and letters sorting like document tagging. The new methods significantly improved the performance of the NLP tasks since the previous rules

were pieces of handmade decision boundaries.

The introduction of neural network-based approaches in the early 2000s was a significant development in NLP. One of the critical papers in this regard is Bengio et al. (2003) in which he proposed a new framework for developing language models using neural networks. The model developed by Bengio sought to learn a distributed representation for words, which would be then utilized to predict the next word in a sequence. Thus, it provided a conceptually appealing alternative to traditional N-gram models. The neural probabilistic language model employed a feedforward neural network with a novel architecture that included a projection layer where words are represented as dense vectors. This helped to alleviate the dimensionality issues that plague traditional language models and enabled this model to capture complex relationships between words more effectively. The NPLM was an important milestone in the development of more sophisticated deep learning models in NLP, such as RNNs and eventually Transformers.

A decade later, Mikolov et al. (2013) at Google in 2013 introduced Word2Vec, an approach to learning a learned dense vector representation of words from a large corpus. The primary innovation of Word2Vec is using shallow neural networks to learn word associations from a large corpus. There are two versions of training a Word2Vec model: Continuous Bag of Words and Skip-Gram. Word2Vec effectively captures a vast, extremely accurate, yet narrow window of syntactic and semantics word relations, substantially impacting the performance of associated NLP tasks, like text classification, and text generation, when used in conjunction with RNNs or LSTMs.

GloVe, which stands for Global Vectors for Word Representation, is an algorithmic word embedding technique developed by the researchers at Stanford University. Described by Pennington et al. (2014) Pennington et al. , the model builds on an explicit piece of count data, derived from a given word's co-occurrence matrix. Through application of matrix factorization methods, it becomes possible to approximate the matrix and produce sound semantic and syntactic word embeddings. Although different in implementation, Word2Vec and GloVe are associated with the use of pre-trained word embeddings, which can be applied as a first layer in the RNN and LSTM network. As a result, the network has an opportunity to produce and understand natural speech thanks to the use of pre-determined word contexts.

RNN or Recurrent Neural Networks heralded a new potential for sequential data processing, as they can remember the past in order to better understand the future. Introduced issues of vanishing and exploding gradients could further be mitigated by the use of more advanced neural networks such as LSTM or Long Short-Term Memory and GRU – Gated Recurrent Units (Quast, 2016; Lent et al., 2022; Hochreiter and Schmidhuber, 1997). RNN are considered as an effective method of text generation, and there are no obstacles that may prevent the use of Word2Vec and GloVe embeddings. In fact, the latter could be associated with a better understanding of words and their relationships, which is critical for text generation. By initializing the embedding layer with pre-trained word vectors, RNNs and LSTMs can effectively leverage prior knowledge and achieve better performance even with smaller training datasets or more complex generative tasks.

The circuit of expert alternative both on embeddings and different models gave rise to the development of the Transformer architecture by Vaswani et al. (2017) who proposed a self-attention mechanism. The previously developed RNNs and LSTMs, as well as other architectures, processed the data sequentially, and such an approach consistently limited the speed and scalability, especially when working with long sequences provided in the input. Particularly, older models had difficulties with long-term dependencies, such as those indicating the vanishing and exploding of gradients. However, the Transformer permitted performing self-attention so that the importance of all other parts in the data it needs to examine could be simultaneously assessed and encoded instead of sequentially. It gave rise to an improvement of speed and efficiency, which is now a relatively high level, ideal for extensive applications.

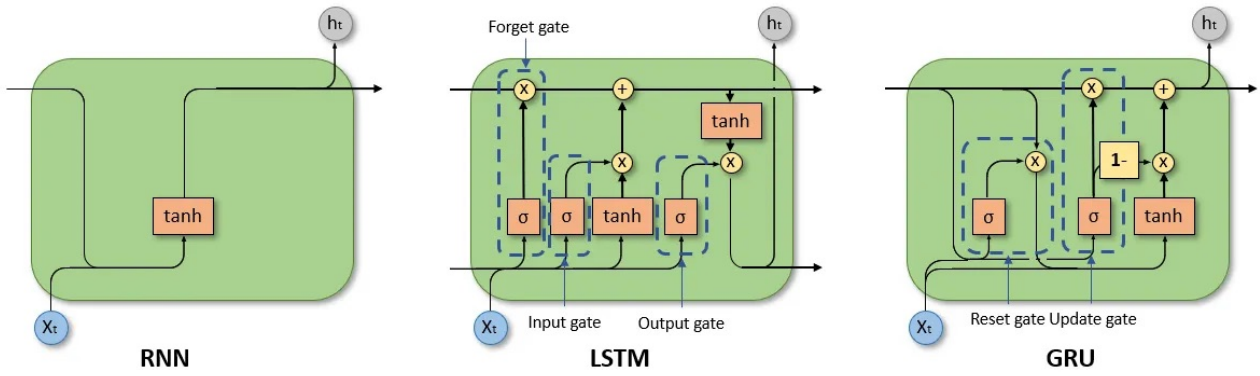


Figure 1: Architectural Comparison of Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM) models (Dancker, 2022)

The Transformer’s most unprecedented advantage is its scaling and generality. It is because the attention mechanism allows transmitting information between any two content elements, unlike RNNs and LSTMs, which by definition require looking back to understand what is next. To clarify, even the elements that used to be considered quite remote now have direct relationships with each other. Therefore, this fact accompanies the improved handling of long-range dependencies, which is remarkable because it is essential for understanding context and complex sentence structures. Another valuable property of the Transformer architecture is the possibility of its layer-wise design, which can also be scaled to be associated with building models that have a notably large number of such layers. In addition, unlike when constructing very deep RNNs or LSTMs, no such concerns with the dynamics of training dynamics complexity occur. These features largely determine the spread of models like BERT, GPT, and the use of Transformer architecture as the foundation of many NLP advances.

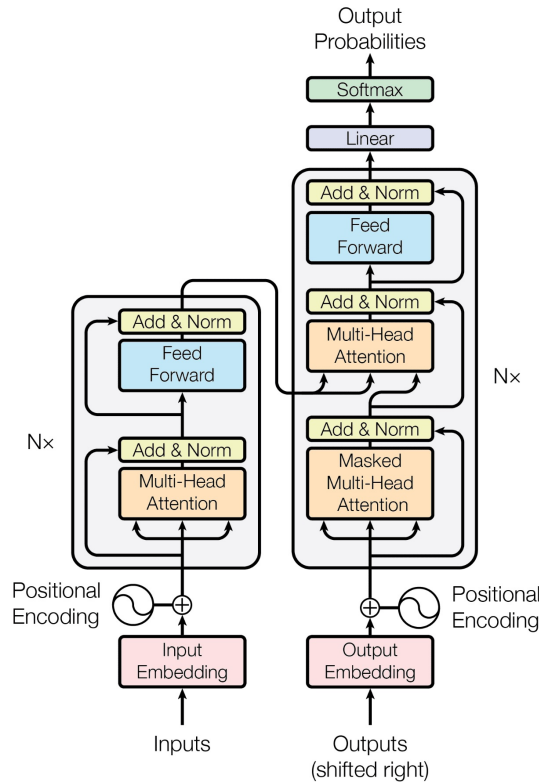


Figure 2: Transformer Model Architecture: Overview of the key components, including attention mechanisms, feed-forward layers, and positional encodings Vaswani et al. (2017).

The main bottleneck in training LSTMs is that, given the sequential processing nature, it processes only one piece of input data at a time in series. Here, each step will wait for the previous step’s computations to finish, resulting in a problem of sequential dependency that not only takes time but will also not allow the maximum utilization of modern hardware. GPUs are designed to run in parallel, yet, with RNNs having to wait for each step to complete, it will hinder the performance of such hardware. A different approach is used in Transformers through self-attention. What is fundamentally different, in this case, is that the model did not have to wait for the continuation of work at any point and all parts of the input were processed at the same time as well. Here, the model will be able to compute the output for each individual position in input data all at once, making it more suitable for parallel processing. Such a feature correlates well with the parallel computation functions of GPUs as well, which are capable of running thousands of computations at a time.

Therefore, as a matter of fact, running multiple samples of input data at the same time in parallel will greatly reduce the time required to train models. In the case of deep learning models, such a problem is even more accentuated due to the nature of having to process billions of parameters in multiple layers. With such a necessity, the double, or even more, change in training time through the use of parallel processing becomes vitally important in the context of neural networks that are very large and deep, which correlates well with modern NLP tasks:

$$\text{Output}_i = \text{LayerNorm}(x_i + \text{Sublayer}(x_i))$$

In the Transformer, every sublayer, every self-attention, and feed-forward network can be computed in parallel at each position i in the input sequence. This parallelization allows to quickly process any amount of data in parallel within a batch, and thus take full advantage of the processing power of GPUs. At the same time, it allows researchers to design more potent and processing-intensive models that would not be otherwise possible due to sequential data dependencies in the network architecture. Such models have advanced the ability to process and understand the complex and nuanced patterns of real-world languages, as can be seen in models such as GPT-3, which relies on a simplified autoregressive approach and hundreds of layers of non-recurrent neural networks to operate.

2.1.2 Current State of the Art in Generative Models

The current state-of-the-art in NLP is dominated by models based on the Transformer architecture, such as GPT-3 (Brown et al., 2020), BERT (Devlin et al., 2019), and T5 (Raffel et al., 2020). GPT-2 (Radford et al., 2018) and GPT-3, developed by OpenAI, are examples of large-scale generative models that have pushed the boundaries of text generation. GPT-3, with its 175 billion parameters, can generate impressively coherent and contextually relevant text on a wide range of topics and styles [5].

BERT stands for Bidirectional Encoder Representations from Transformers and was developed by Google. It is based on the transformer architecture and aims to generate a deep bidirectional language model by conditioning to the left and right context in all layers of the model. A significant advantage here is that pretrained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a variety of NLP tasks, such as question answering or language inference.

T5, or Text-to-Text Transfer Transformer is similar to GPT-3 and builds on the transformer architecture but presents a unique, text-to-text approach to NLP. Its basic idea is that all NLP tasks can be reframed into a single text-to-text format and understood as a “text-in, text-out” formulation. This makes the model easily transferable and enables deploying it directly on downstream tasks with just a single extra training run.

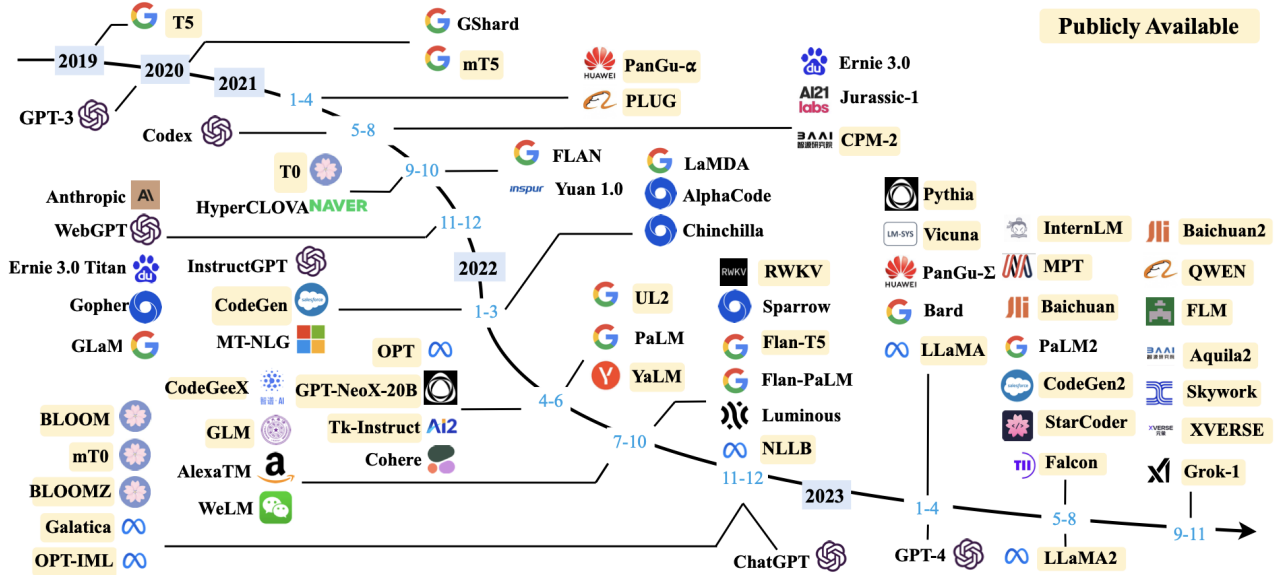


Figure 3: Timeline of Large Language Models Over 10B: This figure highlights key release dates and evaluates publicly accessible models, marked in yellow (Zhao et al., 2023)

2.2 Challenges of Low-Resource Languages

In the domain of natural language processing, the inequality of availability of resources between languages is stark. This chapter addresses the special needs of low-resource languages, giving an understanding of how these languages are classified and the hurdles on the way to creating NLP technologies for them. Besides, an overview of different approaches developed by researchers to confront the issue will be presented.

2.2.1 Definition of Low-Resource Languages

A low-resource language in the context of NLP is a notion that is typically used in accordance with the scarcity of digital texts, the limits of technological tools, and low interest on the part of the community of researchers. Thus, these languages are critically underrepresented in the global digital sphere in terms of the existence of a large portion of written literature, available and reliably annotated corpora for training NLP technologies, and a great deal of publications in computational linguistics. The classification is based on the volume of text-based data and the availability, as well as the absence of secure resources. Examples of low-resource languages are spread across the globe; they include indigenous languages in the Americas, as well as languages in highly polyglot countries or severely unused endemic dialects.

Table 1: Statistics of Wikipedia Articles by Language (Sizes in Parquet Files)

Language	Number of Articles	Size (GB)
English	6,816,956	11.36
Azerbaijani	199,588	0.22

2.2.2 Specific Challenges: Data Scarcity

The field of natural language processing (NLP) faces significant challenges when applied to low-resource languages, such as Azerbaijani. As Hedderich et al. (2021) highlight, the primary issue is the scarcity of large-scale, annotated datasets, which are essential for training modern

deep learning models. This lack of data hinders the development of robust and effective language models, as these models typically require extensive data to learn meaningful patterns and nuances of language. Additionally, low-resource languages often lack the technological infrastructure and tools, such as tokenizers and syntactic parsers, which further complicates the task of developing effective NLP applications.

Table 2: Document and token counts for selected languages from the CulturaX dataset, which includes data from mC4 and OSCAR corpora. This dataset encompasses 16TB of data in parquet format, expanding to 27TB when unpacked, and dedicates over half of its resources to non-English languages to support multilingual model training (Nguyen et al., 2023).

Language	# Documents	# Tokens	# Tokens (%)
English	3,241,065,682	2,846,970,578,793	45.13
Russian	799,310,908	737,201,800,363	11.69
Spanish	450,937,645	373,845,662,394	5.93
German	420,017,484	357,030,348,021	5.66
French	363,754,348	319,332,674,695	5.06
Azerbaijani	5,084,505	3,513,351,967	0.06

A paper that seeks to understand and help low resource languages’ specific challenges is the one by (Yang et al., 2023). It introduces BigTranslate that is built on top of the LLaMA framework. It is a model that aims at improving the support for over a 100 languages. Its main aim is to aid in supporting low resource languages and has helped improve on the abilities to carry out English-Azerbaijani and Azerbaijani-English directions of BigTranslate. Azerbaijani has about 0.1% of the data. The model struggles with extremely low-resource languages, particularly due to its data-induced set-back. Adequately low-data sets significantly affect the model’s abilities, thus requiring further investigations. As such, there is a need to learn mechanisms for training good language models. Therefore, it is essential to develop means of calibrating any training data for training large-scale, diverse, and multilingual models for low resource languages.

Another paper that helps in understanding the challenges in the field involved in training a language model for low resource languages is the one published by (Raffel et al., 2020). The paper points out the considerable challenges that are in pre-training language models for low resource languages. A pre-trained model mostly uses a large-scale multilingual dataset with different sizes for supporting different languages. In most cases, there is a high disparity in the given languages size. The imbalanced dataset sizes can have a negative impact on how well the model gets trained. The temperature-scaled mixing approach is key in helping choose a task and a dataset uniformly at random so that it can fit to decrease the chances of overfitting. The mix-up strategy is particularly key in low-data scenarios. The most important point learned from the paper is that the mixing and the mixed-up strategy help choose a task and a dataset uniformly at random. The models trained in such a way would be likely to perform adequately across its linguistic tasks.

