



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

**ASSISTANCE IN CREATING MEDICAL REPORTS USING
LARGE PRETRAINED LANGUAGE MODELS**

ASISTENCE PŘI TVORBĚ LÉKAŘSKÝCH ZPRÁV POMOCÍ VELKÝCH PŘEDTRÉNOVANÝCH JAZYKOVÝCH
MODELŮ

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Bc. PATRIK PRICL

SUPERVISOR

VEDOUCÍ PRÁCE

RNDr. MAREK RYCHLÝ, Ph.D.

BRNO 2024

Master's Thesis Assignment



162623

Institut: Department of Information Systems (DIFS)
Student: **Pričl Patrik, Bc.**
Programme: Information Technology and Artificial Intelligence
Specialization: Bioinformatics and Biocomputing
Title: **Assistance in Creating Medical Reports using Large Pretrained Language Models**
Category: Artificial Intelligence
Academic year: 2023/24

Assignment:

1. Get acquainted with the concept and possible usage of large pretrained language models, explore and compare different types of these models (LLaMA, Alpaca). Familiarise yourself with the format and structure of medical reports, analyse available datasets.
2. Propose a method for utilising large pretrained language models for completing new and correcting existing text in the process of creation of medical reports. Select appropriate types and settings of the models, training procedures, and training datasets. Also, experiment with already available pretrained models.
3. Perform training of selected language models according to the previous point using a suitable dataset of medical reports. Choose an appropriate method for measuring the usability of the trained models for the given purpose and evaluate them.
4. Implement a tool for demonstrating the completion of new and correction of existing text in the creation of medical reports using the trained models.
5. Test the entire solution, evaluate the results, and discuss them.

Literature:

- ZHAO, Wayne Xin, et al. A survey of large language models. arXiv preprint arXiv:2303.18223, 2023. Available at: <https://arxiv.org/abs/2303.18223>
- YUNXIANG, Li, et al. Chatdoctor: A medical chat model fine-tuned on llama model using medical domain knowledge. arXiv preprint arXiv:2303.14070, 2023. Available at: <https://arxiv.org/abs/2303.14070>

Detailed formal requirements can be found at <https://www.fit.vut.cz/study/theses/>

Supervisor: **Rychlý Marek, RNDr., Ph.D.**
Head of Department: Kolář Dušan, doc. Dr. Ing.
Beginning of work: 1.11.2023
Submission deadline: 31.7.2024
Approval date: 8.7.2024

Abstract

The thesis deals with the use of pre-trained language models for summarizing medical documentation in the form of dismissal reports. To create a functional model, the LLaMA architecture is used. Already existing models of the given architecture were used. The aim of the experiments was to find a model that can generate part of the documentation in the Czech language. The selected model was fine-tuned using the LoRA method. Subsequently, the inference parameters were experimented with to find the most suitable ones so that the model generates the most correct output. Web-client and server applications were also created as part of the thesis, the purpose of which is to demonstrate the functionality of the fine-tuned model.

Abstrakt

Práca sa zaoberá využitím predtrénovaných jazykových modelov na sumarizáciu zdravotnej dokumentácie do formy prepúšťacích správ. Na vytvorenie funkčného modelu bola využitá architektúra LLaMA a experimentovanie už s existujúcimi modelmi danej architektúry. Cieľom experimentov bolo nájsť model, ktorý zvládne generovať časť dokumentácie v českom jazyku. Vybraný model sa finetunoval pomocou metódy LoRA. Následne sa experimentovalo s inference parametrami na nájdenie tých najvhodnejších, aby model generoval čo najkoreknejší výstup. Taktiež bola v rámci práce vytvorená webová a server aplikácia, ktorých účelom je demonštrovať funkčnosť fine-tunovaného modelu.

Keywords

NLP, Text sumarization, Artificial intelligence, Large language models, Transformers, Peft, LoRA, LLaMA, BERT, GPT, ChatGPT-3.5, Medical documents, Flask, Python, fine-tuning

Klíčové slová

Spracovanie prirodzeného jazyka, Sumarizácia textu, Umelá Inteligencia, Predtrénované jazykové modely, Transformers, Peft, LoRA, LLaMA, BERT, GPT, ChatGPT-3.5, Zdravotnícka dokumentácia, Flask, Python, fine-tuning

Reference

PRICL, Patrik. *Assistance in Creating Medical Reports using Large Pretrained Language Models*. Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor RNDr. Marek Rychlý, Ph.D.

Assistance in Creating Medical Reports using Large Pretrained Language Models

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of RNDr. MAREK RYCHLÝ, Ph.D. The supplementary information was provided by company STAPRO s.r.o. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....

Patrik Priel
July 31, 2024

Acknowledgements

Contents

1	Artificial Intelligence	6
1.1	Symbolic Artificial Intelligence	8
1.1.1	Rule-based Expert System	8
1.2	Machine Learning	9
1.2.1	Decision Tree and Random Forest	10
1.2.2	Naive Bayes Classifier	11
1.3	Artificial Neural Network	12
1.3.1	Human Neuron	12
1.3.2	Perceptron	13
1.3.3	Basis function	13
1.3.4	Activation function	14
1.3.5	Layers	18
1.3.6	Model Training	20
1.3.7	Artificial Neural Networks for unstructured data	22
2	Large Pre-trained Language Models	25
2.1	Tokens	25
2.1.1	Vocabulary	25
2.1.2	Tokenization	25
2.1.3	Token embeddings	28
2.1.4	Special tokens	29
2.1.5	Prompt	30
2.2	Large Pre-trained Language Model's Architectures	30
2.2.1	Attention mechanism	31
2.2.2	BERT	33
2.2.3	GPT	34
2.2.4	LLaMA	35
2.3	Learning	36
2.4	Inference and evaluation	37
2.4.1	Inference	38
2.4.2	Evaluation techniques	38
3	Assistant for Creating Medical Reports	40
3.1	Dataset	40
3.1.1	Entities	40
3.1.2	Dismissal reports	41
3.1.3	Data for Supervised fine-tuning	42

