Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Иркутский национальный исследовательский технический университет»

Байкальский институт БРИКС

Baikal school of BRICS

**О Т Ч Ё Т**

|  |  |  |
| --- | --- | --- |
| о прохождении | | производственной |
|  | | (вид практики: учебная/производственная) |
|  | технологической (проектно-технологической) практики | |
| (тип практики: технологическая/научно-исследовательская работа/преддипломная и др.) | | |
|  | | |
|  | в БИ БРИКС ФГБОУ ВО ИРНИТУ | |
| (наименование профильной организации или структурного подразделения вуза) | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Обучающегося | Лежнев Евгений | ИИКб-23-1 | \_\_\_\_\_\_ |
|  | (ФИО) | (группа) | (подпись) |
| Руководитель практики от БИ БРИКС  Григорьев С.В., доцент \_\_\_\_\_\_\_\_\_\_\_  ФИО (должность) (подпись) | | | |

|  |
| --- |
| Руководитель ООП  Афанасьев А.Д., профессор \_\_\_\_\_\_\_\_\_\_\_  ФИО (должность) (подпись) |

Оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Григорьев С.В. \_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» 2025 г.

(ФИО) (подпись)

|  |  |  |
| --- | --- | --- |
| Содержание отчета на |  | стр. |
| Приложение к отчету на |  | стр. |

Иркутск 2025 г.

**Индивидуальное задание на прохождение**

|  |
| --- |
| производственной |
| технологической (проектно-технологической) практики |

для Лежнев Евгений обучающегося 2 курса, группы ИИКб-23-1

|  |  |
| --- | --- |
| по направлению подготовки/специальности | 09.03.01 Информатика и |
| вычислительная техника (англоязычная программа) | |

|  |  |
| --- | --- |
| по программе | Искусственный интеллект и компьютерные науки /  Artificial Intelligence and Computer Science |

|  |  |
| --- | --- |
| Место прохождения практики: | БИ БРИКС ФГБОУ ВО ИРНИТУ |

Сроки прохождения: с 09.06.2025 по 06.07.2025

|  |  |
| --- | --- |
| Цели и задачи прохождения практики: | To design and develop a prototype of a Telegram bot utilizing a trained translation model. |
| 1) To create and maintain a GitHub project repository for the internship work.  2) To analyze a real-world enterprise problem and propose a solution using an appropriate Artificial Intelligence method or model.  3) To design and develop a prototype of a computer application based on the selected AI method or model. | |

|  |  |
| --- | --- |
| Содержание практики, вопросы, подлежащие изучению: |  |
| 1) Sequence-to-sequence (seq2seq) model architectures, used in machine translation tasks. | |
| 2) Working with pre-trained models (Helsinki-NLP/opus-mt-en-ru) and their fine-tuning. | |
| 3) Integration of a trained model into an interactive application (using a Telegram bot). | |
|  | |

|  |  |
| --- | --- |
| Планируемые результаты практики: | Report for Internship and Project Defense |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель практики от института | | | |
|  | / | Григорьев С.В. | / |
| (подпись) |  | (ФИО) |  |
| Согласовано:  Руководитель ООП | | | |
|  | / | Афанасьев А.Д. | / |
| (подпись) |  | (ФИО) |  |

С настоящим индивидуальным заданием и с программой практики ознакомлен(а), задание принято к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ «9» июня 2025 г.

(подпись)

**Содержание / Contents**

[Introduction 4](#_Toc202680329)

[1 Creating a Project from GitHub 5](#_Toc202680330)

[1.1 Creating a repository 5](#_Toc202680331)

[2 Training a translation model based on a pre-trained Seq2Seq model using the Transformer architecture. 5](#_Toc202680332)

[2.1 What is Seq2Seq model? 5](#_Toc202680333)

[2.1.1 Encoding the Input Sequence 6](#_Toc202680334)

[2.1.2 Decoding the Output Sequence 6](#_Toc202680335)

[2.1.3 Advantages of Seq2Seq Models 6](#_Toc202680336)

[2.1.4 Disadvantages of Seq2Seq Models 6](#_Toc202680337)

[2.2 what is the Transformer architecture? 7](#_Toc202680338)

[2.2 What is the Helsinki-NLP model? 8](#_Toc202680339)

[2.3 Prepare a dataset 9](#_Toc202680340)

[2.4 Training a translation model 9](#_Toc202680341)