A major challenge in pre-training language models for low-resource languages like Azerbaijani is the high rate of out-of-vocabulary (OOV) words. High-resource language models, when applied to low-resource contexts, often fail to recognize many local terms, particularly in specialized domains where specific jargon is prevalent. This results in significant performance degradation, as the model cannot effectively understand or generate language-specific or domain-specific content. Additionally, the sparse data available for these languages exacerbates the problem, making it difficult to train models that can accurately capture the nuances of the language (Zhang et al., 2020).

Regarding the integration of low-resource languages into large-scale language models, the principal challenge is the lack of high-quality, diverse data. Generally, low-resource languages experience a lack of digital data. Data is vital for training robust models for which the data is sufficient. Additionally, models trained on predominantly over-represented languages inherit and reify biases, perform worse when deployed under different models, and may reinforce not only the selection and treatment of the data but the culture it embodies. One relevant paper shows evidence of more than 4 times the data utility, the model for the considered language can achieve when if it gets a collection closer to the two main languages' sizes. Over 81% of the data available for training language models come from just 21% of the languages (Chi et al., 2024). This imbalance in data is evident in the poor performance of models on low-resource languages and the challenge for applicability of pre-trained models on them.

Specifically, we chose the paper to assess the challenge of using pre-trained models and tokenizers in low-resource languages. The challenge of scarce data, in this case, provides evidence by the fact that this sufficient data limit the applicability of tokenizers and models. Base and foundational tools fail or perform severely when tokenizing sentences for low resource, underrepresented languages because of their inability to understand and learn whose words and subwords. The main problem is that the foundation models are not exposed to data of such languages and can not distinguish patterns and words of those languages. These challenges show the risks of their application in different global contexts without inclusive data treatment.

2.2.3 Strategies for Overcoming Challenges

In the field of NLP for Azerbaijani, several fundamental works highlight the potential benefits of applying ML methods to the unique challenges of the language. For example, Aida-zade et al. (2016) conducted research on the use of Support Vector Machine in speech recognition systems by enhancing acoustic speech models. According to the results, using the MFCC and LPC features, the use of SVM with radial basis function and polynomial kernels can recognize Azerbaijani speech better than the traditional Multilayer Artificial Neural Networks. Concurrently, Hasanli and Rustamov (2019) copied a step-by-step guide to sentiment analysis of Azerbaijani-language tweets with the use of such algorithms as Linear Regression, Naïve Bayes, and SVM to determine the sentiment polarity. Furthermore, this research undeniably proves the adaptability to the Turkic languages of the methodology mentioned in Section 3. Finally, Rustamov and Mustafayev (2009) revealed that the challenges of Azerbaijani-specific ASR can be solved by offering a methodology that combines neural networks trained from different initial points. As a result of the conducted research, the reliability and quality of speech recognition have increased hence, the advantages in combining neural networks are enormous. To summarize, the information on the advancement of the NLP research for low-resource languages suggests examples of application and current problem points.

For the problem of low-resource languages, several valuable approaches have been proposed by academicians according to Hedderich et al. (2021). For instance, transfer learning can be applied through fine-tuning models pre-trained on high-resource languages on new low-resource data via sharing the common properties of a specific language. Data augmentation has been beneficial and involves increasing the size of a training data set through altering samples of data or constructing a new synthetic example. Distant supervision and cross-lingual learning are also valuable, as they can use external resources and parallel languages to generate annotations. These approaches can assist in mitigating the curse of scarce data for low-resource languages.

Marchisio et al. (2023) introduce the concept of "Mini-Model Adaptation," which offers a computationally efficient method for extending pre-trained masked language models (MLMs) to new languages, particularly those with limited resources. This approach involves creating a smaller, shallow mini-model from a fraction of a large model's parameters, allowing for the

rapid and efficient training of new language-specific embeddings. These embeddings can then be integrated back into the larger model, thus enhancing its cross-lingual transfer capabilities. The study demonstrates that this method not only preserves the performance of traditional full-model training but also significantly reduces the computational resources required, achieving comparable results with 1.6 to 2.3 times less compute on average across various NLP tasks such as XNLI, MLQA, and PAWS-X.

The research on the use of pre-trained multi-lingual language models conducted by Ogueji et al. (2021) highlights valuable implications for addressing the numerous challenges associated with low-resource languages and shows the numerous opportunities to create competitive NLP tools with limited data. Their research introduces a multilingual language model named “AfriBERTa”. The model is trained on text data belonging to 11 African languages only, and the volume of the data is less than 1 GB. As a result, the created language model can effectively compete with models such as mBERT and XLM-R on numerous NLP tasks. Therefore, creating linguistic inclusivity with such language models seems a way to go, and similar technologies can be employed when creating multi-lingual models with other language groups such as Turkic languages. This approach could offer a new possible direction for research since it becomes necessary to create appropriate technologies for a large number of underrepresented languages worldwide.

Expert-authored generative language models prepared by Micallef et al. (2022) for the Azerbaijani language demonstrate that the multiple challenges associated with low-resource languages can be addressed easily. First and foremost, the results of their research have proven that multilingual models such as mBERT have demonstrated substantial cross-lingual capacity and great potential for use in many low-resource languages. However, the exclusion of many underrepresented languages in the world from these multi-lingual models demonstrates that it becomes necessary to develop more language-specific models or larger multi-lingual models. Secondly, the results of the research state that pre-training a model using the training corpus of relevant size from various domains provides better results than solely relying on a general domain source, such as Wikipedia. Specifically, the research shows that the monolingual BERT model created from scratch with a new corpus generation for Maltese mostly outperforms the further pre-trained multi-lingual models and sometimes competes closely with them. I believe that similar results could be achieved with the Azerbaijani language thanks to the use of such an approach. Thirdly, the creation of the models’ vocabulary has proven to be a crucial aspect that aids in the improvement of the model’s performance. It becomes possible to include domain-specific terms among frequent OOV terms in the model’s vocabulary so that the model could have a better understanding of the domain’s language. It is also possible to employ the structure in domain-specific unlabeled data for the creation of synthetic tasks so that the model can be pre-trained on the data commonplace for the domain’s specific type. This condition can significantly lower the vocabulary gap for training and testing data and continuously widen for domain-specific data from the training corpus tested during the development of the model, just as it is stated by Zhang et al. (2020).

Scao et al. (2022) suggest a multifaceted approach. Engaging community involvement is crucial for accessing authentic and diverse language data. By integrating linguistic experts and native speakers in the data curation process, the BLOOM project ensured the quality and contextual relevance of the training data for low-resource languages. Technological innovations such as advanced tokenization and robust training algorithms were employed to capture the unique linguistic features of these languages more effectively. Moreover, ethical considerations were prioritized to ensure the data did not perpetuate harmful biases, aiming for fair representation across all languages included in the training corpus.

Chi et al. (2024) proposes cross-lingual knowledge transfer methods, which aim to utilize pre-trained models developed for English to enhance language processing capabilities in other

languages. This approach includes techniques such as parameter initialization and knowledge distillation, which help transfer general language understanding and task-specific knowledge from models trained on rich English datasets to models intended for other languages. Such methods are particularly beneficial in scenarios where annotated data or extensive corpora are not available in the target language. Additionally, a rebalanced sampling strategy during pre-training is suggested, which increases the model’s exposure to low-resource languages, thereby improving its performance across a broader linguistic spectrum.

2.2.4 Integration with Broader Research Goals

Based on the challenges and potential solutions or strategies related to improving NLP support for the Azerbaijani language as a low-resource language, several strategies have been identified. First, transfer learning is one of the most notable ones. It implies that the pre-existing NLP models that have been developed and trained using high-resource languages are adapted and fine-tuned using the available data for a low-resource language. This approach appears to be effective and has been already employed for various languages as numerous pre-trained models are already available. Fine-tuning refers to the process of adapting a pre-trained model to language structure or using a model trained for high-resource language to complete initial training using a smaller set of data available for a low-resource language. However, this strategy cannot be regarded as perfectly effective. Tokenizers used for the pre-trained models may not be effective with unusual syntax or phonetics of low-resource languages leading to poor solutions or incorrectly extracted tokens. At the same time, using data for high-resource languages to train models may lead to the reduced effectiveness of the models for low-resource languages and potential cultural inappropriateness for or errors in these languages. However, multilingual models can be a solution as an advanced strategy. It implies that a single model can be trained to support multiple languages. Another strategy is data augmentation, and, particularly, synthetic data generation. Approaches such as text simulation, back-translation, and using linguistic rules to create models appear to be useful. Crowdsourcing data from native speakers is another notable strategy, which can be employed using different platforms facilitating data transfer. Finally, policies and funding play an important role in the development of NLP technologies for low-resource languages. If the active development and promotion of digital technologies is encouraged as a government policy or supported by public or international funding, it can be beneficial for the development of the field. Thus, multiple strategies can be used to address related challenges and improve NLP technology for low-resource languages, including the Azerbaijani language. At the same time, these strategies are crucial to further developing NLP and digital technologies in general and making further changes.

2.3 Gap in Research for Azerbaijani Language Processing

Though language model technology has significantly advanced in recent years, there is still a considerable gap in research and resources for many languages, such as Azerbaijani. Azerbaijani, spoken by millions of people, is largely underrepresented in the training data of modern pre-trained language models. Most models’ training data contain minuscule amounts of Azerbaijani language content, if any at all, making the models nearly incapable of understanding or generating Azerbaijani text. Further, the relative non-existence of Azerbaijani in academic and commercial NLP research only compound this issue. Practically no (large) language models are publicly available that leverage Azerbaijani datasets. This lack of resources has also stymied the implementation of Azerbaijani-focused NLP applications, such as digital assistants that speak Azerbaijani, writing assistants that are well-versed in Azerbaijani grammar and Lexicon, and more advanced translation services to and from Azerbaijani. As a result, advancements in the field of NLP are heavily skewed towards high-resource languages, and the vast majority

of the Azerbaijani-speaking population remains without access to the modern technologies of NLP. The purpose of this research is to address this disparity by introducing a large, state-of-the-art model exclusively trained on a diverse Azerbaijani corpus, pushing the capabilities of low-resource language processing to their limits.

3 Dataset Acquisition and Curation

3.1 Source of Data

Data quality and diversity are instrumental in artificial intelligence, and in particular for generative language models. This is especially true for underrepresented languages like Azerbaijani, where lack of data hinders technological progress. Since the paper is devoted to the specification of data sources used for pretraining a generative language model for the Azerbaijani language, it must rely on certain criteria of selection. The latter should be related to the volume, diversity, relevance, and representativeness of the text from real-world language.

The sources included OSCAR, mC4, Wikipedia, and some books and news datasets; they were selected due to their complimentary nature and unique linguistic corpora.

OSCAR, as described by Julien Abadji et al. in their paper “Towards a Cleaner Document-Oriented Multilingual Crawled Corpus,” is the resource used to meet the challenge of using web-crawled data to train large generative language models. In the paper OSCAR and its pipeline Ungoliant are considerably enhanced by moving from the language classification of lines to that of each document. This allows multilingual whole documents to be preserved so that they can be read and used for natural language processing, and a new method of language identification and automatic annotation performed after the classification permits better querying of the corpus and its use based on the quality and length of each document. This makes OSCAR a significant, structured resource for linguistic research, specifically, for the training of efficient language models. Having used the Azerbaijani subset of OSCAR, we have approximately 1.1 million documents of text which corresponds to 3GB of information. It basically covers everyday topics along with everyday non-standard Azerbaijani used in text generation.

Another heavily multilingual and vast source data used for training of the Azerbaijani generative language model is the mC4 subset coming from Common Crawl. It includes texts of over 100 languages, which was the principal reason behind selecting it along with its volume which is useful for training. The size of mC4 is approximately 1.5 million of documents. This type of data is beneficial for training generative language models and their efficiency as it is derived from various types of sources from websites to forums and blogs, each with differing styles of text and different levels of language.

Building upon the existing individual datasets, the CulturaX dataset developed by Nguyen et al. (2023) is a powerful addition, providing significant value for contributors and making a substantial change in the landscape of large language models. The CulturaX dataset is the largest such dataset, containing 6.3 trillion tokens in 167 languages, and implementing a cleaning process that corrects language identification errors and removes undesired content and outlier documents. In the process of deduplication, the authors used MinHashLSH and a method of exact URL match to ensure the removal of exceptionally similar texts. This process allowed the data preparation to address one of the main identified drawbacks of previous datasets—an excess of clutter and redundancy that hampers the model’s ability to learn generalizable features of the language. The prepared CulturaX dataset is versatile, offering a broad range of data in a large variety of languages, and will undoubtedly help train large language models to better understand and generate the language. The model is completely open-source and is available on the HuggingFace platform—its openness further helps create a culture of sharing in AI and NLP. For researchers working in some of the less represented languages, the presence of the CulturaX dataset can help level the playing field, making AI and NLP research more accessible and equitable.

Talking about the quality of datasets, it is essential to note that Wikipedia is a well-structured resource, which is highly edited, and its content is vastly correct and authoritative. The Azerbaijani Wikipedia dataset, although small at 150 thousand of text data, is also ben-

essential for the model to be able to learn the formal tone of writing, some technical vocabulary, as well as correct handling of information. It is in sharp contrast with the colloquialism usually provided by informal web crawlings. These web crawlings constitute the second crucial, albeit smaller, source of data. Finally, the books and news datasets were also handpicked to ensure that they contain the literary and journalistic styles of the given language. The books dataset spans a variety of genres, teaching the model to write creatively in many different ways, while the news dataset offers a more varied selection of current affairs, reports, and opinion pieces. The need to include such a large variety of applications of the Azerbaijani language is paramount for teaching. The models need to be exposed to all applications, from colloquial and informal dialogue to journalistic writing and formalities.

3.2 Dataset Curation: Cleaning and Filtration Techniques

In the development of generative language models, data curation and quality assurance are prerequisite. Such a process will have a significant impact on the performance as well as the efficiency and effectiveness of resulting models. Data selection which is optimal for models is tightly linked to the probabilistic distributions in which they are expected to perform Albalak et al. (2024). This can enhance model prediction accuracy and generalizability. This perspective also reveals the necessity of selecting data not only to improve model performance but also for operational efficiency: reducing dataset size and costs (Schreiber et al. (2020); Carlini et al. (2020)). In order to address this, we must honestly assess the integrity of all evaluation standards and strategies for mitigating unwanted model behaviour—such as harmful biases and toxic behaviors that could harm humans when applied in real life situations. This calls for a careful selection of data for use in detailed data management (Dodge et al. (2021); Welbl et al. (2021)). As indicated by recent studies, the process of data curation includes discarding duplicate or low-quality data, a step that is essential for building robust language models CONNEAU and Lample (2019); Rae et al. (2021)). In constructing the Yi model, Young and colleagues (2024) focus on screening a large-scale dataset for its quality and then feeding this into a densely engineered transformer architecture. Underlining this is their perception of big data requirements in order to maximize the potential of standard architectures, make them fully functional (as signified by Yi-34B matching GPT-3.5 in performance) and also put them within reach for deployment on consumer electronics through techniques such as 4/8 bit quantization. In this way, hard data of such quality is indispensable to the performance of AI models and in order to make it applicable on consumer normal equipment this level is needed. This comprehensive approach to data curation and model training illustrates the multifaceted benefits of careful data selection – from better model performance and reduced operational costs to ethical AI model training. It is a fundamental element in building generative language models, particularly for languages like Azerbaijani where high-quality, culturally appropriate data is the key to developing effective AI tools. Given all this research demonstrates the importance of high-quality data, a major process was put in place to ensure both cleanliness and suitability of the data: modifying documents for standardisation and clean-up of the content, plus adding a number filters based on various text characteristics to further refine the dataset.

In preparing documents for language model training, detailed modifications and filtering steps are crucial to ensure the data’s quality and relevance. The approach is structured into two phases: document modification and document filtering, each involving specific methods to refine and optimize the text.