4	Implementation and testing	44
4.1	Programming Languages, Libraries and Technologies	44
4.1.1	MySQL	45
4.1.2	Python	45
4.1.3	Transformers and PEFT libraries	45
4.1.4	Flask	45
4.1.5	Javascript	46
4.1.6	React.js	46
4.2	Data preparation	46
4.3	Suitable model	47
4.3.1	Dataset	49
4.3.2	Prompt	50
4.3.3	Inference and testing	54
4.4	Fine-tuning of selected model	58
4.4.1	Prompt	59
4.4.2	Dataset improvements	60
4.4.3	Experiments with LoRA training parameters	64
4.5	Inference of fine-tuned model	67
4.5.1	Experiments with Inference parameters	67
4.5.2	Demonstration tool	73
	Bibliography	77

List of Figures

1.1	Venn diagram of Artificial Intelligence’s subsets mentioned in this thesis. .	8
1.2	Design of Rule-based expert system. Image taken from [2].	9
1.3	Diagram of Decision tree defining research designs. Image taken from [26] .	11
1.4	Visual description of a human neuron. Image taken from [7].	13
1.5	Visualization of a perceptron. Image taken from [4].	14
1.6	Graph of Binary step function.	15
1.7	Graph of Bipolar step function.	16
1.8	Graph of Identity.	16
1.9	Graph of Sigmoid function.	17
1.10	Graph of ReLU Activation Function.	18
1.11	Visualization of individual types of the neural network based on connection. Image taken from [5].	19
1.12	Progress of back-propagation algorithm. Image taken from [15].	21
1.13	Progress of finding minimum loss. Image taken from [15].	21
1.14	Visualization of CNN architecture. Image taken from [24].	23
1.15	Process of prediction sequence of new words by Recurrent neural networks . Image taken from [22].	23
2.1	Tokenization of the sentence written in the English language.	27
2.2	Tokenization of the sentence written in the Czech language.	27
2.3	Word2vec predicts the words in the neighborhood of a central word by logistic classifier L. Image taken from [22].	28
2.4	Token embeddings for Transformers, like BERT or GPT, contains position information duo to parallel computations of Transformers. Image taken from [22].	29
2.5	Computation of a contextual embedding for a single token “mouse” by at- tention mechanism. Image taken from [22].	31
2.6	Visual representation of multi-head attention. Image taken from [22]. . . .	32
2.7	Scheme of stacking multiple transformer layers. Image taken from [22]. . . .	33
2.8	Transformer model predicts the next token in token’s sequence. Image taken from [22].	34
2.9	Comparison between word prediction by BERT and text generation by GPT. Image taken from [22].	35
2.10	Difference between types of attention. Image taken from [3].	35
2.11	LoRA principle using initial pre-trained weights and two much smaller ma- trices. Image taken from [12].	37
3.1	Entity relationship diagram of given dataset.	41
3.2	Example of input for model.	43

3.3	Example of label for model for input in Figure 3.2.	43
4.1	Simple schema of fine-tuning process of large language model.	44
4.2	Prompt without system prompt.	50
4.3	Prompt with system prompt.	51
4.4	First definition of instruction variable.	51
4.5	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	52
4.6	Second definition of instruction variable.	52
4.7	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	53
4.8	Final definition of instruction variable.	53
4.9	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	53
4.10	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	55
4.11	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	57
4.12	Prompt for learning process of model.	60
4.13	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	61
4.14	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	63
4.15	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	64
4.16	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	65
4.17	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	66
4.18	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	67
4.19	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	68
4.20	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	69
4.21	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	70
4.22	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	71
4.23	Output from model for input in Figure 3.2, which has label in Figure 3.3. .	71
4.24	Output from ChatGPT 3.5 for input 3.2, which has label as 3.3.	72
4.25	Sequence diagram of API communication between client-side application and server-side application.	73
4.26	Client's application.	74

Introduction

The expansion of artificial intelligence within the framework of natural language processing leads to its more frequent use in various work areas. By using artificial intelligence, people can be relieved of redundant administration by having artificial intelligence manages it by itself. Medicine is one of these sectors.

In addition to treating patients, doctors must also write the patient's medical documentation. By relieving doctors of this repetitive administrative tasks, doctors can focus more on their main work, which is treating patients. The aim of this thesis is to find a way to assist doctors in the creation of medical documentation and make the administrative part of their work easier with the help of artificial intelligence methods such as Large Pre-trained Language models.

In the chapter 1 of this thesis, artificial intelligence as whole will be discussed and which of the individual artificial intelligence methods can work with text to help with the creation of medical documentation.

In the chapter 2, Large Pre-trained Language models are introduced as one of the Artificial Intelligence methods used in natural language processing, which will also be used to find a solution to the problem of this thesis. Mainly, the architecture and learning process of Large Pre-trained Language models will be discussed. Additionally, the chapter introduces methods for evaluating the model's performance.

The available dataset and the structure of medical documentation is shown in the chapter 3. In that chapter, it is also shown how to modify the available dataset to be used for training of Large Pre-trained Language models. Additionally, the chapter provides a description of the use of techniques for optimal model performance.

In the chapter 4, implementation of dataset creation as well as training process of Large Pre-trained language model are described and the chapter also provides a description of the use of techniques for optimal model performance. Evaluation of the individual steps of this process and also the evaluation of the success of the model to generate medical documentation is also included in this chapter. Description of demonstration tool is also part of this chapter.

Chapter 1

Artificial Intelligence

A wide range of programs fall into the category of artificial intelligence. It can be an program that classified a thing to 2 categories or an assistant, which can drive a car from some city to another city.