[2.5 Results of model training 9](#_Toc202680342)

[3 Integrating a trained translation model into a Telegram bot 11](#_Toc202680343)

[3.1 what is a BotFather? 11](#_Toc202680344)

[3.2 Building the Bot with python‑telegram‑bot 11](#_Toc202680345)

[3.3 Bot implementation 12](#_Toc202680346)

[Conclusion 12](#_Toc202680347)

[Перечень условных обозначений / The listing of document symbols 14](#_Toc202680348)

[References 15](#_Toc202680349)

# Introduction

Natural Language Processing (NLP) technologies have become a key area of research and development in the field of artificial intelligence in recent years. Among their numerous applications, machine translation holds a special place — the task of converting text from one language to another while preserving its semantics and structure. This problem is not only of theoretical interest for the study of language models, but also of practical value in the context of globalization, where the need for instant access to information in multiple languages is becoming critically important.

During my internship, I set out to design and implement a complete end‑to‑end English‑to‑Russian machine translation service. The project encompassed every stage of an ML workflow: initializing and structuring a GitHub repository, collecting and preprocessing a parallel corpus of texts, fine‑tuning a pre‑trained Seq2Seq Transformer model, analyzing training results, and finally deploying the trained model as a Telegram bot. The primary motivation was to gain hands‑on experience with advanced NLP tools and techniques, working with large datasets, adjusting hyperparameters of deep neural networks, and integrating a model into an actual user‑facing application.

The goal of the report is to to design and develop a prototype of a Telegram bot utilizing a trained translation model. [In order to achieve the set](https://context.reverso.net/%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4/%D0%B0%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9-%D1%80%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9/In+order+to+achieve+the+set) goal it is necessary to solve the following objectives:

1. Create GitHub Project;
2. Build a generative neural network model for language processing;
3. Build an application based on Generative Language Model.

# 1 Creating a Project from GitHub

# 1.1 Creating a repository

When initializing the repository, I carefully considered its folder structure to clearly reflect the workflow’s stages. The dataset/ directory contains scripts and templates for loading Parquet files and performing basic cleaning. Under src/, you’ll find module responsible for data preparation, tokenization, and training a model. The model/ folder holds the training scripts (train.py) and configuration files detailing TrainingArguments settings. The bot/ folder houses the Telegram bot implementation code.

To keep documentation separate from code, I established a docs branch that stores written reports in DOCX format, exported graphs, and slide deck templates for the final presentation. This approach preserved the main branch’s cleanliness and ensured that bulky assets did not clutter the codebase’s history.

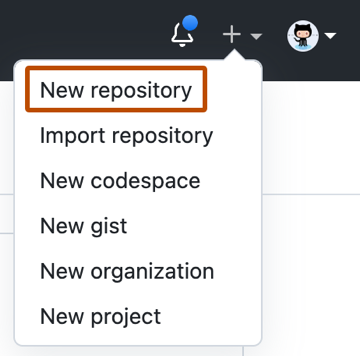


Figure 1 - Creating a repository

# 2 Training a translation model based on a pre-trained Seq2Seq model using the Transformer architecture.

# 2.1 What is Seq2Seq model?

The Sequence-to-Sequence (Seq2Seq) model is a type of neural network architecture widely used in machine learning particularly in tasks that involve translating one sequence of data into another. It takes an input sequence, processes it and generates an output sequence. The Seq2Seq model has made significant contributions to areas such as natural language processing (NLP), machine translation and speech recognition.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Figure 2.1 - Encoder and Decoder Stack in seq2seq model

# 2.1.1 Encoding the Input Sequence

The encoder processes the input sequence token by token, updating its internal state at each step. After the entire sequence is processed, the encoder produces a context vector, a fixed-length representation that summarizes the important information from the input.

# 2.1.2 Decoding the Output Sequence

The decoder takes the context vector as input and generates the output sequence one token at a time. For example, in machine translation, it can convert the sentence “I am learning” into “Je suis apprenant” sequentially, predicting each token based on the context and previously generated tokens.

# 2.1.3 Advantages of Seq2Seq Models

1. **Flexibility**: Can handle tasks like machine translation, text summarization and image captioning with variable-length sequences.
2. **Handling Sequential Data**: Ideal for sequential data like natural language, speech and time series.
3. **Context Awareness**: Encoder-decoder architecture captures the context of the input sequence to generate relevant outputs.
4. **Attention Mechanism**: Focuses on key parts of the input sequence, improving performance, especially for long inputs.