3.2.1 Document Modification

The first step in document preparation is initial phase, which involves three key parts to standardize text structure. It is also a great nuisance of the White Space Standard, where non-

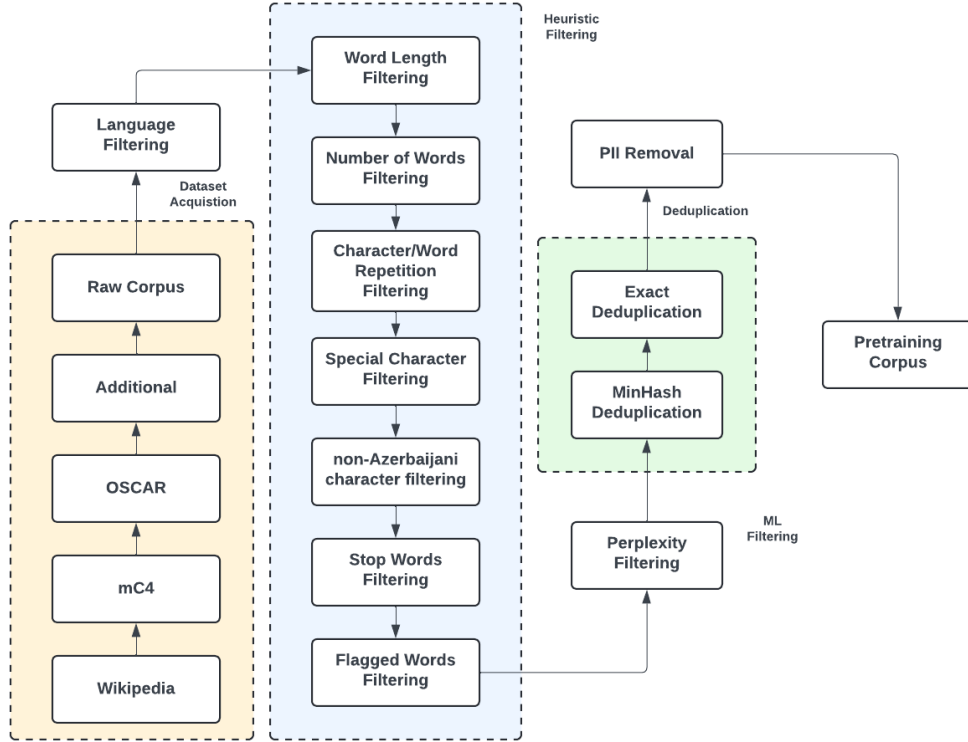


Figure 4: Filtering pipeline for AzLlama training corpus of 3B tokens.

typical spaces – peculiar IBM compatible word processors’ spaces, for instance—are changed to a normal space (‘ ’). Doing this standardization enables us to go on and segment the text more neatly, thereby making it easily interpretable part by part. After that it is time to implement the Long Word Removal, aimed at words of more than an operator-defined length. Shorthand long words need to go out. This step can exclude from processing potentially corrupt data or unrelated texts simply because they are URLs or codes. Post-stripping, by splitting the document into words, stripping them of around punctuation marks, and then getting rid of those which are too long. The third step, Removal of Words with Incorrect Substrings, means removing words containing strings of characters that have been defined in advance as wrong (for example, "http", "www", ".com", "of", "//"). This method aims for links and Web-related text in particular—that could unduly twist the training of a language model away from normal language use.

3.2.2 Heuristic-based Document Filtering

The second phase, document filtering, employs both heuristic-based methods and machine learning techniques to further refine document selection. Initially, heuristic filters are applied: Filtering Based on Word and Character Counts excludes documents whose word count falls outside predefined acceptable ranges, aiming to exclude texts that are too short to provide adequate context or too lengthy, potentially covering multiple themes or subjects. The Character Repetition Ratio filter examines the frequency of character n-grams, removing those with a high ratio of repeated n-grams to total n-grams, with a normalization function (square root) applied to mitigate the influence of document length on this metric. Similarly, the Word Repetition Ratio focuses on word n-grams, discarding documents with excessive repetitions indicative of poor quality or spam-like content. Special Characters Ratio targets documents exhibiting an unusually high proportion of special characters, including punctuation and emojis, which are often not conducive to coherent text processing. The Stop Words Ratio, using a list of

165 Azerbaijani-specific stop words, evaluates documents based on their stop word proportion, filtering out texts with either excessively high or low ratios. Likewise, the Flagged Words Ratio employs a list of 115 words relevant to Azerbaijani to exclude objectionable or irrelevant content.

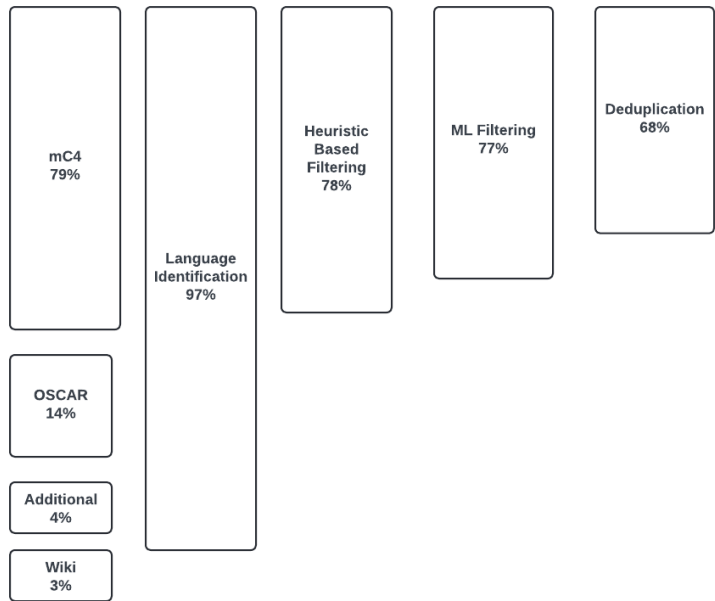


Figure 5: Stages for each filtering pipeline. The diagram outlines a multi-stage data processing workflow with percentages indicating the proportion of data remaining after each step. Starting with 'mC4' at 79%, it proceeds to 'Language Identification' at 97%, followed by 'Heuristic Based Filtering' at 78%, 'ML Filtering' at 77%, and concludes with 'Deduplication' at 68%. Supplementary sources of data include 'OSCAR' at 14%, 'Additional' at 4%, and 'Wikipedia' at 3%.

3.2.3 Machine Learning Document Filtering

In addition to such heuristic methods, more sophisticated machine learning techniques were used throughout the refinement process of the dataset. As an advanced filtering method, a machine learning heuristic checks the documents' Language Identification Prediction Score. It uses fastText-based model (Joulin et al. (2017)). The low-prediction-confidence documents are excluded from the dataset by assessing the texts for their internal language consistency and filtering out documents with low-confidence in being of Azerbaijani origin. Moreover, the Perplexity Score has been calculated for determining the KenLM – based metric (Heafield (2011)) of text complexity, measured with trigrams. This model was implemented on Azerbaijani Wikipedia, as no other models were available for Azerbaijani, significantly enhancing language model prescreening: the documents with low perplexity have been filtered out, as disjointed or exceptional text patterns are introducing a bias to the training of the language model. The introduction of the KenLM model for Azerbaijani is a significant advance for the field, as the naturalness and fluency of the text is finally a quantitative measure, lacking before. This method allowed a better deduction of the text complexity and coherence and can provide better parameters for filtering the data. As Azerbaijani continues to evolve in an online manner, the filtering threshold and score can be readjusted. The extension of the KenLM above the Azerbaijani Wikipedia could increase the accuracy of a more specific and more official Azerbaijani model. Complementary heuristic and machine-learning methods offer a complete toolkit for structuring the dataset for processing a language model. This integration also provides a

way of continuous adaptation and refinement as more data sources are added or the Azerbaijani language undergoes more changes in digital context. Duplicates Subtask. Deduplication is an integral component of the training data set preprocessing, and it must be completed to fulfill translation models' value and uniqueness. MinHash Deduplication and Exact Deduplication are the two deduplication strategies used to avoid overrepresentation in the data set by minimizing the number of duplicates.

3.2.4 Deduplication

MinHash deduplication is a probabilistic technique that was developed to ascertain the presence of near-duplicate documents quickly and efficiently remove them. MinHash was developed by Broder (1997) and is based on the theory behind locality-sensitive hashing. The concept of MinHash is straightforward: one creates a signature of a document in a highly compact form. Consequently, the comparison is conducted using a small amount of representative data; for this particular reason, the method is called MinHash. Two such small signatures, naturally, can be compared much faster. Therefore, by comparing MinHash signatures of documents, much quicker and scalable comparison can be done for the entirety of the corpus. Any documents shown to have similar MinHashings are near-duplicates, and they are subsequently removed from the data corpus. MinHash deduplication is ideally suited for calculating the duplication distance for individual pairs of large volumes, as it amounts to traversing all possible pairs explicitly. Thus, it is a powerful tool for cases where exactly comparing all pairs of texts infeasible due to computational limitations. After the first pass with MinHash deduplication, documents are subject to Exact deduplication, ensuring that all duplicates are erased definitively. After MinHash, Exact deduplication forms a binarized matrix. In this situation, the binary classification is simple: either two elements are fully duplicates, or they are not. Unlike MinHash, in the Exact deduplication case, no hashing occurs, and, consequently, the checks are fully assured. Only documents with the verbatim overlap removed in this step, ensuring the retention of only unique entries. Thus, if just one duplicate remains after the MinHash procedure, it will be removed in this step.

3.2.5 Personally Identifiable Information Removal

In the processing of emails and other text-based communications for training language models, identifying and filtering out Personally Identifiable Information (PII) is crucial to protect privacy and comply with data protection regulations. Using tools that combine regular expression-based identification with context-sensitive analysis, enhances the detection of PII. This approach employs a sophisticated rule-based system tailored to recognize various types of PII, including names, email addresses, phone numbers, and other identifiers. By integrating regular expressions with natural language processing techniques, this method effectively pinpoints and removes PII from large datasets, ensuring that the training material used for language models is both safe and compliant with legal standards. This level of filtration is important, especially when handling data with potential privacy implications like emails, which often contain sensitive personal information.

3.3 Challenges and Solutions in Data Curation

In the context of data curation for language models, quality filtering heuristics and machine learning (ML) techniques each present their own sets of advantages and drawbacks.

Quality filtering heuristics have an advantage of being controlled and robust, relying on relatively clear prior assumptions regarding what constitutes a valid document. For instance, simple rules concerning word or character counts can lead to the excluding of the systematic

majority of texts failing to meet certain criteria. Therefore, it is simple to ensure consistency in the developed data set. Nonetheless, such methods also have severe limitations, as they are based solely on the surface characteristics of documents. They may discard more useful data than is necessary, as they do not account for the specifics of texts that could make their repetition a meaningful pattern. Furthermore, hyperparameter tuning of such heuristics is demanding, as it often becomes challenging to strike a balance between including and discarding too much content. For example, high character repetition ratio filter required refinement – the method was not precise, as it disproportionately targeted short texts or misidentified long documents as repetitive due to their length. A square root normalization function was introduced in this paper to reduce the influence of document length. However, the ‘n’ parameter in n-gram analysis had to be properly selected, as small ‘n’s would excessively penalize the longer texts, especially in ideogram languages like Chinese. This adjustment was essential to ensure the selected data was refined while retaining sufficient content to train the language model.

On the other hand, ML filtering techniques allow for more nuanced quality and content-based filtering, which can more effectively target documents that are truly relevant and of high quality. These methods are less likely to strictly adhere to superficial characteristics and can adapt to complex patterns in data. However, they come with their challenges, particularly in estimating the impact of training documents on the final model. ML methods may also introduce unwanted biases, especially if the training data itself is not representative of broader language use.

Addressing domain-specific requirements adds another layer of complexity. It is important to inspect the effect of filtering on domain-specific data. For instance, extracting and manually inspecting a sample of ten documents per domain, such as from top URLs, can provide insights into how well the filtering process preserves important domain-specific information. Crafting domain-specific filters and hyper-parameters then becomes crucial to ensure that the curated data accurately reflects the variety and nuances of each domain. This approach must also be adapted for handling multiple languages, which may require different strategies based on linguistic and cultural characteristics.

Finally, the approach to choice between deterministic and stochastic selection methods determines the nature of data points and curation flexibility. While hard thresholds work as powerful decision points precipitating the choice for filtering, they could also exclude borderline cases. Simultaneously, stochastic smoothing of rules could create a softer approach to decision-making, with a number of documents not being exactly in or out but having a certain probability of getting in based on their features. Notably, this can reduce certain rigidity of thresholds recruitment and potentially contribute to more nuanced data.

4 Model Development

4.1 Overview of Model Architecture

The LLaMA-2 model architecture is an intricate upgrade over the previous model, built to support extensive data processing capabilities, and optimized for faster inference – an invaluable feature, particularly for dialogue systems. This is heavily based on the transformer architecture, which introduced an attention mechanism for more effective handling of sequential data. Attention mechanisms have radically transformed natural language processing by allowing the model to dynamically focus on different parts of the input data according to the relevance for a given token. The central component of the LLaMA-2 model is the embedding layer, which transforms input non-semantic “tokens” into high-dimensional vectors. These vectors, between 512 and 1024 dimensions depending on the scale of the model starting from an initial random initialization, converge during training to encode the word’s semantic meaning. Mathematically, it can be defined as $\mathbf{E} = \mathbf{W}_e \mathbf{x}$, where W_e is the embedding matrix, and x represents the input tokens. Additionally, the model includes several layers, each consisting of multi-head attention and feed-forward neural networks. Multi-head attention is necessitated by the model’s ability to attend to multiple parts of the sequence simultaneously, improving its ability to capture complex dependencies. It can be defined as follows:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}^O$$

with each individual head computed by:

$$\text{head}_i = \text{Attention}(\mathbf{Q} \mathbf{W}_i^Q, \mathbf{K} \mathbf{W}_i^K, \mathbf{V} \mathbf{W}_i^V)$$

where \mathbf{W}_i^Q , \mathbf{W}_i^K , and \mathbf{W}_i^V are the parameter matrices for the queries, keys, and values, respectively, in each head, and \mathbf{W}^O is the output transformation matrix.

The attention within these heads operates through the scaled dot-product attention mechanism, where the inputs are queries and keys of dimension d_k , and values of dimension d_v , resulting in an output that is a weighted sum of the values, weighted according to the dot product of the query with all keys:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q} \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}$$

This expression uses the softmax function scaled by $\sqrt{d_k}$, which helps stabilize the gradients during training.

A novel modification in LLaMA-2 is the integration of Rotary Positional Embedding (RoPE) (Su et al., 2023) which incorporates positional information directly into the attention calculation, contrasting with traditional methods that add positional vectors to the embeddings. This integration is particularly effective in tasks that heavily rely on the positional context, such as interpreting long documents or dialogues.

The feed-forward networks within each transformer layer of LLaMA-2 perform a critical function by applying a nonlinear transformation at each position independently but identically:

$$\text{FFN}(x) = \max(0, x \mathbf{W}_1 + b_1) \mathbf{W}_2 + b_2$$

Here, \mathbf{W}_1 , \mathbf{W}_2 , b_1 , and b_2 are network parameters. This model advances the standard by incorporating SwiGLU, an activation function that refines the gating mechanism within the feed-forward layers, facilitating more selective information processing which enhances pattern recognition capabilities in complex datasets. Second, by utilizing FlashAttention, LLaMA-2 is

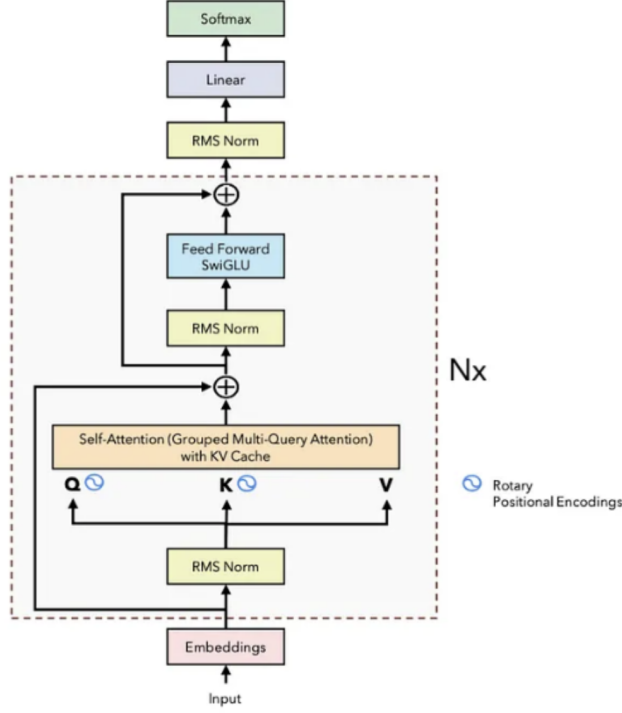


Figure 6: LLaMA Architecture Ibe (2024)

able to optimize the computation of attention more remarkably by reducing memory consumption and enables the system to process sequences of large sizes. The approach is imperative, especially when working with prolonged sequences of up to 4,000 tokens and when accelerated response is a critical factor in real-time applications. FlashAttention works by improving the standard attention computation approach through better organization of memory and computation, thereby ensuring that the model operates faster and more efficiently. The new model builds on the previous model’s capabilities through various architectural changes and takes advantage of improved embedding techniques, better attention approaches using RoPE, and customized feed-forward networks with SwiGLU. By incorporating FlashAttention (Dao et al., 2022), the transformer architecture progressively evolves and adapts into a significant step in the future of natural language processing systems.