Program, categorized as artificial intelligence, is called **agent**. An agent is just something that acts. Computer agents are expected to do: operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals. [27]

Based on this, we divide artificial intelligence into [14]:

- **Narrow Artificial Intelligence:** Also called Weak Artificial Intelligence. It can be trained to perform a single or narrow task, often far faster and better than a human mind can. However, it cannot perform tasks other than the one it was trained for. Even OpenAI's ChatGPT is considered a form of Narrow Artificial Intelligence, because it is limited to the single task of text-based chat.
- **Generative Artificial Intelligence:** Also known as Strong AI. This type of AI can solve more problems and truly understands what is happening. There may even be emotions and creativity. Only few companies decide to develop this type of AI and are still working on it. This type of Artificial Intelligence is still theoretical.
- **Super Artificial Intelligence:** Artificial superintelligence is strictly theoretical. If ever realized, Super Artificial Intelligence would think, reason, learn, make judgements and possess cognitive abilities that surpass those of human beings. The applications possessing Super Artificial Intelligence capabilities will have evolved beyond the point of understanding human sentiments and experiences to feel emotions, have needs and possess beliefs and desires of their own.

Artificial Intelligence is also divided into four categories based on functionalities [14]:

- **Reactive Machine Artificial Intelligence:** Reactive machines are Artificial Intelligence systems with no memory and are designed to perform a very specific task. Since they can't recollect previous outcomes or decisions, they only work with presently available data. Reactive Artificial Intelligence stems from statistical math and can analyze vast amounts of data to produce a seemingly intelligence output.
- **Limited Memory Artificial Intelligence:** This form of Artificial Intelligence can recall past events and outcomes and monitor specific objects or situations over time. Limited Memory Artificial Intelligence can use past- and present-moment data to decide on a course of action most likely to help achieve a desired outcome. However, while Limited Memory Artificial Intelligence can use past data for a specific amount of time, it can't retain that data in a library of past experiences to use over a long-term period. As it's trained on more data over time, Limited Memory Artificial Intelligence can improve in performance.
- **Theory of Mind Artificial Intelligence:** Theory of Mind AI is a functional class of AI that falls underneath the General AI. Though an unrealized form of AI today, AI with Theory of Mind functionality would understand the thoughts and emotions of other entities. This understanding can affect how the AI interacts with those around them. In theory, this would allow the AI to simulate human-like relationships. Because Theory of Mind AI could infer human motives and reasoning, it would personalize its interactions with individuals based on their unique emotional needs and intentions. Theory of Mind AI would also be able to understand and contextualize artwork and essays, which today's generative AI tools are unable to do.
- **Self-Aware Artificial Intelligence:** Self-Aware AI is a kind of functional AI class for applications that would possess super AI capabilities. Like theory of mind AI, Self-Aware AI is strictly theoretical. If ever achieved, it would have the ability to understand its own internal conditions and traits along with human emotions and thoughts. It would also have its own set of emotions, needs and beliefs.

In this thesis, attention will be directed only to Narrow artificial intelligence, as the other types currently operate only at a theoretical level. Narrow Artificial Intelligence includes the categories Reactive Machine AI and Limited Memory AI, so attention will be also focused only on these two categories based on functionality.

The Narrow Artificial Intelligence can be also divided by the approach how to solve the given problem. In the thesis, the approach using Symbolic Artificial Intelligence and machine learning will be described.

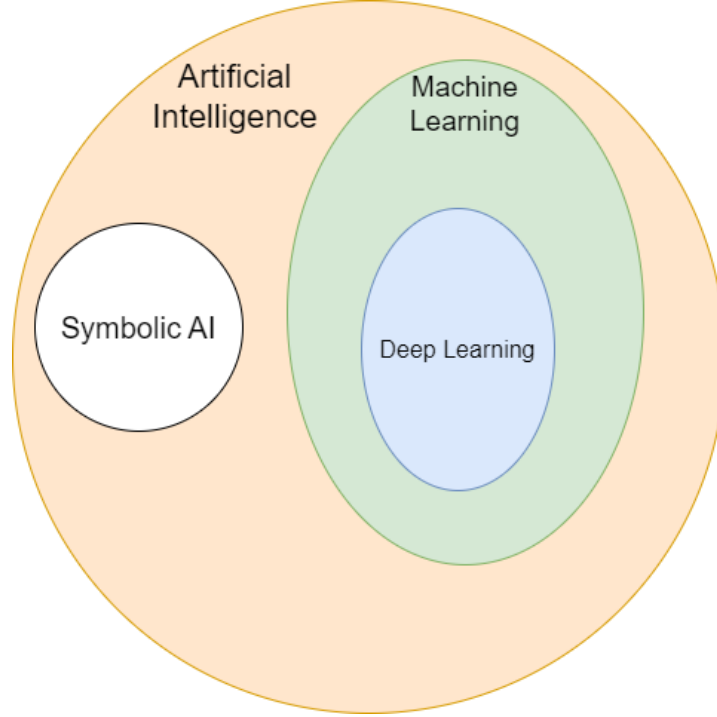


Figure 1.1: Venn diagram of Artificial Intelligence’s subsets mentioned in this thesis.

1.1 Symbolic Artificial Intelligence

Symbolic artificial intelligence, a fundamental approach in artificial intelligence, intricately involves the explicit modeling of intelligent systems with a strong focus on clarity and unambiguous representation. This approach requires an in-depth analysis and understanding of the problem. The acquired knowledge is transformed into symbolic or abstract forms.

Symbolic representations can take the form of graphs, logic formulas, and rules [8]. They are used to preserve the characteristics of individual knowledge. The solution is created by traversing complex relationships and rules that provide the basis for sophisticated problem solving.

These symbolic representations must be defined by human experts. The expert describes how the system perceives and acts in its environment. Due to the need for high expertise in the area of the given problem and precisely defined knowledge for effective functioning, this approach places higher demands on human resources than the machine learning approach. Nevertheless, this approach is often preferred due to the clear definition of how the program should behave in a given situation and the simple correction in case of unexpected behavior.