# 2.1.4 Disadvantages of Seq2Seq Models

1. **Computationally Expensive**: Requires significant resources to train and optimize.
2. **Limited Interpretability**: Hard to understand the model's decision-making process.
3. **Overfitting**: Prone to overfitting without proper regularization.
4. **Rare Word Handling**: Struggles with rare words not seen during training.

# 2.2 what is the Transformer architecture?

transformer is an architecture based on the multi-head attention mechanism, in which text is converted to numerical representations called tokens, and each token is converted into a vector via lookup from a word embedding table. At each layer, each token is then contextualized within the scope of the context window with other (unmasked) tokens via a parallel multi-head attention mechanism, allowing the signal for key tokens to be amplified and less important tokens to be diminished.

Transformers have the advantage of having no recurrent units, therefore requiring less training time than earlier recurrent neural architectures (RNNs) such as long short-term memory (LSTM). Later variations have been widely adopted for training large language models (LLMs) on large (language) datasets.

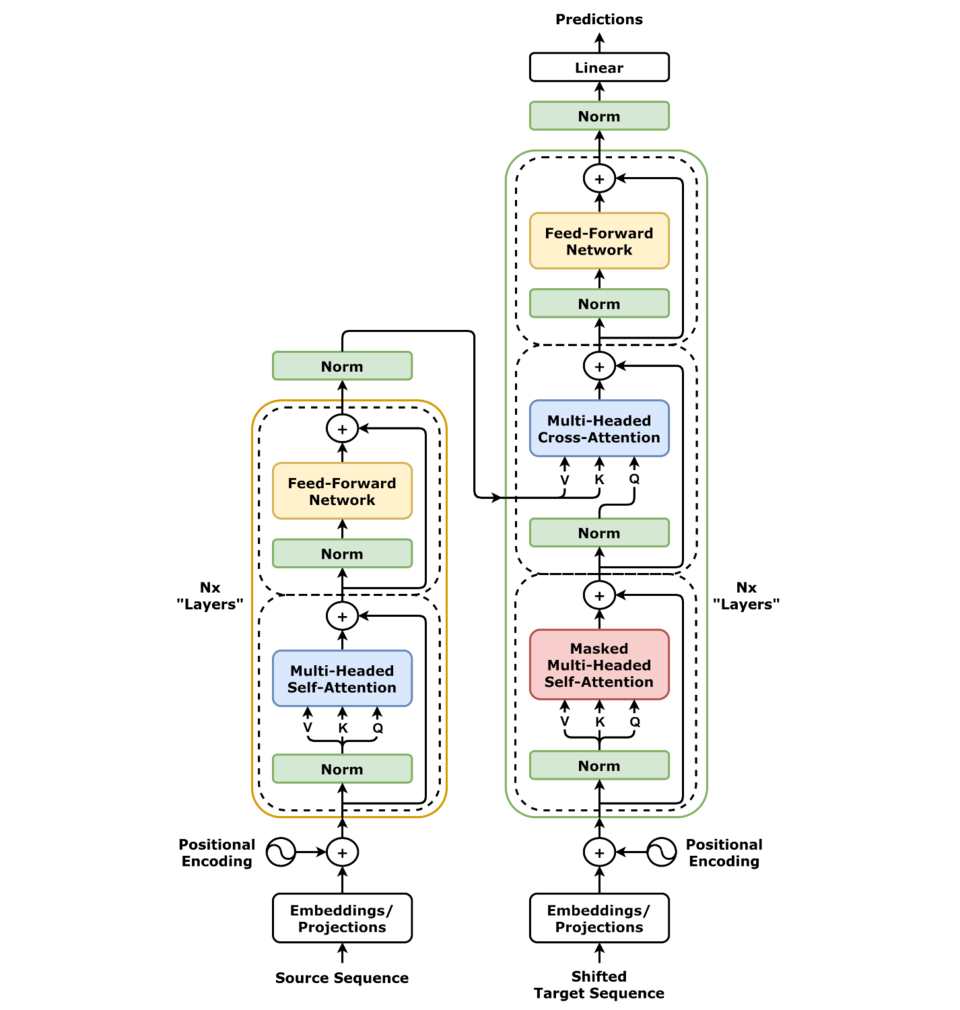


Figure 2.2 - A standard Transformer architecture

# 2.2 Reasons for Selecting the Helsinki-NLP Model

For fine‑tuning, I chose the Helsinki‑NLP/opus‑mt-en-ru model, a MarianMT implementation based on the Transformer architecture. This model features six encoder and six decoder layers, an embedding size of 512, and eight attention heads per layer. It was pre‑trained on extensive parallel corpora from the OPUS project, such as EuroParl and OpenSubtitles, giving it a strong initial grasp of English‑Russian translation patterns.