One difference in the LLaMA model from the original Transformer model is the use of RMSNorm instead of the typical layer normalization. RMSNorm is a different layer normalization variant used in normalizing the activations of the previous layer by the root mean square of the layer inputs. Layer normalization is vital in ensuring that the learning process is done stably by normalizing the input of the layer to have a mean of zero and standard deviation of one. RMSNorm is effective and computationally less expensive since it does not compute across all features for normalization. Moreover, the LLaMA models use the pre-normalization variant of RMSNorm. In the pre-normalization, the normalization is done before the major layer in a transformer block other than after. The method provides more stable activation distribution with the waveform of the network during training. It plays an essential role in the LLaMA deep learning model in preventing the vanishing and exploding gradient since they are difficulties in training deep networks.

Regarding activation functions, LLaMA models utilize the SwiGLU activation function instead of the widely used ReLU (Rectified Linear Unit). SwiGLU, short for Sigmoid-Weighted Linear Unit, is a variant of the Gated Linear Unit (GLU) that uses the sigmoid function, σ , as a gating mechanism. The formula for SwiGLU in LLaMA can be described as follows:

$$\text{SwiGLU}(x) = x \odot \sigma(\mathbf{W}_g x + \mathbf{b}_g)$$

where x is the input to the layer, \mathbf{W} and \mathbf{W}_g are weight matrices, and are bias vectors. The sigmoid function acts as a gate to control the flow of information, allowing the model to learn complex patterns more effectively. This function generally leads to better performance in terms of learning complex dependencies compared to ReLU.

In positional encoding, LLaMA abandons traditional absolute or relative positional embeddings in favor of Rotary Positional Embedding (RoPE). RoPE encodes the position information directly into the attention mechanism, which helps in maintaining rotational equivariance. This means that the positional information adjusts the attention scores based on relative positions rather than absolute ones, thereby improving the model’s ability to understand and generate contextually relevant text sequences over long distances.

One significant innovation in LLaMA-2 is the use of grouped query attention (GQA) instead of traditional multi-head attention. GQA groups the queries into fewer, larger groups, which reduces the computational burden typically associated with handling each query individually in multi-head attention.

It is introduced by Ainslie et al. (2023) as a middle-ground between Multi-Head and Multi-Query attention variants. GQA addresses the memory access bottleneck, introduced by the KVC, by grouping keys and values across attention heads, thus reducing the overall memory footprint of KVC-enabled MHA. By aggregating multiple queries before the computation of attention, GQA significantly accelerates inference times and curtails the cache size needed for operations, thereby optimizing both computational speed and memory utilization. This method speeds up inference significantly while still capturing essential information from the input data.

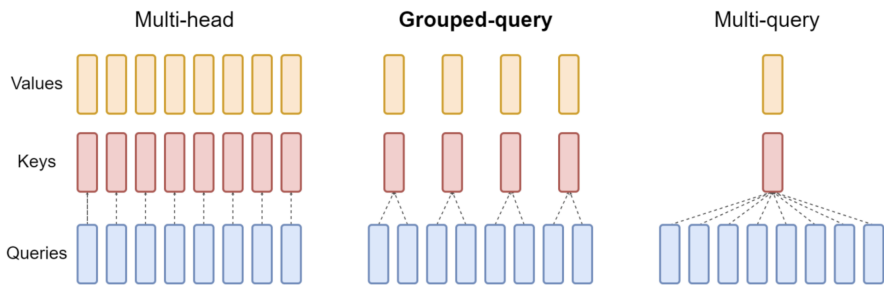


Figure 7: Comparison of Multi-Head, Multi-Query, and Grouped Query Attention (GQA) mechanisms Ainslie et al. (2023).

Additionally, LLaMA models employ Key-Value (KV) caching to accelerate the inference process. KV caching stores the results of previous computations so that the model can reuse them without recalculating the attention scores for repeated queries. This technique is particularly useful in scenarios where inputs have repetitive patterns or sequences, leading to faster response times and reduced computational load.

In the pre-training of AzLlama, a generative language model for Azerbaijani, optimization involves minimizing a sophisticated loss function under constraints that promote generalization and stability. The primary loss function utilized is the cross-entropy loss, which for a sequence of words can be extended as:

$$L = -\frac{1}{N} \sum_{t=1}^N \log p(w_t | w_1, \dots, w_{t-1}; \theta)$$

Here, N is the length of the word sequence, w_t represents the actual word at position t , and $p(w_t | w_1, \dots, w_{t-1}; \theta)$ is the probability of w_t predicted by the model given the preceding words and model parameters θ . Optimization is performed using Adam, an algorithm that computes adaptive learning rates for each parameter. Adam updates the parameters as follows:

$$\theta_{\text{new}} = \theta_{\text{old}} - \frac{\eta}{\sqrt{\hat{v}} + \epsilon} \hat{m}$$

where η is the step size, \hat{m} and \hat{v} are estimates of the first and second moments of the gradients, respectively, and ϵ is a small scalar added to improve numerical stability. Furthermore, regularization techniques such as L2 regularization and dropout are applied to prevent overfitting and to improve the model’s generalization capabilities. L2 regularization adds a penalty term to the loss function:

$$L_{\text{reg}} = L + \lambda \|\theta\|^2$$

where λ is the regularization strength. Dropout randomly sets a fraction of the input units to zero at each update during training, which helps make the model robust to slight variations in input data. These strategies collectively enhance the model’s ability to generalize across diverse linguistic contexts in Azerbaijani, aiming to produce text that is not only coherent but also rich in contextual appropriateness. This comprehensive optimization approach ensures that AzLlama achieves a high level of predictive accuracy and maintains stability during training.

Algorithm 2 Pseudocode for Training a Decoder-Only Transformer Model: This pseudocode outlines the training process of a decoder-only Transformer model, starting with the initialization of model parameters, the application of Transformer layers including self-attention and feed-forward networks, and the steps involved in the forward pass, loss calculation, and parameter updates during training. It highlights key operations such as embedding, positional encoding, RMS normalization, and the use of attention masks to manage sequence information effectively. Finally, periodic validation checks are performed to monitor model performance and save the best model based on validation loss.

```

1: Input: Clean Data, Hyperparameters
2: Output: Trained Model
3:
4: initialize model parameters (include embeddings, attention weights, feed-forward network
   parameters)
5: for each epoch in epochs do
6:   shuffle Clean Data
7:   for each batch in Clean Data do
8:      $X, Y \leftarrow$  batch input, batch labels
9:      $X_{\text{mask}} \leftarrow$  create attention masks for  $X$ 
10:     $X_{\text{embedded}} \leftarrow$  token embedding( $X$ ) + positional encoding( $X$ )
11:     $X_{\text{transformed}} \leftarrow X_{\text{embedded}}$ 
12:    for each layer in Transformer Layers do
13:       $A_{\text{output}} \leftarrow$  scaled dot-product attention( $X_{\text{transformed}}, X_{\text{transformed}}, X_{\text{transformed}}, X_{\text{mask}}$ )
14:       $X_{\text{transformed}} \leftarrow$  RMS Normalization( $X_{\text{transformed}} + A_{\text{output}}$ )
15:       $F_{\text{output}} \leftarrow$  feed-forward network( $X_{\text{transformed}}$ )
16:       $X_{\text{transformed}} \leftarrow$  RMS Normalization( $X_{\text{transformed}} + F_{\text{output}}$ )
17:    end for
18:    predictions  $\leftarrow$  linear output layer( $X_{\text{transformed}}$ )
19:    loss  $\leftarrow$  cross-entropy loss(predictions,  $Y$ )
20:    gradients  $\leftarrow$  compute gradients(loss)
21:    update model parameters(gradients, learning rate)
22:  end for
23:  if epoch % validation frequency == 0 then
24:    validation loss  $\leftarrow$  evaluate model(Validation Data)
25:    print "Epoch", epoch, "Validation Loss:", validation loss
26:    if validation loss < best loss then
27:      save model parameters as best model
28:    end if
29:  end if
30: end for
31: return best model

```

Table 3: AzLlama-150M Model Architecture Parameters: Hidden Size refers to the dimensionality of the hidden layers in the model, affecting the capacity to learn complex patterns. Intermediate Size specifies the size of the feed-forward layer within each transformer block, which is crucial for processing information between attention layers. Max Position Embeddings defines the maximum number of tokens the model can process in a single sequence, determining the scope of context the model can consider. The number of Attention Heads indicates how many parts the model divides its attention mechanism into, allowing it to simultaneously attend to different parts of the input sequence from different representational perspectives. The number of Hidden Layers represents the depth of the model, where each layer can learn a different level of abstraction. The number of Key-Value Heads specifies how many sets of key and value pairs are used in the attention mechanism, which can affect the model’s ability to differentiate and prioritize different aspects of the input data.

Parameter	Value
Hidden Size	768
Intermediate Size	3072
Max Position Embeddings	512
Number of Attention Heads	12
Number of Hidden Layers	12
Number of Key Value Heads	4

The architecture of AzLlama, which contains 150 million parameters, is a focal point of our study. This model size was selected to balance computational efficiency with the capability to capture complex language patterns effectively. Models with a large number of parameters are renowned for their deep learning capabilities, enabling them to model the complexities of language with higher fidelity. However, this comes at the cost of increased computational demand for both the training and inference phases.

The choice of 150 million parameters for AzLlama necessitates substantial computational resources, particularly in terms of processing power and memory. This scale is also crucial for determining the data requirements for effective training. According to the Chinchilla study (Hoffmann et al., 2022), the optimal data-to-parameter ratio is approximately 4:1. Given that we used 9 billion tokens for training AzLlama, our dataset exceeds this ratio, suggesting an ample data volume relative to our model’s size under Chinchilla’s guidelines. However, a contrasting perspective is presented in Hu et al. (2024) findings, where an optimal data-to-parameter ratio of 200:1 is recommended for maximizing model performance. By this metric, our 9 billion tokens fall short, indicating the potential for significant performance gains through further data scaling.

This discussion is underscored by the recent training endeavors of Meta’s LLaMA 3 model (considered to be GPT4 level model on benchmarks), which utilized 15 trillion tokens. Despite this extensive dataset, the loss continued to decrease logarithmically, highlighting that even massive datasets might not saturate the learning capabilities of highly parameterized models. This evidence strongly supports the hypothesis posited by Hu et al. (2024) and suggests a revisit of our data acquisition strategy to fully leverage AzLlama’s learning potential.

4.2 Pre-training the Language Model

4.2.1 Training Data Preparation

In the context of natural language processing, the creation of training data is the first preparatory measure that shapes the performance of models such as LLaMA-2. Tokenization is one of the elements of training data preparation as it modifies raw text into a form that allows the

model to process it with ease. This is especially important when the language model is trained from scratch, as is the case with the Azerbaijani language tokenizer based on LLaMA-2.

The decision to develop a tokenizer from the start rather than utilizing a pre-trained model is fueled by the uniqueness and complexity of the Azerbaijani language. Pre-existing tokenizers, including one of the previous LLaMA models, are ineffective in processing Azerbaijani text owing to the limited frequency of this language in their training data. Suboptimal tokenization results from this, as the model often sub-divides Azerbaijani words into fewer meaningful fragments due to its inability to recognize particular linguistic processes unique to Azerbaijani. This gives rise to a fragmentation of text that decreases the model’s capability of understanding and generation as it destroys the language context conveyed by longer tokens. Thus, a SentencePiece BPE tokenizer based on Kudo and Richardson (2018) is trained on Azerbaijani using a large corpus of approximately 2 million cleaned and filtered Azerbaijani documents. SentencePiece is a popular choice for tokenizer training because it is language-independent and performs well with languages that do not have clear word boundaries.

SentencePiece BPE is particularly useful because it does not require any preprocessing or text separation. BPE works by combining the most frequent byte pairs in the corpus supplied, treating the text as a sequence of raw byte strings or characters, and learning this over time. It learns to combine pairs of bytes into tokens in order to assign meanings to strings of characters. SentencePiece BPE is more advantageous than other BPE implementations because BPE typically relies on tokenized input or preprocessing, while SentencePiece BPE does not. The BPE algorithm Union these most frequent pairs of text bytes at each stage. Initially, the text is treated as a sequence of raw byte pairs, which it then learns to combine into more complex byte strings. The advantage of this approach is that it is data-driven and allows the model to mold itself to the properties of the text. Training the tokenizer on Azerbaijani using SentencePiece BPE from scratch ensures a fine-tuned model for this language. Initially, the vocabulary consists of individual characters or, specifically, byte pair sequences that form a part of the training data. Therefore, the vocabulary V can be set to the byte or character set C of the corpus.

$$V = C$$

In each iteration of the training process, SentencePiece BPE looks for the most frequent pair of adjacent tokens (x, y) in the corpus and merges them to form a new token z . This merging process is repeated iteratively and can be represented by the formula:

$$V = V \cup \{z\}$$

$$z = x \cdot y$$

where x and y are the tokens being merged. The selection of pairs (x, y) is based on their frequency of co-occurrence in the dataset, and this frequency can be mathematically represented as:

$$\text{freq}(x, y) = \sum_{\text{contexts}} \mathbf{1}_{(x,y)}(\text{context})$$

where $\mathbf{1}_{(x,y)}(\text{context})$ is an indicator function that counts the occurrence of the pair (x, y) in each context within the corpus.

In this way, with each merge the vocabulary grows slightly, and the model effectively coarse-grains itself from a fine-grained character-level representation to a more semantically charged abstract representation of frequent subwords or sequences of characters. The total vocabulary size (or the number of merge operations) is a hyperparameter that can be tuned based on the complexity of the task and on the unique properties of the target language. The goal of the tokenizer is to segment any given input into a stream of tokens using the learned vocabulary. The segmentation of an input string s to a sequence of tokens T is defined as follows:

$$T = \operatorname{argmin}_T \left(\sum_{t \in T} -\log P(t) \right)$$

where $P(t)$ is the probability of token t in the vocabulary V , which is indirectly related to the frequency of token t during the training phase. Besides its model though, SentencePiece BPE also differs significantly from more traditional methods of tokenization, such as classic BPE, in that it does not rely on any language-specific preprocessing of the input text – a fact that makes it especially useful in richly morphological languages such as Azerbaijani, where we would otherwise have to deal with very large numbers of little-seen word forms that might bloat our vocabulary unnecessarily. Additionally, SentencePiece’s model helps to strike a balance between word-based and character-based tokenization so that it captures the most informative representations by looking at both their appearance frequencies and how they appear in the same context in the corpus.

In short, using the trained, customised SentencePiece BPE tokenizer allows the model to better capture the linguistic particularities of Azerbaijani, thereby making it more efficient in training and yielding better performance in terms of the resulting LLaMA-2 model’s ability to decode and generate it, by capturing the most linguistically informative and context-sensitive combinations of subwords.

4.2.2 Training Process and Parameters

The training configuration starts with initialising the model weights randomly because the model is not using any pre-trained embeddings. The choice was to be especially sensitive to training the language model so that learning is specifically related to the linguistic facts and characteristics of Azerbaijani language and, most importantly, learning is thoroughly adapted to the Azerbaijani linguistic context rather than being biased towards patterns present in pre-trained language model embeddings.

Our computational infrastructure was set up with four NVIDIA V100 GPUs, each with 16 GB VRAM, because it’s a solid GPU processing setup that can handle huge amounts of data and models with considerable complexity required for training massive natural language models. For enhanced management of the model training infrastructure, Lightning AI was used. Experiment tracking and visualisation of training progress, including the whole training process, was done using the Weights and Biases (Wandb) platform. This tool was invaluable not only for its real-time data-logging capabilities but also for its ability to provide insights into the model’s performance dynamics and operational adjustments.