1.1.1 Rule-based Expert System

One of the methods that is used nowadays is Rule-Based Expert Systems. Its use human expert knowledge to solve real-world problems that normally would require human intelligence. Expert knowledge is represented in the form of rules with needed data saved in computer’s memory.

Depending upon the problem requirement, these rules and data can be recalled to solve problems. Rule-based expert systems have played an important role in modern intelligent

systems and their applications in strategic goal setting, planning, design, scheduling, fault monitoring, diagnosis.[2]

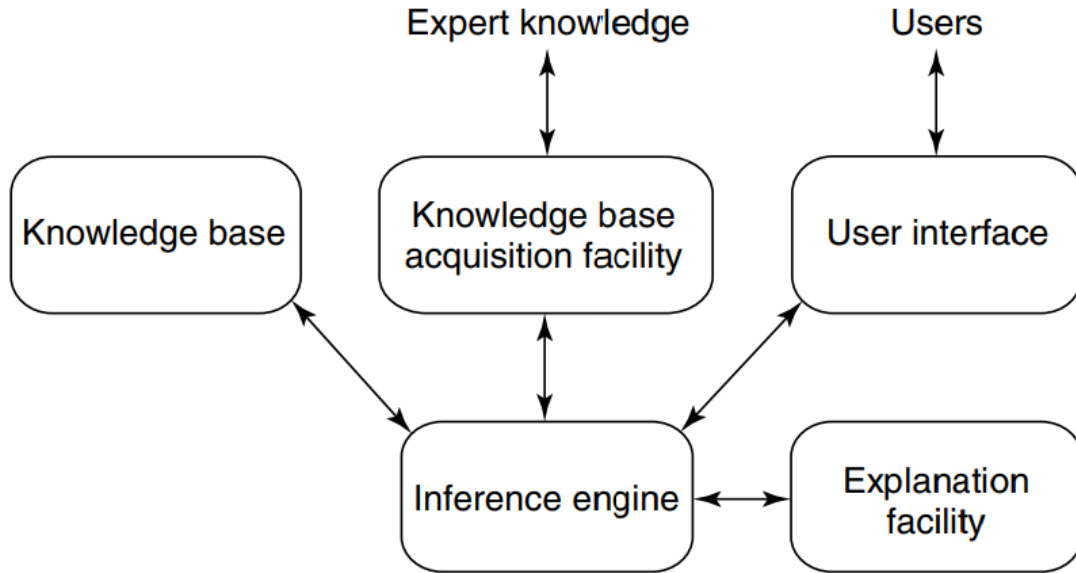


Figure 1.2: Design of Rule-based expert system. Image taken from [2].

The basic components of an expert system are illustrated in Figure 1.2.

The knowledge base stores all relevant information, data, rules, cases, and relationships used by the expert system. A knowledge base can combine the knowledge of multiple human experts. The purpose of the inference engine is to seek information and relationships from the knowledge base and to provide answers, predictions, and suggestions in the way a human expert would. The explanation facility allows a user to understand how the expert system arrived at certain results. The purpose of the user interface is to ease use of the expert system for developers, users, and administrators. [2]

This method cannot directly process text. However, it can be combined with methods that extract information from unstructured texts, recognize entities, and their relationships. These extracted pieces of information serve as inputs to the expert system. Subsequently, when the expert system provides an answer, a method is also needed to generate this answer in a user-friendly text format. Such a combination of algorithms can be particularly effective in the healthcare field, where it is essential to accurately diagnose and determine the appropriate treatment, which may be specific to each patient.

1.2 Machine Learning

Machine Learning, unlike Symbolic Artificial Intelligence, does not need an expert with knowledge in the problem's topic. What it needs, however, is data of the given problem, on which it will „learn“ the solution. This approach is really inspired by the human ability to learn.

Learning

In machine learning are 3 types that defines how can agent learn the solution.

- In *unsupervised learning* the agent learns patterns in the input even though no explicit feedback is supplied. The most common unsupervised learning task is clustering: detecting potentially useful clusters of input examples.
- In *reinforcement learning*, the agent learns from a series of reinforcements—rewards or punishments. If the agent gets a reward, it knows that is a good way to the solution. On the other hand, if the agent is punished, it knows that is a bad way to the solution.
- In *supervised learning*, the agent observes some example input–output pairs and learns a function that maps from input to output.

Process of learning, that turn machine learning models with bad results to models, which give useful results, contains these steps:

- First step is **Data Preparation**: Data should be in the best condition. Flaws can cause a significantly worse result. The cleaned and modified data are divided into training, validation and test datasets. The training dataset is used to train the model, the validation dataset provides an evaluation of the model’s fit during training, and the test dataset is required for the final evaluation of the model.
- Second step is **Model Training** : After randomly initializing the model parameters, input data is fed forward through the model, producing output. The difference between the produced output and the desired value is calculated, and through iterations, this difference is minimized by adjusting the model parameters.
- Third step is **Evaluation and Testing**: Using the validation dataset, the model is evaluated with metrics such as accuracy, precision, and recall. This type of evaluation is performed after every iteration during the training phase. Once the training is done, the testing dataset is used to evaluate metrics and assess the ability of the model to solve the given problem.

One of the main problems during model training is **overfitting**. It means that the model has learned the data in great detail, including any noise or flaws captured in the data. As a result, the model fails when it encounters new, unseen data.

1.2.1 Decision Tree and Random Forest

This model is mostly used for classification, but it also manages the regression or the anomaly detection. Model has tree structure, where leafs represent final class as it is shown in the Figure 1.3. A tuple with quantitative and qualitative attributes is used as input. The classification process starts from the root node and continues to other non-leaf nodes, where the value of attributes is evaluated. Branches represent the outcomes of this evaluation. Supervised learning is selected in order for the model to be able to assign tuples to the correct class.