This particular model was selected for its favorable trade‑off between translation quality and resource requirements. It runs efficiently on a single modern GPU with 8 GB of memory, making it practical for both development and deployment. The adoption of MarianMT through Hugging Face Transformers also provided a well‑supported ecosystem for tokenization, training, and inference.

# 3. Training a Translation Model Based on the Helsinki-NLP Model

# 3.1 Prepare a dataset

My training data came from the cointegrated/nli-rus-translated-v2021 dataset on Hugging Face, which comprises automatically translated English sentences paired with their Russian counterparts. The training split consists of three Parquet files (train1.parquet, train2.parquet, train3.parquet) totaling roughly 1.76M sentence pairs, while a fourth file (val.parquet) holds approximately 34.6K pairs for validation.

First, I loaded all Parquet files into Pandas DataFrames and concatenated the training parts into a single cohesive DataFrame. Data cleaning steps included dropping any rows with missing values in either premise or premise\_ru columns, trimming leading and trailing whitespace, and normalizing line breaks and extra spaces. I then constructed a list of dictionaries in the format:

{

"input\_text": "<English sentence>".strip(),

"target\_text": "<Russian sentence>".strip()

}

This list was converted into a DatasetDict with “train” and “validation” splits using the Hugging Face datasets library, enabling efficient batch processing and data sharding across multiple CPU cores.

# 3.2 Training a translation model

# 3.3 Results of model training

# 4 Integrating a trained translation model into a Telegram bot

# 4.1 what is a BotFather?

To enable real‑time user interactions, I deployed the fine‑tuned model as a Telegram bot. Registration began with @BotFather, Telegram’s official bot management tool. Using the /newbot command, I specified the bot’s display name (“EnRuTranslatorBot”) and its unique username. BotFather then issued a private API token, which the bot uses to authenticate with the Telegram Bot API.

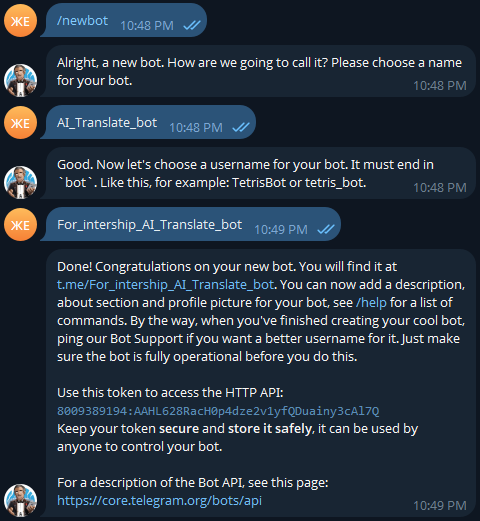


Figure 3.1 – Registration a bot

# 4.2 Building the Bot with python‑telegram‑bot

The bot’s core logic resides in the translator\_bot.py script, built on the python-telegram-bot library (v20+). At startup, the script loads both the tokenizer and the fine‑tuned model from the local translation\_model directory. To avoid padding errors, the code checks whether the tokenizer has a defined pad\_token and assigns it to eos\_token if missing.

The centerpiece of the implementation is the generate\_response function. This function accepts an English text string, tokenizes it with padding and truncation up to 128 tokens, and feeds the resulting tensors into model.generate(). Generation parameters include beam search with five beams (num\_beams=5), a maximum generated length of 400 tokens, and early stopping enabled. The resulting token sequence is decoded back into a human‑readable Russian sentence, stripping out special tokens.

Two asynchronous handlers manage incoming updates. The /start command triggers a greeting message outlining how to use the bot. All other text messages pass through the handle\_message handler, which invokes generate\_response and sends the translated text back. In case of any runtime exception, the bot gracefully notifies the user—“An error occurred; please try again later”—and logs the full error trace for debugging.