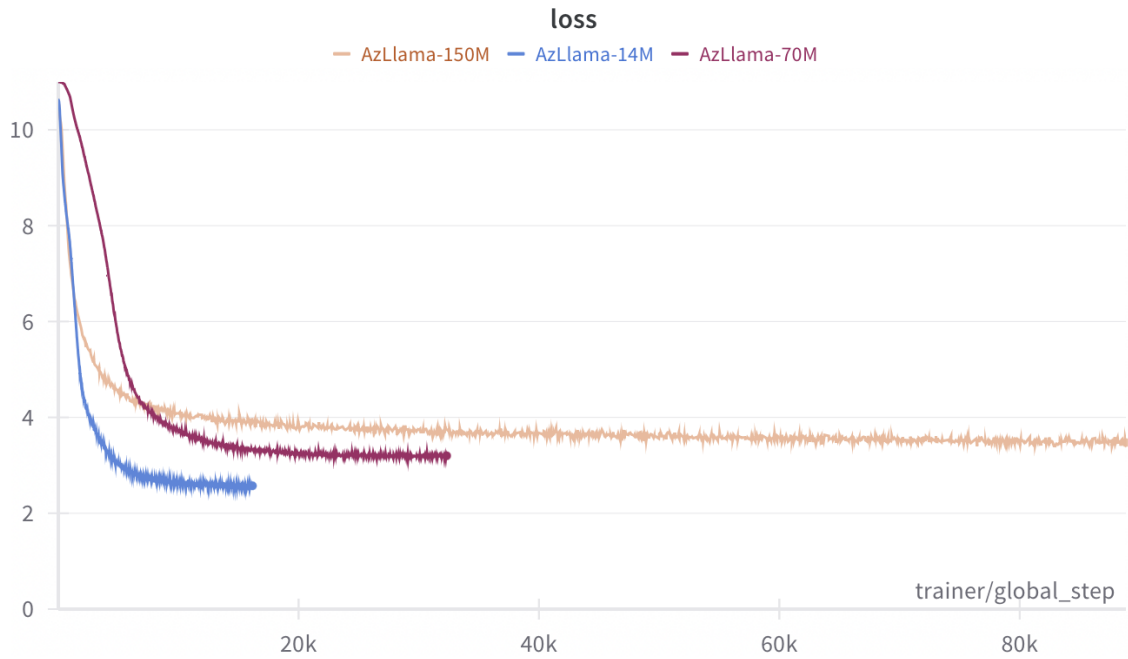


Figure 8: Training loss. In practice, bigger models tends to have lower loss when compared to lower counterparts. But here we observe the vice versa. The reason for this is that first the small models (AzLlama-14M) has been only trained only Wikipedia while the biggest model has been trained on all the available data including internet. And the second reason is that the 150M model is undertrained which means upon training additional epochs would end up in the convergence (See Figure 11).

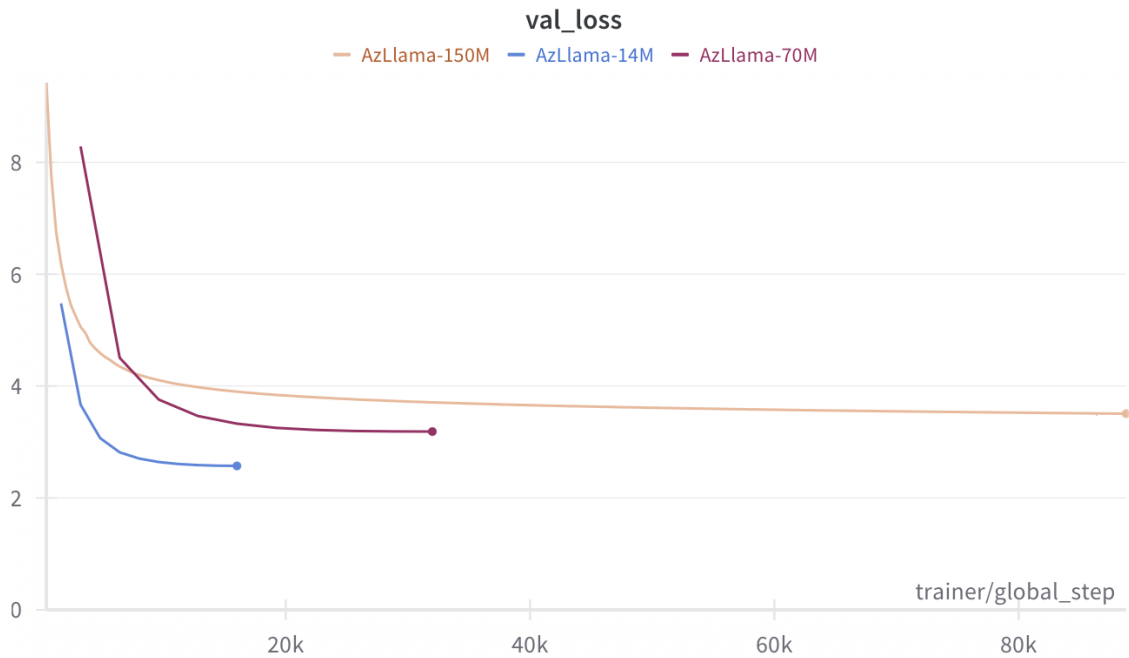


Figure 9: Validation loss

The exact training parameters were chosen to maximise both learning effectiveness and computational efficiency. The initial learning rate of 0.0006 was chosen experimentally and found to be the best value, which makes possible for our model to converge with the given

training data and architecture in the scalable fashion. A global batch size of 512 was set. This was divided into a micro batch containing 32 examples. With this segmentation, we were able to achieve the best GPU utilisation and memory usage, effectively allowing the hardware to run near 99 per cent capacity.

We set up the training to cycle through a corpus of about 9.6 billion tokens (a large but reasonable corpus to induce the subtleties of the language into the model), and run for three epochs. The learning rate was also warmed up and decayed. This means the learning rate ramped up over 1400 steps, peaked, and then underwent a cosine decay, which stabilised the rate of learning and led to smooth parameter optimisation.

Some other things we took into consideration in our training were a weight decay factor 0.1, to help prevent overfitting by punishing larger weights, a network optimised by an Adam optimiser with the beta values 0.9 and 0.95 and a clipping gradient threshold of 1.0 so that the flows of gradient in backpropagation are stable.

Table 4: Training Parameters and Performance Metrics of the AzLlama-150M Model

Training Parameters	Value		
Learning Rate	0.0006		
Weight Decay	0.1		
Micro Batch Size	32		
Max Tokens	9e+9		
Gradient Accumulation Steps	4		
beta1	0.9		
beta2	0.95		
Minimum Learning Rate	0.00006		
		Performance Metrics	Value
		Validation Loss	3.506
		Validation Perplexity	33.33
		Batches per Second	9.17
		FLOPs per Second	1.24e+14

Regular intervals for logging, evaluation, and checkpointing were integrated into the training schedule. These frequent checkpoints, set to occur every 100 iterations, were critical for continuous monitoring and evaluation of the model’s performance. They provided essential data points for adjusting training parameters and strategies as needed, ensuring the model’s development was on track and aligned with our objectives.

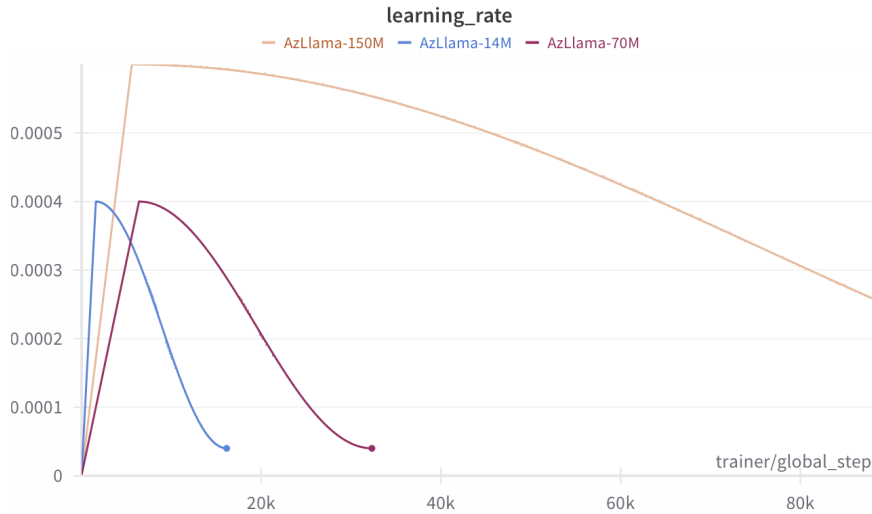


Figure 10: Learning Rate Schedule for AzLlama: The model employs a learning rate schedule that includes a warmup phase for the initial 10,000 steps out of a total of 90,000 steps, followed by a cosine decay. The maximum learning rate is set at $6e-4$, with a minimum learning rate of $6e-5$. This approach helps in stabilizing the training process early on and gradually fine-tuning the learning rate to enhance model performance over time.

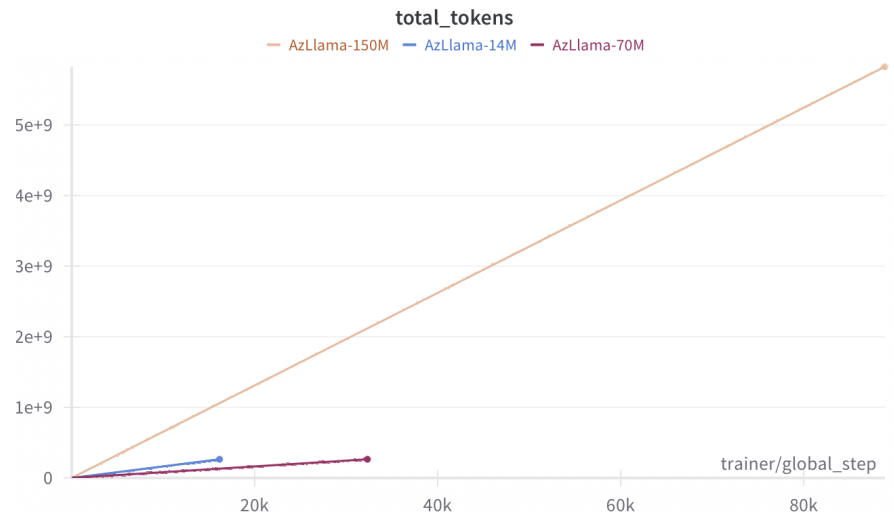


Figure 11: Number of Global Training Tokens for AzLlama: Currently, AzLlama has been trained on 5 billion tokens, which suggests potential undertraining. To optimize its performance, there is scope to extend training for one or two additional epochs, potentially increasing the training volume to between 9 and 12 billion tokens. This expansion could significantly enhance the model’s understanding and generation capabilities in Azerbaijani.

4.3 Instruction Fine-tuning

4.3.1 Foundation Model vs. Supervised Fine-tuning

The foundational base model is typically trained to predict the next token given the context that came before; this is usually through unsupervised training on large sets of data, and the model gains knowledge of many patterns of language through exposure. For tasks done

in high-resource languages like English, such foundational models achieve good performance across a wide variety of applications (Devlin et al., 2019; Brown et al., 2020). Going from general-purpose to instruction-following requires something known as supervised fine-tuning.

However, transitioning from a broad-capability model to a specialized instruction-following model necessitates supervised fine-tuning. Supervised fine-tuning involves re-training the model on a dataset where the inputs are paired with the desired outputs, typically high-quality and specific to the tasks the model needs to perform. This method not only refines the model’s abilities to follow explicit instructions but also enhances its precision in generating responses that adhere closely to the specified guidelines (Houlsby et al., 2019).

4.3.2 Instruction-following dataset

For this project, the choice of the Alpaca instruction dataset from Stanford (Taori et al., 2023) is pivotal. Originally in English, this dataset has been carefully translated into Azerbaijani to cater to the local linguistic context. The translation is critical because it adapts culturally and contextually relevant instructions, thereby making the dataset more effective for training in a specific language environment.

The supervised fine-tuning (SFT) leverages the translated Alpaca dataset, focusing on the quality and specificity of the data. While pre-training can work with large volumes of lower-quality data, fine-tuning requires significantly less data but of much higher quality—on the order of 10,000 to 50,000 high-quality pairs. This is because fine-tuning aims to sharpen the model’s ability to execute tasks precisely as instructed, where the correctness and relevance of the response are crucial (Raffel et al., 2019).

4.3.3 Training Details and Adjustments for Azerbaijani Language

In tailoring the pre-training process for the Azerbaijani version of the Alpaca instruction dataset, specific training configurations have been carefully chosen to optimize the model’s learning capacity and efficiency. The model, “AzLlama” has been set to train across three epochs over 3 billion tokens. This number of epochs is chosen to balance between adequate learning and preventing overfitting, given the high-quality but limited size of the fine-tuning dataset.

The training arguments leverage the use of advanced optimization techniques, specifically the paged AdamW 32-bit optimizer optimizer. This optimizer is part of newer strategies in machine learning that aim to improve memory efficiency and training speed, which are critical when training models with large parameter spaces on limited hardware resources. Gradient checkpointing is another crucial feature employed here, allowing the model to handle larger portions of data at a time by saving only necessary intermediate states, thus reducing the overall memory requirement.

Table 5: Configuration Parameters of the Instruction Following Model

Parameter	Value
learning_rate	0.0002
num_hidden_layers	12
num_attention_heads	12
hidden_size	768
intermediate_size	3,072
max_position_embeddings	512
vocab_size	32,000
optimizer	"paged_adamw_32bit"
gradient_accumulation_steps	2
max_length	20
warmup_steps	0
weight_decay	0

Batch size settings are adaptively configured based on the availability of "flash attention" technology. If enabled, 256 batch sizes are used per device, allowing the model to process more data at once, leveraging the efficient memory management provided by flash attention. Without flash attention, the batch size is reduced significantly to 4 to ensure that training remains feasible without excessive memory usage. This is complemented by setting gradient accumulation steps to 2, which helps in managing GPU memory better by accumulating gradients over multiple steps before updating model weights.



Figure 12: Training loss on the instruction following dataset.

Learning rate is another critical parameter, set at $2e-4$ for this training phase, chosen to provide a steady and gradual adjustment of model weights, minimizing the risk of diverging during training. A low warmup ratio of 0.03 is used to slowly ramp up the learning rate at the beginning of training, which helps in stabilizing the model updates early in training. The learning rate scheduler is kept constant, which simplifies the training dynamics and is often sufficient for fine-tuning scenarios where the model is already pre-trained on a large corpus and only needs minor adjustments.

The use of mixed precision training, specifically choosing bf16 over fp16, reflects a strategic choice to balance training speed and model performance. This decision is particularly important given the computational resources typically available for language model training, where bf16 offers a good compromise between precision and speed. The model's configuration also explicitly disables tf32 and fp16, favoring bf16 for its efficiency on compatible hardware.

4.3.4 Challenges with Translated Data and Domain-Specific Content

One significant challenge in this process is the use of translated training data. While translation allows for training in Azerbaijani, nuances and contextual meaning can sometimes be lost, potentially leading to less effective training outcomes. Moreover, the lack of domain-specific question-answer pairs in Azerbaijani limits the model's ability to perform well in niche areas without additional data augmentation or synthetic data generation strategies.

5 Model Evaluation

5.1 Evaluation Metrics

In the evaluation of language models, benchmarks such as the Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2020), ARC Challenge (Clark et al., 2018), and HumanEval (Chen et al., 2021) are typically employed to test a model’s ability to understand and generate responses across a broad range of topics and problem types. However, the AzLlama model, due to its specialized nature and smaller scale, does not align well with these extensive benchmarks. Instead, we have chosen a suite of metrics that are better suited to assessing a language model focused on Azerbaijani.

Table 6: (Large) Language Models (LLMs) Information

Model	# Parameters	# Training Tokens	Access Type	Organization
claude-haiku	>1T	No Information	proprietary	Anthropic
gpt3.5-turbo	>100B	No Information	proprietary	OpenAI
llama3-70b-instruct	70B	15T	open-weight	Meta
AzLlama-150M	150M	9B	open-weight	ADA & GW
mixtral-8x22b-instruct	8x22B	No Information	open-weight	Mistral AI
llama3-8b-instruct	8B	15T	open-weight	Meta
gemma-7b-it	7B	6T	open-weight	Google
mixtral-8x7b-instruct	8x7B	No Information	open-weight	Mistral AI
mistral-7b-instruct	7B	8T	open-weight	Mistral AI
llama2-70b-instruct	70B	2T	open-weight	Meta

5.1.1 Contextual Alignment Score

The Contextual Alignment Score, including F1, Recall, and Precision components, is derived from BERTScore by Zhang et al. (2019), which utilizes contextual embeddings from a BERT model to evaluate the quality of text generated by language models. It compares the similarity of word embeddings between generated and reference texts. It specifically measures precision, recall, and F1 score, using embeddings from models like BERT. For this work, we utilize multilingual BERT to support the Azerbaijani language. To compute the BERTScore, each token in the generated text is compared to each token in the reference text by calculating the cosine similarity between their embeddings. The precision for a generated document is the average cosine similarity of each token in the generated text to its most similar token in the reference text.

$$\text{Precision} = \frac{1}{|C|} \sum_{c \in C} \max_{r \in R} \cos(\text{embed}(c), \text{embed}(r))$$

where C is the set of tokens in the candidate sentence, R is the set of tokens in the reference sentence, $\text{embed}(x)$ denotes the BERT embedding of token x , and \cos denotes the cosine similarity. Conversely, recall is the average of each token in the reference text to its most similar token in the generated text.

$$\text{Recall} = \frac{1}{|R|} \sum_{r \in R} \max_{c \in C} \cos(\text{embed}(r), \text{embed}(c))$$

where the roles of C and R are reversed compared to the precision calculation. The F1 score is the harmonic mean of precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The BERTScore, through these metrics, leverages the power of BERT’s deep contextual embeddings to capture the subtle semantic and syntactic nuances of language. By evaluating the cosine similarities between embeddings, the BERTScore effectively measures how tokens contribute to the overall context, making it a powerful tool for assessing the quality of responses generated by large language models.