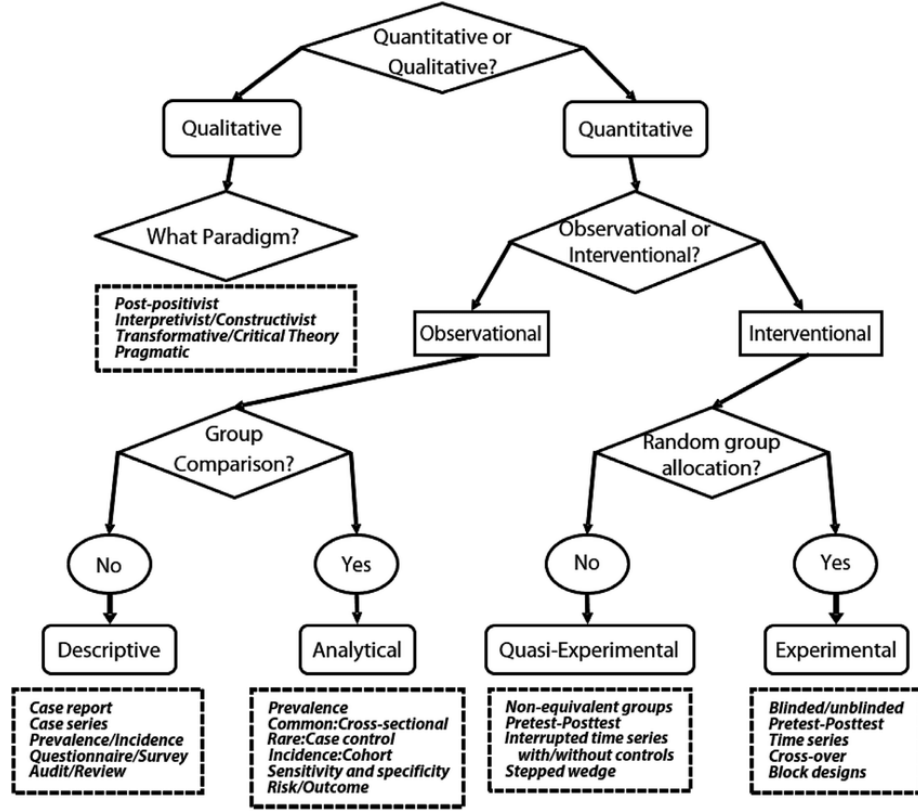


Figure 1.3: Diagram of Decision tree defining research designs. Image taken from [26]

A random forest is a model using multiple decision trees. Each tree is created based on randomly chosen subset of data. A random forest is good for classification tasks. Each of the individual trees vote for final class. The selected class is the class with the most votes. One of the advantages of this model is its resistance to overfitting.

Decision trees can be a powerful tool for extracting and classifying information from text. This includes tokenization, which is the process of splitting text into words or phrases. Feature extraction is conducted by converting the preprocessed text into a numerical representation. The data is split at each node based on feature values to maximize the separation between classes or minimize variance. High dimensionality is a significant challenge when dealing with text data, as it often results in a large number of features, making the decision tree complex and prone to overfitting. The advantages of decision trees is their interpretability. Visualizing the tree can provide insights into how decisions are made based on textual features.

1.2.2 Naive Bayes Classifier

The Naive Bayes Classifier is a machine learning model based on probability. The foundation is Bayes's Theorem. This theorem uses conditional and marginal probabilities to calculate the probability of a certain class. The theorem can be described by the mathematical equation 1.1, where $P(X)$ is probability of event X and $P(X|Y)$ is conditional probability of event X if event Y already happened.

$$P(X|Y) = \frac{P(Y|X) P(X)}{P(Y)} \quad (1.1)$$

For the Naive Bayes classifier, the mathematical equation 1.1 is modified into the equation 1.2. C_k represents the k-th class and X_i represents the i-th attribute of the input tuple. The result is class with higher probability.

$$Y = \underset{k=1,2,\dots,m}{\operatorname{argmax}} P(C_k) \prod_{i=1}^n P(X_i|C_k) \quad (1.2)$$

The reason why this model is called „Naive“ is that the dependence between individual features is not considered.

This method is also used in natural language processing, particularly for classification tasks such as spam detection. Initially, the text undergoes tokenization and feature extraction, similar to the decision tree method. However, the outcome is the probability of assigning the text to a specific class. Naive Bayes assumes feature independence, simplifying the model but potentially not capturing the true nature of text data. Despite this, Naive Bayes is more computationally efficient and scalable for large datasets.

1.3 Artificial Neural Network

Artificial Neural Networks are a cornerstone of modern machine learning, inspired by the biological neural networks of the human brain. They are designed to recognize patterns and learn from data, making them highly effective for tasks such as image and speech recognition, natural language processing, and predictive modeling. By imitating the way neurons and their interconnections function, Artificial Neural Network enable machines to perform complex tasks and improve their performance over time through learning. Large Pre-trained Language models are built upon Artificial Neural Networks, showcasing the current state-of-the-art in natural language processing.

1.3.1 Human Neuron

A cell that is a basic part of the human nervous system. A neuron consists of a nucleus, which is also called **Soma**. Soma is round or oval, with a prominent nucleolus [29]. Next, there are filaments extruding from Soma and these filaments are used for communication with other neuron cells. The layer between communicating filaments or filament and another type of cell is called **Synapse**. These filaments are divided into [29]:

- Dendrites: Serve to receive input information. A neuron has usually a larger number of dendrites. They tend to be shorter but richly branched.
- Axon: Used to send information from the body of the neuron. It is usually one, but at the end it can be considerably branched. In most cases, the axon is wrapped along its entire length by a myelin sheath, which plays a significant role in the transmission of impulses. The longer the nerve fiber and the thicker the myelin sheath, the faster it conducts the impulse.