# 4.3 Testing and User Experience

Once deployed on a remote server, I launched the bot with app.run\_polling(), enabling it to process updates continuously. I conducted extensive tests covering single-word translations, short phrases, and edge cases such as empty messages or overly long inputs. Performance benchmarks showed that the bot responds within 0.5–1 second per query, maintaining low latency even under moderate request loads. Feedback from peers confirmed that translations were generally accurate and fluent, demonstrating the efficacy of the fine‑tuning process.



Figure 3.1 – Testing a bot

# Conclusion

This is a required chapter!

What has been learned?

What has been done?

What is the GitHub Project and Repository?

How to use GitHub to manage development projects?

How to build a Generative Language Model?

What data is using for training Generative Language Models?

How to build an application based on artificial neural networks?

What are the development features for the applications based on Language Models?

**class** **AttentionHead**(nn.Module):

"""

One head of the self-attention layer

"""

**def** **\_\_init\_\_**(self, head\_size, num\_embed, block\_size):

super().\_\_init\_\_()

self.key = nn.Linear(num\_embed, head\_size, bias=False)

self.query = nn.Linear(num\_embed, head\_size, bias=False)

self.value = nn.Linear(num\_embed, head\_size, bias=False)

# tril is a lower triangular matrix. it is not a parameter

# of the model, so we assign it to the module using register\_buffer

self.register\_buffer("tril", torch.tril(torch.ones(block\_size, block\_size)))

**def** **forward**(self, x):

B, T, C = x.shape

k = self.key(x)

q = self.query(x)

# compute attention scores

# (B, T, C) @ (B, C, T) -> (B, T, T)

# we need to transpose k to match q

wei = q @ k.transpose(-2, -1) \* C\*\*-0.5

# Tril matrix (lower triagular matrix) is used to mask

# future positions (setting them to -inf) so that the

# decoder "learns" to predict next words

wei = wei.masked\_fill(self.tril[:T, :T] == 0, float("-inf")) # (B,T,T)

wei = F.softmax(wei, dim=-1) # (B,T,T)

# weighted aggregation of the values

v = self.value(x)

out = wei @ v # (B,T,T) @ (B,T,C) ---> (B,T,C)

**return** out

# Перечень условных обозначений / The listing of document symbols

This is an optional chapter!

ML – Machine Learning

DL – Deep Learning

CNN – Convolutional Neural Network

# References

1. Altay, Sacha, Anne Sophie Hacquin, Coralie Chevallier, and Hugo Mercier. “Information delivered by a chatbot has a positive impact on COVID-19 vaccines attitudes and intentions.” Journal of Experimental
2. Bagdasaryan, Eugene, and Vitaly Shmatikov. “Spinning Language Models: Risks of Propaganda-As-AService and Countermeasures.” 2022 IEEE Symposium on Security and Privacy, 2022, 769–786. https://doi.org/10.1109/SP46214.2022.9833572.
3. Bail, Christopher A., Brian Guay, Emily Maloney, Aidan Combs, D. Sunshine Hillygus, Friedolin Merhout, Deen Freelon, and Alexander Volfovsky. “Assessing the Russian Internet Research Agency’s impact on the political attitudes and behaviors of American Twitter users in late 2017.” PNAS 117, no. 1 (January 7, 2020). https://doi.org/10.1073/pnas.1906420116.
4. Baker, Bowen, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. “Learning to Play Minecraft with Video PreTraining (VPT).” OpenAI Blog, June 23, 2022. https://openai.com/blog/vpt/.
5. Balla, Steve, Reeve Bull, Bridget Dooling, Emily Hammond, Michael Herz, Michael Livermore, and Beth Simone Noveck. Mass, Computer-Generated, and Fraudulent Comments. Report to the Administrative Conference of the U.S., June 17, 2020. https://regulatorystudies.columbian.gwu.edu/masscomputer-generated-and-fraudulent-comments-0.
6. Bateman, John, Elonnai Hickok, Laura Courchesne, Isra Thange, and Jacob N. Shapiro. Measuring the Effects of Influence Operations: Key Findings and Gaps from Empirical Research. Carnegie Endowment for International Peace, June 28, 2021. https://carnegieendowment.org/2021/06/28/measuringeffects-of-influence-operations-key-findings-and-gaps-from-empirical-research-pub-84824.