Table 7: Contextual Alignment Metrics

Model	Precision (%)	Recall (%)	F1 Score (%)
claude-haiku	73.02	74.00	73.50
gpt3.5-turbo	71.31	72.56	71.92
llama3-70b-instruct	68.94	72.58	70.69
AzLlama-150M	67.78	69.01	68.35
mixtral-8x22b-instruct	65.72	68.93	67.26
llama3-8b-instruct	64.50	69.15	66.72
gemma-7b-it	64.75	68.78	66.66
mixtral-8x7b-instruct	65.52	66.89	66.18
mistral-7b-instruct	63.34	64.44	63.87
llama2-70b-instruct	59.08	63.40	61.13

5.1.2 Semantic Scores Using Sentence Transformers

A measure of the overall semantic similarity between whole sentences of generated texts and reference texts is provided by this score which uses Sentence-BERT by Reimers and Gurevych (2019) (a variant of BERT optimized for generating sentence-level embeddings). This metric evaluates how well does a model preserve meaning over complete sentences, which is important for applications that require high comprehension and output quality. In order to compare full sentences instead of individual words or tokens, standard BERT architecture was modified by Sentence-BERT. Each sentence is passed through the Sentence-BERT model to obtain dense vector representation for the entire sentence. Then cosine similarity between these vector representations calculates semantic similarity. Mathematically, the similarity between two sentences can be expressed as:

$$\text{Semantic Similarity} = \cos(\vec{v}_1, \vec{v}_2) = \frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \|\vec{v}_2\|}$$

where \vec{v}_1 and \vec{v}_2 are the vector embeddings of the generated text and the reference text, respectively. Instead of concentrating on sentence-level alignment and semantic features, BERTScore operates with token-level alignments and their contextual embeddings to calculate precision, recall, and F1 measures. The text being evaluated is divided into smaller parts by BERTScore. It does this by checking if tokens in a generated text correspond with those in a reference text at the word or phrase level using their embeddings. This technique enables one to know how accurate terms are within a given context semantically. Rather than assessing individual phrases or words like BERTScore does, Semantic Scores appraise whole sentences’ aggregate semantics content. Unlike considering single entities only when embedding them Sentence-BERT looks at broader implications and relations between different parts that make up complete thoughts while representing complex ideas through language structures which convey such meanings across cultures too; hence this method can be useful for evaluating texts where overall clarity or theme development matters most rather than specific lexical items’ truthfulness alone.

Table 8: Semantic Scores

Model	Semantic Scores
claude-haiku	85.14
gpt3.5-turbo	84.96
AzLlama-150M	82.87
mixtral-8x22b-instruct	82.49
llama3-8b-instruct	82.30
llama3-70b-instruct	81.55
mixtral-8x7b-instruct	75.60
llama2-70b-instruct	65.78
gemma-7b-it	60.03
mistral-7b-instruct	59.57

5.1.3 Character n-gram F-score (CHRF)

The Character n-gram F-score (CHRF) is an evaluation metric which judges the degree of character n-grams overlapped between generated text and reference texts. Azerbaijani has complex morphology due to its agglutinative structure; therefore, this measure gives a detailed estimation at the level of word formation that other methods might miss for such languages as it measures only morpheme boundaries or individual characters without considering their order within words. Basically, CHRF scores how well does generated text match with reference by checking if all its character n-grams are present there or not and also whether they are correct or incorrect. It uses three values – precision, recall and F1 score – where each stands for itself but combines two others into single number representing overall quality of given answer set according to some criteria.

Precision and recall for character n-grams measure the accuracy and coverage of generated text relative to a reference. Precision assesses the proportion of correct n-grams in the generated text, while recall evaluates the extent to which the generated text covers the n-grams in the reference text.

$$\text{Precision } P = \frac{c_{\text{match}}}{c_{\text{gen}}} \quad \text{Recall } R = \frac{c_{\text{match}}}{c_{\text{ref}}}$$

where c_{match} is the number of matching character n-grams found in both the generated and reference texts, c_{gen} is the total number of character n-grams in the generated text and c_{ref} is the total number of character n-grams in the reference text.

Finally, F-score (CHRF) is the harmonic mean of precision and recall, and it is often weighted to emphasize recall more than precision, reflecting the importance of not missing critical content:

$$\text{F-score} = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R}$$

where β is a parameter that weights recall more heavily than precision, often set to a value greater than 1.

For Azerbaijani, an agglutinative language where affixes concatenated to a base word form convey significant syntactic and semantic information, CHRF is particularly informative. The metric’s focus on character-level analysis allows it to capture and evaluate the linguistic nuances that arise from the language’s morphological richness. This is essential for ensuring that language models accurately replicate the complex structures of Azerbaijani, maintaining both the linguistic integrity and the subtleties of meaning that character n-grams can embody.

Table 9: Character n-gram F-score (CHRF)

Model	CHRF Score (%)
claude-haiku	42.73
gpt3.5-turbo	41.06
llama3-70b-instruct	40.49
AzLlama-150M	34.82
llama3-8b-instruct	33.29
mixtral-8x22b-instruct	33.05
mixtral-8x7b-instruct	31.51
gemma-7b-it	31.47
mistral-7b-instruct	26.10
llama2-70b-instruct	23.36

5.2 Performance Comparison with Open-Weight and Proprietary Models

To put AzLlama’s performance into perspective, it was compared against many well-known models in the field, including open-weight models like GPT-4 and proprietary ones such as Llama-3 and Mistral. GPT-4 served as a reference point because it is both very strong and widely recognized among language model researchers as being useful for benchmarking purposes. The measures we used to compare these systems are detailed above.

Compared to other open-weight alternatives, AzLlama-150M achieves an F1 of 68.35 showing good precision and recall for contextual alignment given its smaller size. The only non-proprietary model that does better than AzLlama-150M is the LLaMA-3 70B which is on par with GPT-4 but has been trained on 70 billion parameters over 15 trillion tokens where 5% of them were non-English. This large corpus training helps a lot in Azerbaijani despite its being mainly an English model. In terms of proprietary models, Claude tops with an F1 score of 73.5 followed by GPT-3.5 from the same organization as our reference model GPT

In semantic similarity, AzLlama-150M’s performance of 82.87 is notably competitive, even surpassing several larger models like LLaMA 3 70B. This achievement underscores its specialization in understanding and generating semantically rich responses in Azerbaijani, affirming its suitability for applications requiring nuanced language use. Only proprietary models can perform better than AzLlama where the order is the same, Claude Haiku followed by GPT 3.5. We should not say that Those two models are considered to be the same level in the market former was developed by Anthropic and the latter by OpenAI.

The Character-Level Accuracy (CHRF) score of 34.82, where the order of the initial models is identical to the contextual alignment score shows the model to handle the complex morphology of Azerbaijani. And again the only open-weight model that outperforms the AzLlama is LLaMA 3 70B.

5.3 Strengths and Limitations of the Developed Model

The potential of AzLlama in specialized applications within the Azerbaijani language is promising but contingent on several factors. Specifically, the model has proven its ability to abstract more linguistically accurate outputs in Azerbaijani than other larger models; therefore, scalable efficiency could be stored in increasing its size and training on more tokens. In this case, it might also abstract more culturally and contextually nuanced possible completions. Similarly, the possible customization of this model could be beneficial for applications targeting a region, such as AzLlama serving as a model benchmark for educational tools or local content creation

systems. However, this specialization is only possible if the model is customized and expanded to handle more tasks and complexities within the Azerbaijani language. This focus could also allow AzLlama to outperform some proprietary models that have not gone through extensive model scaling or rely on a consistent model that underscores topical or industry coverage. However, model scaling remains a concern since stronger computational resources will be required to achieve more focused use without overfitting to a known model. The best outcome of a better-used case than giants could be achieved through a stronger model and more diverse dataset. Data diversity and representation will especially affect its performance, highlighting that claims of current bias and gaps in language data could also be a limiting factor. To conclude, AzLlama is set up to serve better use cases within its niche potential, aiming at a truly impactful deployment driven forward by larger scalability, model focus, and increased data diversity within a fluid natural language processing space.

6 Conclusion

6.1 Potential Use Cases in Azerbaijani

Finally, the language model developed for Azerbaijani demonstrates numerous applications that could directly contribute to benefiting the Azerbaijani-speaking community. With this in mind, the most apparent use case would be in the field of customer support, where the model could be used as a part of chatbots responding to inquiries from native speakers. This would ensure customer satisfaction and overall convenience of interactions. Moreover, given the language model’s potential to generate consistent and contextually relevant text, content generation scenarios show some potential, such as creating articles, stories, and educational text. However, it is essential to mention that the model still needs significant development to ensure coherent high-quality performance in response to such tasks. The ability to increase the model and enhance training on a broader, more diverse dataset can be vital in this case. The same goes for translation and localization, as AzLlama can theoretically be used for such purposes but requires work on its efficiency in handling specific linguistic and cultural characteristics. Finally, content-related surrealism can be used the same way, given these scenarios’ reliance on the model’s internal architecture and training process. These factors require substantial work, but the results have the potential to remove language barriers and enhance global content access for Azerbaijani speakers. However, these capacities are currently limited in the AzLlama model’s capacity regardless of its promising functionality in the content-related field. Moreover, work on AzLlama’s cultural preservation capacities can be conducted to increase this outcome’s efficiency in the context of Azerbaijani historical text translation and interpretation. Again, this would depend on enhanced model quality, which remains mostly theoretical at this stage. Although this potential is almost realized, the work is necessary to form a reliable basis for using this language model for specific content creation and translation needs.

6.1.1 Implications for Low-Resource Language Processing

In conclusion, the present research contributes to the development of NLP in Azerbaijani and sets an example for other low-resource languages. The ability to create a usable and efficient language model with a comparatively small amount of resources makes it feasible to implement similar projects in other marginalized languages. This may reduce the demand for proprietary models and thus make language technology more widely accessible. Furthermore, the development of language models acts as a driver for innovation, potentially attracting more academic and commercial attention to the Azerbaijani language. This can result in developing further linguistic tools and materials and improving the linguistic support for low-resource languages. Finally, the ethical implications of this investigation are significant as well. It demonstrates the need for inclusive technologies that give marginalized linguistic communities equal opportunities and influence on the technological spheres.

6.2 Summary of Findings

The thesis developed ‘AzLlama’, a specialized generative language model for Azerbaijani, leveraging the LLaMA architecture with 150 million parameters. This marks a significant advancement in technology for low-resource languages, utilizing an extensive dataset of approximately 3 billion Azerbaijani tokens from diverse sources such as Wikipedia, OSCAR, mC4, books, and news articles. The model was effectively pre-trained and fine-tuned to demonstrate its capability to generate coherent Azerbaijani text and respond to simple queries. Notably, despite its relatively small size compared to larger multibillion-parameter models, AzLlama competes favorably in linguistic competence, highlighting the effectiveness of targeted training approaches

for low-resource contexts.

The research successfully bridged a significant gap by providing a robust model capable of understanding and generating Azerbaijani text, a notable achievement given the challenges associated with low-resource language modeling. The model’s performance, particularly in generating high-quality text with minor errors, underscores its potential as a valuable tool for various applications requiring Azerbaijani language processing.

6.3 Concluding Remarks

In conclusion, this thesis is a significant contribution to the field of natural language processing as it fills a critical gap in Azerbaijani language processing resources. AzLlama’s success in language modeling and understanding not only shows that creating effective language models for low-resource languages is possible but also paves the way for future prospects. The flexibility and adaptability of AzLlama, as well as the groundwork laid for expanding its capabilities and datasets, lay the groundwork for creating a model capable of solving more complex linguistic puzzles with greater accuracy and efficiency. Future research may look into enlarging the model and offering it more computational power to see how much of the AzLlama efficiency can be maintained while realizing the benefits of a larger machine. Other avenues to explore include a hybrid methodology that combines a large-scale multilingual foundation with smaller customized models to eliminate the resource limitations that low-resource language processing imposes.

Overall, this project, completed with the guidance of Dr. Samir Rustamov and the support of ADA University and George Washington University, demonstrates the capacity of academic research to push the boundaries of what is technologically possible, expanding our knowledge and capacities while also advocating for language diversity in technology.

6.4 Recommendations for Future Work

Building on the foundations laid by this thesis, future research can explore several promising avenues:

- **Model Scaling:** There is substantial scope for enhancing the model by scaling up the architecture and incorporating more comprehensive data sets. Such advancements could improve the model’s accuracy and utility across a broader range of applications.
- **Cross-Linguistic Models:** Investigating the development of models that function across multiple related low-resource languages could leverage shared linguistic features to improve efficiency and performance. Specifically, focusing on the Oghuz branch of the Turkic language family, to which Azerbaijani belongs, could be highly beneficial. This group includes languages like Turkish, Turkmen, and Gagauz, among others. By exploring these linguistically similar languages, we can harness their common grammatical, lexical, and phonological traits to enhance model training and broaden the scope of language technology applications.
- **Synthetic data generation:** In the development of language technologies, generating synthetic data for training large language models (LLMs) like ChatGPT and Claude is particularly beneficial for languages such as Azerbaijani. This approach, superior to machine translation, aligns more closely with the linguistic and cultural nuances of Azerbaijani. Consider Cosmopedia (Ben Allal et al., 2024), an English dataset generated by Mixtral MoE (Mixture of Experts), comprising over 30 million files and 25 billion tokens. It reflects a variety of topics aimed at mirroring world knowledge. Similarly, creating a

comprehensive synthetic dataset for Azerbaijani would enhance the performance of LLMs by integrating cultural and contextual nuances, ensuring outputs that are both linguistically accurate and contextually relevant. By focusing on synthetic data generation, we can create Azerbaijani language models that exceed the capabilities of those merely relying on translated content, paving the way for more sophisticated and localized AI applications.

- **Continuous pre-training:** It is feasible to expand the tokenizer and embedding of already pre-trained English LLMs to further pre-train these models in the Azerbaijani language. Subsequently, these models can be fine-tuned using instruction-following datasets tailored to Azerbaijani, enhancing their functionality and applicability in context-specific scenarios.
- **Application Expansion:** Future work should also consider expanding into other NLP applications, such as speech recognition and generation for Azerbaijani, to fully harness the capabilities of modern AI in supporting spoken as well as written language.
- **Collaborative Projects:** Establishing partnerships with academic institutions, tech companies, and community organizations can enhance data collection efforts, refine model training, and ensure the model’s applications align with the community’s needs.
- **Sustainability and Ethics:** Ongoing research must continuously address the sustainability of these technologies, focusing on ethical considerations such as data privacy, consent, and the potential cultural impact of deploying AI in sensitive contexts.