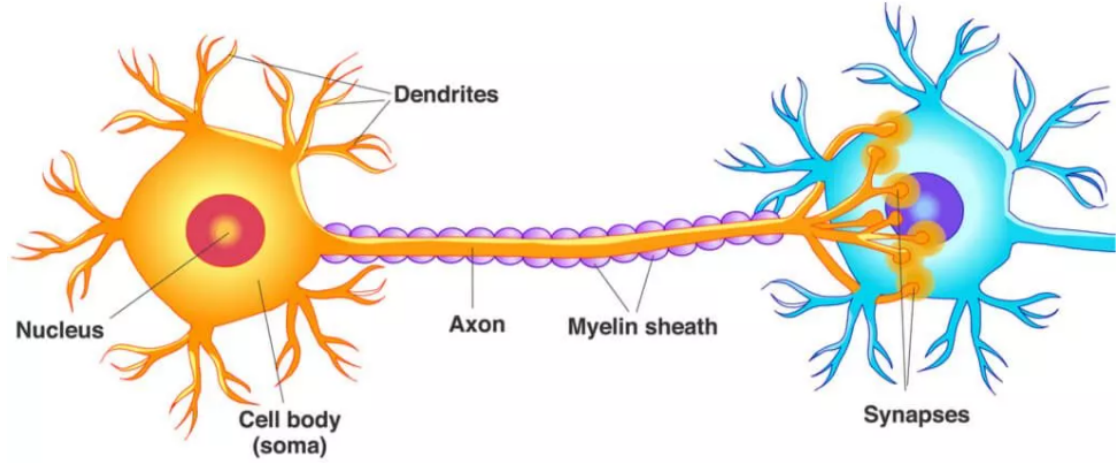


Figure 1.4: Visual description of a human neuron. Image taken from [7].

The main function of a neuron is to receive, process and send a signal. This signal is in electrical form within the neuron and between neurons it is in chemical form. Whether the signal is sent further when one or more Dendrites are activated depends on the cell itself.

1.3.2 Perceptron

A perceptron is an artificial neuron that mimics human neurons in a very simple way. It has multiple inputs, but only one output. Artificial neuron also consist of weights, a basis function and an activation function.

- **Weights:** The numerical values, which indicate how much the given input will influence output's value. Each input has its own weight.
- **Basis function:** It processes the input values together with the corresponding weights. The result serves as input to the activation function.
- **The activation function** sets the output value, and this function is selected based on the task to be solved.

The Figure 1.5 shows the perceptron with Linear basis function graphically and the similarity with the human neuron from the Figure 1.4 is visible.

1.3.3 Basis function

In machine learning, two functions are defined as basis functions: the Linear basis function and the Radial basis function. [27].

- **Linear basis function:** For most neural networks, perceptrons with the Linear basis function are chosen. It can be described as a linear combination of inputs and their weights. This can also be represented by the mathematical equation 1.3, where x is input, y is output of the Linear basis function, w is weight and i is number of input.

$$y = \sum_{n=0}^i (x_i w_i) \quad (1.3)$$

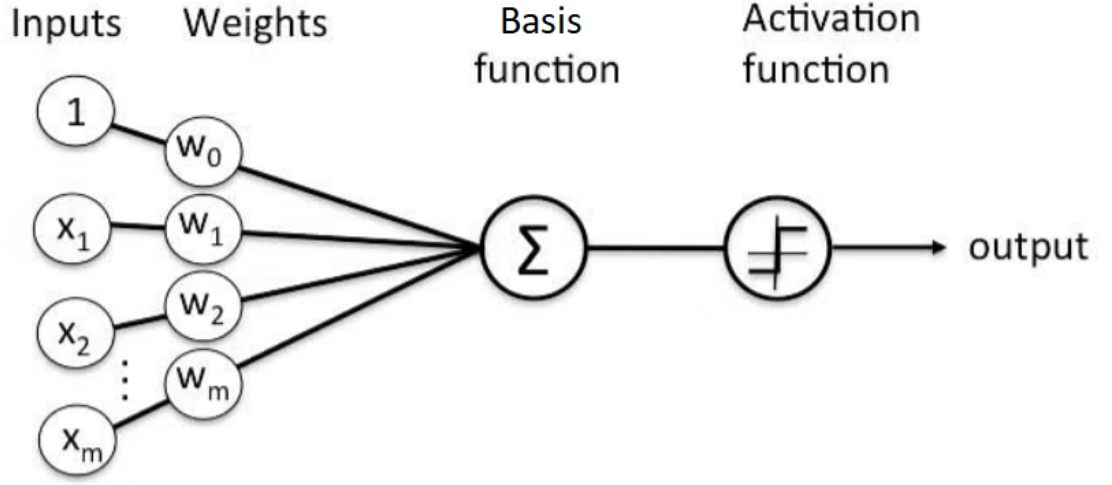


Figure 1.5: Visualization of a perceptron. Image taken from [4].

Perceptrons with the Linear basis function also typically have an input called Bias, always with the value 1. In equation 1.3, input x_0 represents the Bias. The Figure 1.5 shows the perceptron with the Linear basis function.

- **Radial basis function:** This basis function works on other principle than the Linear basis function. The value of the Radial basis function depends only on the distance from a certain point called the center[8]. The coordinates of this point are stored in the parameter weight. For the calculation of this distance, the Euclidean distance is used [30]. Its mathematical description is the Equation 1.4. In the given equation, x is vector of inputs, w is vector of inputs. n is count of all inputs to the neuron.

$$D(x, w) = \sqrt{|x_1^2 - w_1^2| + |x_2^2 - w_2^2| + \dots + |x_n^2 - w_n^2|} \quad (1.4)$$

1.3.4 Activation function

Output from a node depends on this function. As already mentioned, the choice of the activation function depends on the problem that the neural network has to solve and also on the way in which it is expected to solve it. Some activation functions can behave quite radically. Bipolar step function or Heaviside step function are typical examples of this [8].