References

- AI, L. (2023). Litgpt. <https://github.com/Lightning-AI/litgpt>.
- Aida-zade, K., Xocayev, A., and Rustamov, S. (2016). Speech recognition using support vector machines. In *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–4.
- Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebron, F., and Sanghai, S. (2023). GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.
- Albalak, A., Elazar, Y., Xie, S. M., Longpre, S., Lambert, N., Wang, X., Muennighoff, N., Hou, B., Pan, L., Jeong, H., et al. (2024). A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*.
- Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. (2024). Cosmopedia.
- Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., Kandola, J., Hofmann, T., Poggio, T., and Shawe-Taylor, J. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3(6):1137–1155.
- Broder, A. (1997). On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29.
- Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D. X., Erlingsson, Ú., Oprea, A., and Raffel, C. (2020). Extracting training data from large language models. In *USENIX Security Symposium*.
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Chi, Z., Huang, H., Liu, L., Bai, Y., Gao, X., and Mao, X.-L. (2024). Can pretrained english language models benefit non-english nlp systems in low-resource scenarios? *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:1061–1074.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

- CONNEAU, A. and Lample, G. (2019). Cross-lingual language model pretraining. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Dancker, J. (2022). A brief introduction to recurrent neural networks. *Towards Data Science*.
- Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M., and Gardner, M. (2021). Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Hasanli, H. and Rustamov, S. (2019). Sentiment analysis of azerbaijani tweets using logistic regression, naive bayes and svm. In *2019 IEEE 13th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–7.
- Heafield, K. (2011). KenLM: Faster and smaller language model queries. In Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. F., editors, *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.
- Hedderich, M. A., Lange, L., Adel, H., Strötgen, J., and Klakow, D. (2021). A survey on recent approaches for natural language processing in low-resource scenarios. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y., editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568, Online. Association for Computational Linguistics.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. (2022). Training compute-optimal large language models.
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In Chaudhuri,

- K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.
- Hu, S., Tu, Y., Han, X., He, C., Cui, G., Long, X., Zheng, Z., Fang, Y., Huang, Y., Zhao, W., et al. (2024). Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.
- Ibe, C. (2024). Unlocking low-resource language understanding: Enhancing translation with llama 3 fine-tuning.
- Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In Lapata, M., Blunsom, P., and Koller, A., editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.
- Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Blanco, E. and Lu, W., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Lent, D., Novaes, M., Carvalho, L. C., Lloret, J. L., Rodrigues, J. R., and Jr, M. (2022). A gated recurrent unit deep learning model to detect and mitigate distributed denial of service and portscan attacks. *IEEE Access*, 10(-):73229–73242.
- Liu, H., Xue, W., Chen, Y., Chen, D., Zhao, X., Wang, K., Hou, L., Li, R., and Peng, W. (2024). A survey on hallucination in large vision-language models.
- Marchisio, K., Lewis, P., Chen, Y., and Artetxe, M. (2023). Mini-model adaptation: Efficiently extending pretrained models to new languages via aligned shallow training. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5474–5490, Toronto, Canada. Association for Computational Linguistics.
- Micallef, K., Gatt, A., Tanti, M., van der Plas, L., and Borg, C. (2022). Pre-training data quality and quantity for a low-resource language: New corpus and BERT models for Maltese. In Cherry, C., Fan, A., Foster, G., Haffari, G. R., Khadivi, S., Peng, N. V., Ren, X., Shareghi, E., and Swayamdipta, S., editors, *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 90–101, Hybrid. Association for Computational Linguistics.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Nguyen, T., Nguyen, C. V., Lai, V. D., Man, H., Ngo, N. T., Deroncourt, F., Rossi, R. A., and Nguyen, T. H. (2023). Culturax: A cleaned, enormous, and multilingual dataset for large language models in 167 languages.
- Ogueji, K., Zhu, Y., and Lin, J. (2021). Small data? no problem! exploring the viability of pretrained multilingual language models for low-resourced languages. In Ataman, D., Birch, A., Conneau, A., Firat, O., Ruder, S., and Sahin, G. G., editors, *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 116–126, Punta Cana, Dominican Republic. Association for Computational Linguistics.

- Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Quast, B. (2016). rnn: a recurrent neural network in r. *Working Papers*.
- Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2018). Language models are unsupervised multitask learners.
- Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., et al. (2021). Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.
- Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Rustamov, S. and Mustafayev, E. (2009). Principles of construction of speech recognition system by the example of azerbaijani language.
- Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ili’c, S., Hesslow, D., Castagn’e, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurencon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., and Gokaslan, A. (2022). Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100.
- Schreiber, J., Bilmes, J., and Noble, W. S. (2020). apricot: Submodular selection for data summarization in python. *Journal of Machine Learning Research*, 21(161):1–6.
- Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2023). Roformer: Enhanced transformer with rotary position embedding.
- Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. (2023). Stanford alpaca: An instruction-following llama model.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Welbl, J., Glaese, A., Uesato, J., Dathathri, S., Mellor, J., Hendricks, L. A., Anderson, K., Kohli, P., Coppin, B., and Huang, P.-S. (2021). Challenges in detoxifying language models. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2447–2469, Punta Cana, Dominican Republic. Association for Computational Linguistics.

- Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1):1–191.
- Yang, W., Li, C., Zhang, J., and Zong, C. (2023). Bigtrans: Augmenting large language models with multilingual translation capability over 100 languages. *arXiv preprint arXiv:2305.18098*.
- Zhang, R., Gangi Reddy, R., Sultan, M. A., Castelli, V., Ferritto, A., Florian, R., Sarioglu Kayi, E., Roukos, S., Sil, A., and Ward, T. (2020). Multi-stage pre-training for low-resource domain adaptation. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5461–5468, Online. Association for Computational Linguistics.
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.
- Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Appendix

A.1 Parameters for AzLlama-150M Foundation (Base) Model

Table 10: Configuration of the Training Model

Parameter	Value
batch_size	128
beta1	0.9
beta2	0.95
decay_lr	true
devices	4
eval_iters	100
eval_step_interval	100
global_batch_size	512
grad_clip	1
gradient_accumulation_steps	4
learning_rate	0.0006
log_iter_interval	4
log_step_interval	1
logger_name	"wandb"
max_tokens	9,600,000,000
micro_batch_size	32
min_lr	0.00006
model_name	"tiny-llama-1.1b"
name	"lit-tiny-llama-162M-v2"
save_step_interval	100
warmup_iters	5,600
warmup_steps	1,400
weight_decay	0.1

Table 11: Additional Configuration Parameters of the AzLlama-150M Foundation (Base)

Parameter	Value
Attention Bias	false
Attention Dropout	0.0
BOS Token ID	1
EOS Token ID	2
Hidden Activation Function	silu
Hidden Size	768
Initializer Range	0.02
Intermediate Size	3072
Max Position Embeddings	512
Model Type	llama
Number of Attention Heads	12
Number of Hidden Layers	12
Number of Key-Value Heads	4
Pretraining TP	1
RMS Norm Epsilon	1e-05
Rope Scaling	null
Rope Theta	10000.0
Tie Word Embeddings	false
Torch Data Type	float16
Transformers Version	4.38.2
Use Cache	false

Table 12: Summary Metrics of the Model Training

Metric	Value
batches	88,904
batches_per_sec	9.171458342248371
device/batches_per_sec	2.2928645855620933
device/flops_per_sec	31,064,816,991,650.38
device/items_per_sec	37,566.293369849336
device/mfu	0.24851853593320303
device/samples_per_sec	73.37166673798698
epoch	1
flops_per_sec	124,259,267,966,601.52
items_per_sec	150,265.17347939734
iter	88,904
iter_time	0.47941186399839353
learning_rate	0.00025362398796338755
lengths	1,456,603,136
loss	3.4525585174560547
remaining_time	27,543.75131495078
samples	2,844,928
samples_per_sec	293.48666695194794
step	22,226
time	42,527.781641270994
tokens	1,456,603,136
total_tokens	5,826,412,544
trainer/global_step	88,904
val_loss	3.5063562393188477
val_ppl	33.32661207768247

A.2 Parameters for AzLlama-150M Intruction Following Model

Table 13: Key Configuration Parameters of the Model

Parameter	Value
_name_or_path	"eljanmahammadli/AzLlama-152M"
learning_rate	0.0002
num_hidden_layers	12
num_attention_heads	12
hidden_size	768
intermediate_size	3,072
max_position_embeddings	512
vocab_size	32,000
optimizer	"paged_adamw_32bit"
gradient_checkpointing	true
bf16	true
gradient_accumulation_steps	2
max_length	20
warmup_steps	0
weight_decay	0
logging_steps	10
save_steps	500
eval_strategy	"no"

Table 14: Training Statistics of the Model

Metric	Value
Epochs	3
Global Step	7,734
Gradient Norm	1.09765625
Learning Rate	0.0002
Average Loss	1.6629
Total FLOPs	24,403,524,777,934,850
Train Loss	1.9439417214735184
Training Runtime (sec)	2,113.388
Samples Per Second	29.278
Steps Per Second	3.66

A.3 Model Architecture

The architecture of the LlamaForCausalLM model implemented in PyTorch is outlined below:

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(32000, 768)
    (layers): ModuleList(
      (0-11): 12 x LlamaDecoderLayer(
        (self_attn): LlamaSdpaAttention(
          (q_proj): Linear(in_features=768, out_features=768, bias=False)
          (k_proj): Linear(in_features=768, out_features=256, bias=False)
          (v_proj): Linear(in_features=768, out_features=256, bias=False)
          (o_proj): Linear(in_features=768, out_features=768, bias=False)
          (rotary_emb): LlamaRotaryEmbedding()
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=768, out_features=3072, bias=False)
          (up_proj): Linear(in_features=768, out_features=3072, bias=False)
          (down_proj): Linear(in_features=3072, out_features=768, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): LlamaRMSNorm()
        (post_attention_layernorm): LlamaRMSNorm()
      )
    )
    (norm): LlamaRMSNorm()
  )
  (lm_head): Linear(in_features=768, out_features=32000, bias=False)
)
```

A.4 Question and Answer Interactions with AzLlama-150M Model

Below are several interactions featuring questions posed to the AzLlama-150M model, followed by the responses generated by the model. These interactions illustrate the model's capabilities in understanding and generating text based on the input provided. We should not that the model shines mostly in the general knowledge questions rather than factual questions though it can answer simple factual-based questions as well

USER: Təbiət haqqında şeir yaz

**ASSISTANT: Təbiətin gözəlliyi, lütf səndə,
Ürəyimizi dərinliyə aparır.
Hər kölgədə rəqs edirəm,
Təbiət geniş mənzərələrlə üzləşir.
Hər cür canlılara ev sahibliyi etdiyimiz təəccüblü yer,
Göyü sarsılmaz bir hədiyyə.**

**Dünya həmişə gözəllik vahasıdır.
Beləliklə, təbiətin təklif etdiyi hər şeylə yaşayaq.
Möcüzələr diyarı kimi əzəmətli,
təbiət, oh, sirlə və mehriban,
Belə İlahi aləmə səyahət edirik.
Sonsuz zövq, sonsuz potensialımızla,
Bizi bələd olaraq təyin edən hədiyyə,
Onun nemətlərinə hörmət edəcəyik,
Dünyanı bir möcüzə ilə qoruyacağıq.**

Figure 13: General Knowledge Question 1

USER: Qlobal istiləşmənin qarşısını almaq üçün nə etməliyik?

ASSISTANT: Qlobal istiləşmə, əsasən fosil yanacaqların yandırılması, meşələrin qırılması və intensiv əkinçilik kimi insan fəaliyyətindən qaynaqlanan qlobal bir problemdir. Davamlı metodlar tətbiq etməklə istixana qazı emissiyalarını azaltmaq və planetimizin gələcək nəsillər üçün qorunmasına töhfə verə bilərik.

Birincisi, iqlim dəyişikliyinə dağıdıcı təsirlərini azaltmaq üçün bütün ölkələrin dərhal və qətiyyətli hərəkətlərinə keçməliyik. Ölkələrimiz problemin ciddiliyini başa düşməli və onları necə həll edəcəyimizə dair tədbirlər görmüşlər qlobal istiləşmə haqqında. Məsələn, karbon dioksid emissiyalarının azaldılması hədəflərimizə doğru ilk addımı atmağa kömək edəcəkdir.

Bundan əlavə, hamımız öz üzərimizə düşəni edə bilərik. Qlobal istiləşmə ilə mübarizə üçün hamımızdan nə gözlədiyini bilmək vacibdir. Gələcək nəsilləri qorumaq istəyiriksə, hərəkətə keçməyimizə, hərəkətə keçməyə və davamlı və ədalətli bir keçid üçün birlikdə çalışmağımıza çox inanıram.

Məsələn, qlobal miqyasda daha təmiz hava və suya çıxışın təmin edilməsi, bərpa olunan enerji mənbələrinin tətbiqi, dəniz səviyyəsinin yüksəlməsi, biomüxtəlifliyin qorunması və bioloji növlərimiz təhdid edildikdə, hərəkətlərimizə və hədəflərimizə uyğun dəyişikliklər etməliyik. Təhsil, təhsil, səhiyyə və digər ehtiyacları dəstəkləmək üçün daha çox potensialımız olanlara can atmalıyıq.

Figure 14: General Knowledge Question 2

USER: Stresslə mübarizə üçün nə kimi rahatlama texnikaları tövsiyə edirsiniz?

ASSISTANT: Streslə məşğul olmaqdan zövq alırsınızsa, stres səviyyənizi azaltmaq inanılmaz dərəcədə faydalı ola bilər. Budur bəzi tövsiyələr:

1. Dağınıqlıqla mübarizə aparın: dağınıq və xaotik hiss edirsinizsə, ancaq narahatlıq keçirirsinizsə, dağınıqlar və ya maneələr barədə kimsə ilə danışmaq yaxşıdır. Bu ağıl qıcıqlanma gətirəcək və narahatlığı azalda biləcək kiçik şeylər siyahısı hazırlamağa kömək edərdi.
2. Yuxu: siqaret çəkməkdən, alkoqoldan, balanssızlıqdan və həddindən artıq iş yükünüzdən qurtulmağı öyrənin. Bu kömək edə bilər gələcəkdə daha çox irəliləməyə kömək edəcəkdir.
3. Özünü qulluq edin: fiziki və zehni rifah üçün özünü kifayət qədər vaxt ayırdığınızdan əmin olun. Yaxşı istirahət etmək üçün müntəzəm fasilələr verin və ya fiziki fəaliyyətlə məşğul olun ki, nəfəsinizə diqqət yetirə və daha sağlam bir həyat yaşaya bilərsiniz.
4. İdman: mütəmadi olaraq idman etmək əhval-ruhiyyənizi artırmağa kömək edə bilər, konsentrasiyanı və hərəkətliliyi artırır.
5. Kofein və nikotini məhdudlaşdırın: Bəzi dərmanlar pis xolesterol verə bilər (LDL) və ürək xəstəliyi riskinin artması ilə əlaqələndirilir ki, bu da kofeinə olan həvəsinizi azalda bilər.
6. Mədəni və ya emosional olaraq məşq edin-istər fiziki cəhətdən aktiv bir karyera qurmaq, istərsə də səylərinizi buna uyğun istiqamətdə istifadə etmək, stresi azaltmağa və ümumi zehni sağlamlığınızı yaxşılaşdırmağa kömək edə bilər.

Figure 15: General Knowledge Question 3

USER: Uğur qazanmaq üçün nə etmək lazımdır?

ASSISTANT: Uğur, istər şəxsi, istər sosial, istərsə də karyera məqsədləri olsun, uğurlu bir insanın ehtiyac duyduğu bir səviyyədir. Uğur yolunda sizi dəstəkləyəcək və dəstəkləyən güclü motivasiyalı və qətiyyətli bir insan tapmağa imkan verir. Budur sizə kömək edə biləcək bir neçə addım:

- Gücünüzü artırmaq üçün: hədəflərinizi daha kiçik, əldə edilə bilən hədəflərə bölmək, sizi daha böyük və daha çox güclü edəcək.
- Uğurunuzu artırır: bir hədəfə çatdıqdan sonra ona doğru irəliləyişi izləyin. Nə qədər çox çalışsanız, bir o qədər motivator olardı uğur qazanmağınıza kömək edir.
- Əzmkarlıq: nəhayət, irəliləməyə davam etmək üçün gücünüz qalıcı olacaq.
- Sizin üçün vacibdir imkanlar, potensialınız var, buna görə nailiyyətlərinizdən aydın zövq alın və uğursuzluqlardan öyrənin.
- Emosional zəka: güclü bir zehniyyətlə özünüzü mükafatlandıraraq özünüzü inamlı və nikbin hiss edin. Unutmayın ki, zəhmətiniz boşa getməyib və ruhdan düşməyin-uğur sizi gözləyir. Qarşılaşdığınız çətinliklərə görə özünü qarşı çox sərt olmayan, bunu özünüz üçün ayağa qalxmaq və özünüzü yaxşı hiss etdiyiniz zaman fasilə verməyi unutmayın!

Figure 16: General Knowledge Question 4

USER: Fransanın paytaxtı hansı şəhərdir?

ASSISTANT: Fransanın Paytaxtı Parisdir.

Figure 17: Factual Question 2

USER: İkinci dünya müharibəsi nə vaxt olub?

ASSISTANT: İkinci Dünya Müharibəsi 19 noyabr 1939-cu ildə Almanıyanın Polşanı işğal etdiyi zaman elan edildi.

Figure 18: Factual Question 2

A.5 Pre-trained model (not instruction following)

```
<s> S: Azərbaycanın paytaxtı haradır?  
C: Azərbaycanın paytaxtı Bakı şəhəridir.  
S: Türkiyənin paytaxtı haradır?  
C: Türkiyənin paytaxtı Ankara şəhəridir.  
S: Fransanın paytaxtı haradır?  
Fransanın paytaxtı Paris şəhəridir. Parisin tarixi haqqında müxtəlif fikirlər mövcuddur.
```

Figure 19: Few-shot prompting the base model (not instruction following). Since the base model is not able to answer questions as an assistant it is possible to use few-shot prompts to get answers to the question but it is not as stable as the instruction following model. As mentioned before base model is essentially a document completer

```
'<s> Azərbaycanın paytaxtı Bakı şəhəridir. Bakı şəhərinin tarixi hissəsi olan  
İçərişəhərin ərazisində yerləşən bu şəhər Azərbaycanın ən qədim şəhərlərindən  
biridir. İçərişəhərin ərazisində yerləşən Qız qalası Azərbaycanın qədim tarixə malik  
şəhərlərindən biridir. İçərişəhərin ərazisində yerləşən Qız qalası XII əsrdə  
tikilmişdir. Qız qalası XII əsrdə Şirvanşahlar dövlətinin paytaxtı olmuşdur. Qız  
qalası Bakının mərkəzində yerləşən Qız qalasının yaxınlığında yerləşir. Qız qalası  
Bakının qədim tarixə malik olan qədim tikililərindən biridir. Qız qalası Bakının
```

Figure 20: Base model also does not know when to stop and can generate duplicated sentences if repetition penalty is not applied. It is given a sentence and then the model continues to generate more tokens. It is not possible to ask questions and get answers contrary to the instructions following the model.

A.6 Additional training metrics

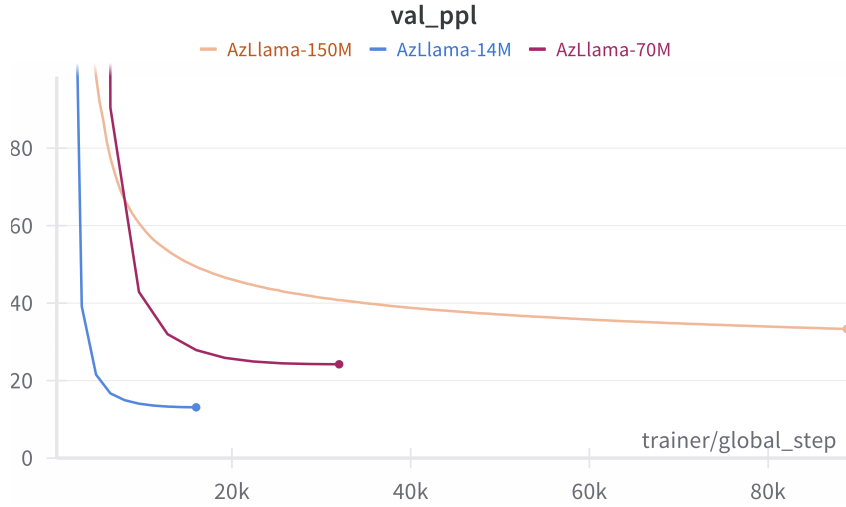


Figure 21: Validation perplexity

A.7 GPU Usage

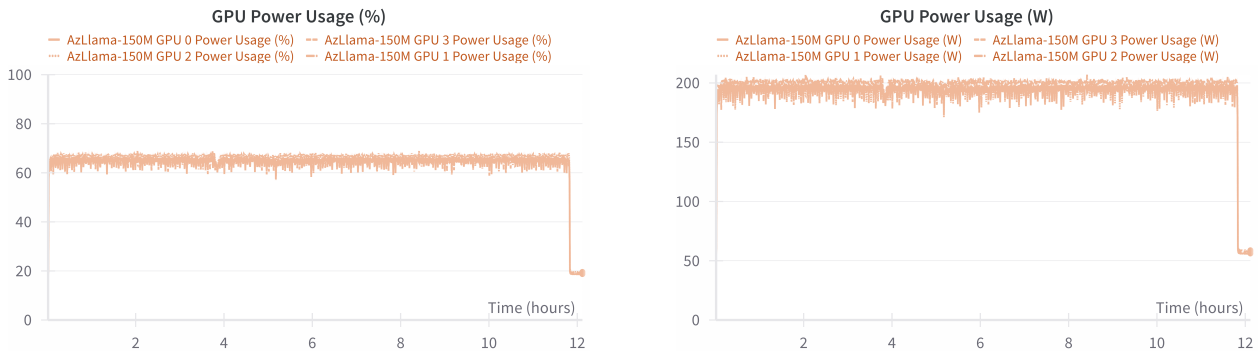


Figure 22: GPU Utilization

A.8 Parameter Counts

Layer Type	Number of Parameters
embedding	24576000
attention	18874368
mlp	84934656
normalization	19200

Table 15: Grouped parameter counts by layer type

A.9 Chat-bot UI

In large language models (LLMs), controlling the style and coherence of generated text is crucial. Parameters such as temperature, top-p, repetition penalty, and top-k play pivotal roles

in managing the diversity and predictability of the output. Here's a brief explanation of each parameter and its impact on text generation. Temperature controls the randomness in text generation, with higher values producing more varied and creative outputs, while lower values yield more predictable and conservative results. Top-p (nucleus sampling) specifies a threshold to sample words from the set of most likely next words, reducing the randomness by focusing only on the most probable subset. Repetition penalty discourages the model from repeating the same words or phrases by reducing the likelihood of previously generated tokens, promoting more diverse and coherent text. Top-k sampling limits the next word choices to the top k most likely candidates, streamlining the generation process by ignoring less probable options.

AzLlama-150M Chatbot

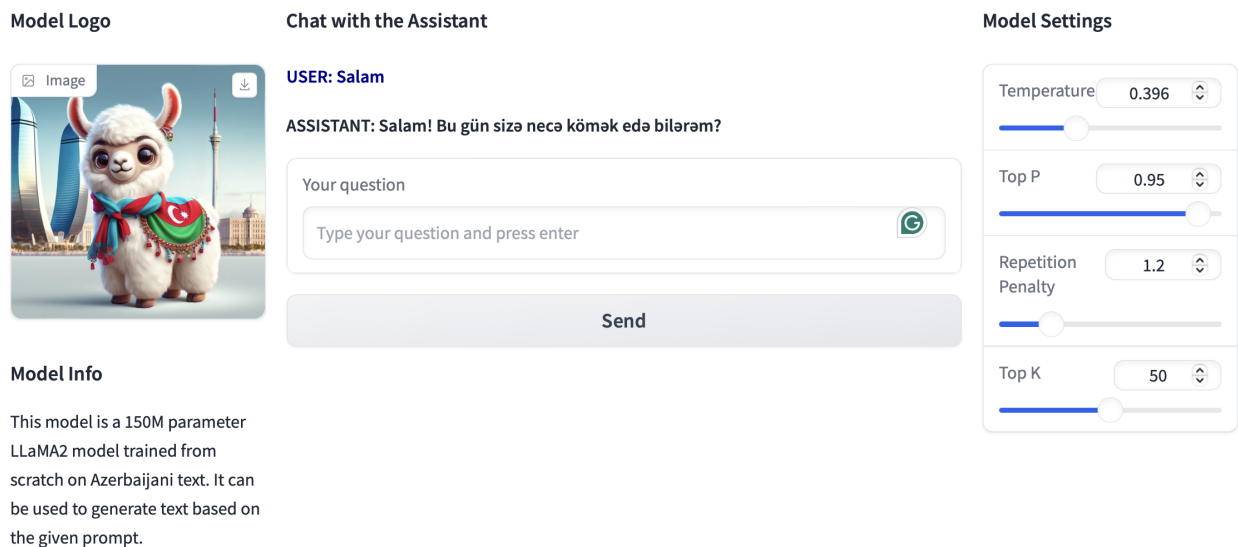


Figure 23: Chat-bot User Interface for AzLlama-150M

A.10 Code snippets for Pre-Training

[H] There are lots of main and utility scripts to prepare text corpus, tokenization, pre-training, converting model checkpoints and evaluation. That is why here is very limited code that is used for mainly pre-training. Pre-training library used for this project is from Lightning AI AI (2023) but there are other scripts for different stages. Below code are mainly for pre-training only.

```

def main(fabric, resume, context_size):
    if fabric.global_rank == 0:
        out_dir.mkdir(parents=True, exist_ok=True)

        config = Config.from_name(
            model_name,
            block_size=context_size,
            vocab_size=32000,
            n_layer=12,
            n_head=12,
            n_embd=768,
            intermediate_size=3072,
        )

        train_dataloader, val_dataloader = create_data_loaders(batch_size=micro_batch_size, block_size=config.block_size)
        train_dataloader, val_dataloader = fabric.setup_data_loaders(train_dataloader, val_dataloader)
        fabric.seed_everything(3407)
        fabric.print(f"Loading model with {config.__dict__}")
        t0 = time.perf_counter()
        with fabric.init_module(empty_init=False):
            model = GPT(config)
            model.apply(partial(init_weights, n_layer=config.n_layer, n_embd=config.n_embd))

        fabric.print(f"Time to instantiate model: {time.perf_counter() - t0:.02f} seconds.")
        fabric.print(f"Total parameters: {num_parameters(model):,}")

        model = torch.compile(model)
        model = fabric.setup(model)
        optimizer = torch.optim.AdamW(
            model.parameters(), lr=learning_rate, weight_decay=weight_decay, betas=(beta1, beta2), foreach=False
        )
        optimizer = fabric.setup_optimizers(optimizer)
        state = {"model": model, "optimizer": optimizer, "hparams": hparams, "iter_num": 0, "step_count": 0}

        if resume is True:
            resume = max(out_dir.glob("*.pth"), key=(lambda p: int(p.name.split("-")[1])))
        if resume:
            fabric.print(f"Resuming training from {resume}")
            fabric.load(resume, state)

        train_time = time.perf_counter()
        train(fabric, state, train_dataloader, val_dataloader, resume)
        fabric.print(f"Training time: {(time.perf_counter()-train_time):.2f}s")
        if fabric.device.type == "cuda":
            fabric.print(f"Memory used: {torch.cuda.max_memory_allocated() / 1e9:.02f} GB")

```

Figure 24: Configuring and Executing Distributed Training Sessions: This Python code is for setting up and executing a distributed training session of a GPT model. The script incorporates initial configuration, data loader preparation, model instantiation, and optimizer setup. It also handles conditional resumption from saved checkpoints, providing detailed output about the training process and resource utilization.

```

def train(fabric, state, train_dataloader, val_dataloader, resume):
    model = state["model"]
    optimizer = state["optimizer"]

    validate(fabric, model, val_dataloader, max_iters=2)
    throughput = ThroughputMonitor(fabric, window_size=5)

    with torch.device("meta"):
        meta_model = GPT(model.config)
        x = torch.randint(0, 1, (micro_batch_size, meta_model.config.block_size))
        model_fwd = lambda: meta_model(x)
        model_loss = lambda y: chunked_cross_entropy(y, x, chunk_size=0)
        measured_flops = measure_flops(meta_model, model_fwd, model_loss)
        fabric.print(f"Measured TFLOPs: {measured_flops * fabric.world_size / 1e12:.2f}")
        del meta_model, x

    max_tokens_per_device = max_tokens // fabric.world_size
    tokens_per_iter = micro_batch_size * model.config.block_size
    max_iters = max_tokens_per_device // tokens_per_iter
    initial_iter = state["iter_num"]
    train_iterator = CycleIterator(train_dataloader)
    print(f"Number of training iters: {max_iters}")

    if resume:
        resume_t0 = time.perf_counter()
        for resume_iter in range(initial_iter):
            next(train_iterator)
            if resume_iter % 1000 == 0:
                fabric.print(f"Resuming dataset: {resume_iter} / {initial_iter}")
        fabric.barrier()
        fabric.print(
            f"Resuming data loader finished. Took {time.perf_counter() - resume_t0:.1f} seconds to reach iteration"
            f" {initial_iter}, epoch {train_iterator.epoch}."
        )

    running_loss = RunningMean(window=gradient_accumulation_steps, sync_on_compute=False).to(fabric.device)
    fabric.barrier()
    total_t0 = time.perf_counter()

    for train_data in train_iterator:
        if state["iter_num"] >= max_iters:
            print(f"Current ite {state['iter_num']}")
            break

        lr = get_lr(state["iter_num"], max_iters) if decay_lr else learning_rate
        for param_group in optimizer.param_groups:
            param_group["lr"] = lr

        state["iter_num"] += 1
        iter_t0 = time.perf_counter()

        input_ids = train_data[:, 0 : model.config.block_size].contiguous().long()
        targets = train_data[:, 1 : (model.config.block_size + 1)].contiguous().long()

```

Figure 25: Optimizing Training Workflows: Monitoring, Efficiency, and Dynamic Adaptation: Code snippet highlights optimizing training workflows for a GPT model. The script involves validation checks, performance monitoring through TFLOP calculations, and dynamic training iteration management based on data distribution across computing nodes. Additionally, it handles efficient resume operations from previous training states and dynamic learning rate adjustments, enhancing both training speed and model performance under distributed computing environments.

```

is_accumulating = state["iter_num"] % gradient_accumulation_steps != 0
with fabric.no_backward_sync(model, enabled=is_accumulating):
    logits = model(input_ids)
    loss = chunked_cross_entropy(logits, targets)
    fabric.backward(loss / gradient_accumulation_steps)

running_loss.update(loss.detach())

if not is_accumulating:
    fabric.clip_gradients(model, optimizer, max_norm=grad_clip)
    optimizer.step()
    optimizer.zero_grad()
    state["step_count"] += 1

if state["iter_num"] % log_iter_interval == 0:
    loss = running_loss.compute().item()
    t1 = time.perf_counter()
    throughput.update(
        time=(t1 - total_t0),
        flops=(measured_flops * log_iter_interval),
        batches=state["iter_num"],
        samples=(state["iter_num"] * micro_batch_size),
        lengths=(state["iter_num"] * micro_batch_size * model.config.block_size),
    )
    metrics = {
        "loss": loss,
        "iter": state["iter_num"],
        "step": state["step_count"],
        "epoch": train_iterator.epoch,
        "iter_time": t1 - iter_t0,
        "remaining_time": (
            (t1 - total_t0) / (state["iter_num"] - initial_iter) * (max_iters - state["iter_num"])
        ),
        "tokens": state["iter_num"] * micro_batch_size * model.config.block_size,
        "total_tokens": state["iter_num"] * micro_batch_size * model.config.block_size * fabric.world_size,
        "learning_rate": lr,
    }

    fabric.print(
        f"iter {metrics['iter']} | step {metrics['step']}: loss {metrics['loss']:.4f}, iter time:"
        f" {metrics['iter_time'] * 1000:.2f} ms{' (optimizer.step),' if not is_accumulating else ','}"
        f" remaining time: {metrics['remaining_time'] / 3600 / 24:.2f} days"
        f" {metrics['remaining_time'] / 3600:.4f} hours"
    )

    throughput_metrics = throughput.compute()
    metrics.update(throughput_metrics)
    fabric.log_dict(metrics, step=state["iter_num"])

```

Figure 26: Gradient Management and Performance Monitoring: This code is for fine-tuning the training process of an AI model. The script uses advanced gradient accumulation techniques to optimize resource utilization and reduce training time. It includes detailed performance logging and monitoring, dynamically adjusting the learning rate, and efficiently managing checkpoints. The code also handles validation at specific intervals to ensure model accuracy.

```

@torch.no_grad()
def validate(fabric: L.Fabric, model: nn.Module, val_dataloader: DataLoader, max_iters: int) -> torch.Tensor:
    fabric.print("Validating ...")
    model.eval()

    losses = torch.zeros(max_iters, device=fabric.device)
    for k, val_data in enumerate(val_dataloader):
        if k >= max_iters:
            break
        input_ids = val_data[:, 0 : model.config.block_size].contiguous().long()
        targets = val_data[:, 1 : (model.config.block_size + 1)].contiguous().long()
        logits = model(input_ids)
        loss = chunked_cross_entropy(logits, targets)
        losses[k] = loss

    model.train()
    return losses.mean()

def get_lr(it: int, lr_decay_iters: int) -> int:
    if it < warmup_iters:
        return learning_rate * it / warmup_iters
    if it > lr_decay_iters:
        return min_lr
    decay_ratio = (it - warmup_iters) / (lr_decay_iters - warmup_iters)
    assert 0 <= decay_ratio <= 1
    coeff = 0.5 * (1.0 + math.cos(math.pi * decay_ratio)) # coeff ranges 0..1
    return min_lr + coeff * (learning_rate - min_lr)

def init_weights(module: nn.Module, n_layer: int, n_embd: int):
    if isinstance(module, nn.Embedding):
        nn.init.normal_(module.weight, mean=0.0, std=math.sqrt(2.0 / 5 / n_embd))
    elif isinstance(module, nn.Linear):
        nn.init.normal_(module.weight, mean=0.0, std=math.sqrt(2.0 / 5 / n_embd))
        if module.bias is not None:
            nn.init.zeros_(module.bias)
    for name, param in module.named_parameters():
        if name == "proj.weight" and isinstance(module, (LLaMAMLP, CausalSelfAttention)):
            nn.init.normal_(param, mean=0.0, std=(1 / math.sqrt(n_embd) / n_layer))

```

Figure 27: Streamlining Model Validation and Optimizing Learning Rates: Those utility functions are for validation and learning rate adjustment during AI training. The script includes a validate function for efficiently assessing model performance on validation data without affecting gradients, and a get learning rate function that dynamically adjusts the learning rate based on training progress, ensuring optimal convergence. Additionally, an initialization of weights function sets initial weights of neural network components to support stable training.