- **Heaviside step function:** This is a very simply defined function. The function returns 1 for a positive input and 0 for a negative input. Mathematically, the function is defined as 1.5 and the development of the function on the 2D numerical axis is shown in the Figure 1.6. The function is also called as Binary step function and is used in the output layer of the neural network, which is described in Subsection 1.3.5, for binary classification tasks. One of the main disadvantage of this function is that the gradient at an output value of 0 is also 0. This means that during learning process, there is no changes in neuron's weights, basically meaning that the neural network learns nothing.

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (1.5)$$

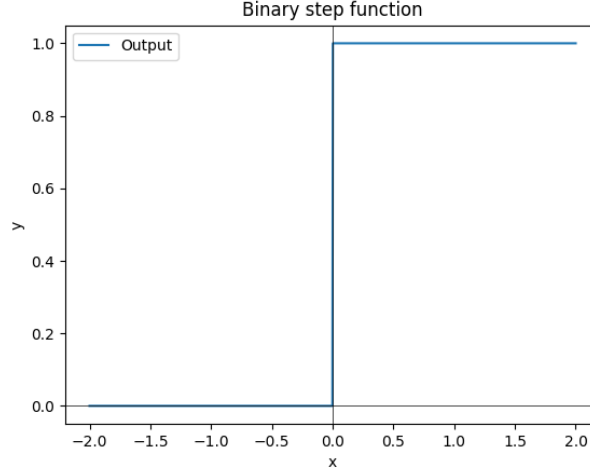


Figure 1.6: Graph of Binary step function.

- **Bipolar step function:** This function is also as simple as the Heaviside step function. The only difference between them is that the Bipolar step function returns -1 for a negative input or 0. The mathematical equation of the function is 1.6 and the graph of the function is shown on Figure 1.7. This function is used also in the output layer of the neural network for binary classification tasks. This function does not have a 0 at the output like the Heaviside step function, defined in the Subsection 1.3.4, so the neuron's weights are changed at both outputs.

$$f(x) = \begin{cases} 1, & x > 0 \\ -1, & x \leq 0 \end{cases} \quad (1.6)$$

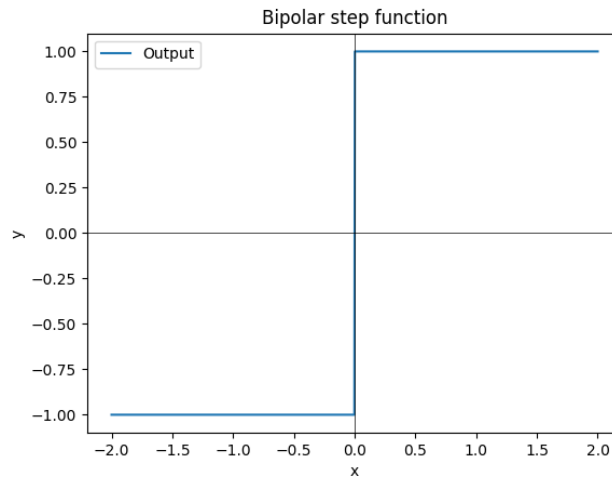


Figure 1.7: Graph of Bipolar step function.

- **Linear function:** This function reacts more „smooth“ than the Heaviside step function or Bipolar step function [8]. The Basic linear function is called Identity and the result of the function is the same as the input as seen in the Figure 1.8. These types of function are used in regression tasks, where the goal is to predict continuous values, or in dimensionality reduction to reconstruct the input data, while preserving its linear structure.

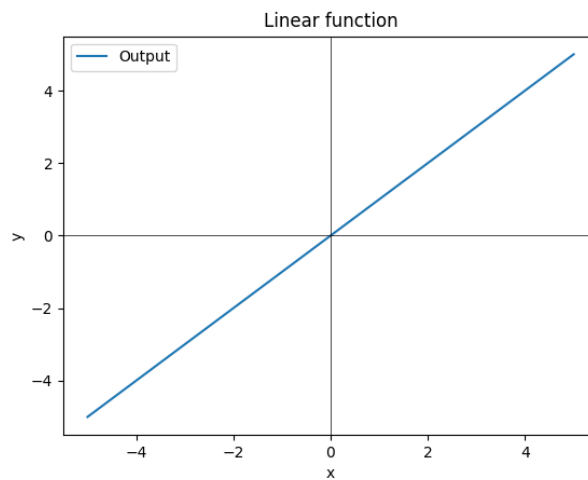


Figure 1.8: Graph of Identity.

- **Sigmoid function:** This function is continuously differentiable and a smooth S-shaped function [28]. On large negative or positive inputs, gradient of this function can be very small, which leads to vanishing gradient problem. This function can be used in hidden layers, but because of vanishing gradient problem, function is mostly used in output layer at binary classification tasks, where the output should be the probability of classifying the element in the given class. This function can be also used in hidden layers. The Sigmoid function is seen in the Figure 1.9.

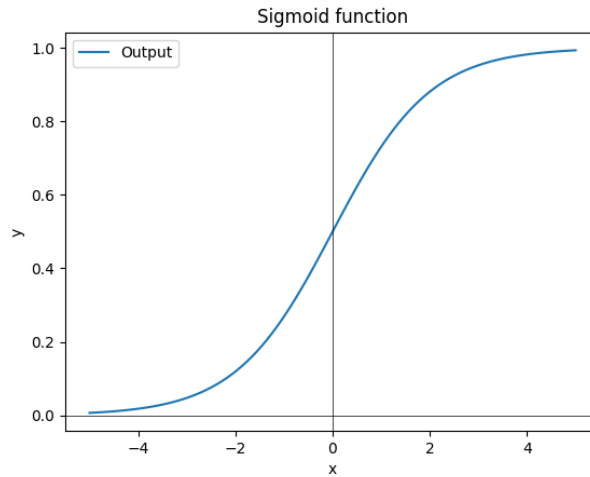


Figure 1.9: Graph of Sigmoid function.

- **ReLU Activation Function:** ReLU stands for rectified linear unit and is a non-linear activation function which is widely used in neural network [28]. The function is based on a mathematical equation 1.7 and its behavior is illustrated in the Figure 1.10. This function is used in hidden layer, described in Subsection 1.3.5, for various tasks from image classification to speech recognition.

$$f(x) = \max(0, x) . \quad (1.7)$$

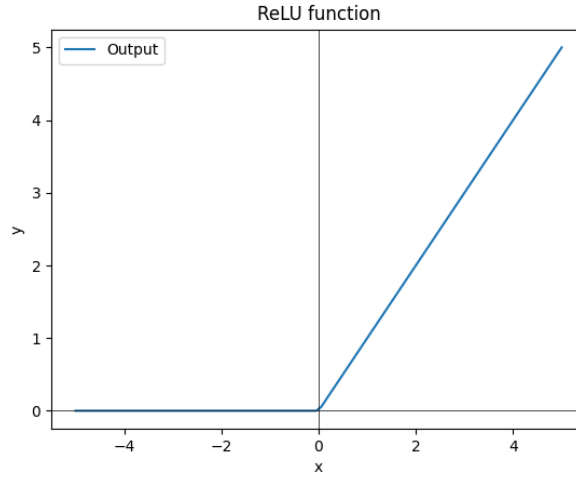


Figure 1.10: Graph of ReLU Activation Function.

- **Softmax function:** Like the sigmoid function, the result is the probability that the given sample belongs to the given class. The difference from the sigmoid function stems from its use, where the sigmoid function is used to determine the probability between two classes and the softmax function produces a probability distribution over multiple classes. Primarily used in the output layer of classification models. A mathematical equation of this function is 1.8, where z is vector, j is index of vector and K is number of classes.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad (1.8)$$

1.3.5 Layers

As the brain is made up of several interconnected neurons, the neural network is also made of multiple perceptrons arranged in layers. A perceptron can also be referred to as a node. These 3 types of layers are used in neural networks:

- **Input Layer:** The input enters the neural network through the input layer. Neurons in this layer perform no other activity than sending the input value to other neurons.
- **Hidden layer :** The layer of this level performs calculations on the input values that will lead the neural network to the correct result. Networks with more than one hidden layer are denoted as Deep Neural Networks. With more hidden layers, the network can learn to solve the given problem more successfully, but the process of training the model is all the more challenging.
- **Output layer :** It processes calculations obtained from neurons from the hidden layer and sends the result to the output.

Each layer has a different purpose and together they ensure the desired functioning. The cooperation of the layers also depends on their type of connection. There are two fundamentally distinct ways [27]:

- **Feed-forward network** : Layers are only connected in one direction. Only the given input will influence output's value. Each input has its own weight. Every node receives input from “upstream” nodes and delivers output to “downstream” nodes. There are no loops. The Feed-forward network, where the input of node is obtained from nodes that are one level of layers higher, is called **Fully-connected network**.
- **Recurrent network** : Unlike the Feed-forward network, this one feeds its outputs back into its own inputs. This means that the activation levels of the network form a dynamical system that may reach a stable state or exhibit oscillations or even chaotic behavior. Moreover, the response of the network to a given input depends on its initial state, which may depend on previous inputs. Recurrent networks can support short-term memory. This makes them more interesting as models of the brain, but also more difficult to understand.

Those two types are graphically illustrated on Figure 1.11.

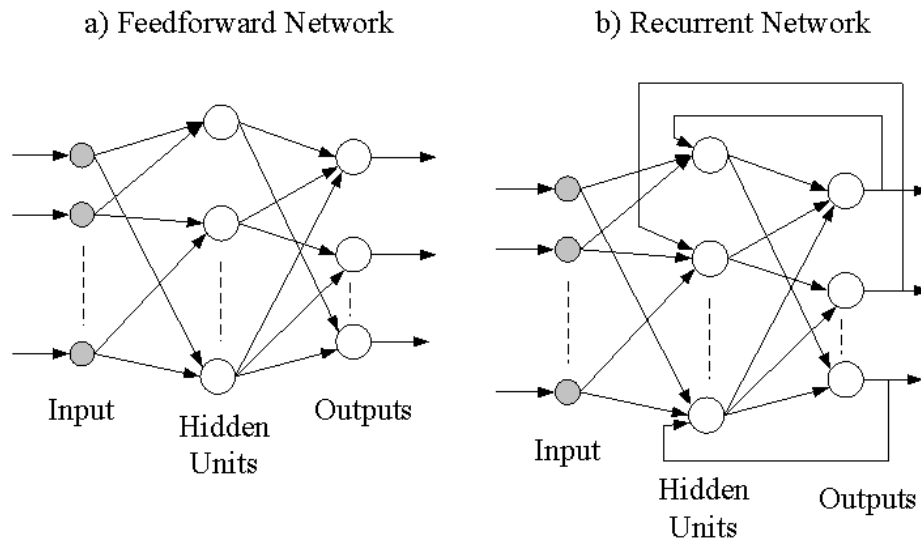


Figure 1.11: Visualization of individual types of the neural network based on connection. Image taken from [5].

1.3.6 Model Training

As mentioned in Section 1.2, first the parameters are randomly initialized, then the input is passed through the network. After all inputs from the training dataset have passed through the network, the difference between the output of the network and the expected output is calculated for each sample in the training dataset. Expected output is also called **label**. A loss function is used for these calculations.

Loss function can be an arbitrary function mapping two vectors to a scalar. This function should be bounded from below, with the minimum attained only for cases where the prediction is correct. [10]

The type of loss function is chosen based on the network architecture and the type of problem. The types of loss function are as follows [10]:

- **Hinge** : For binary classification problems, the classifier's output is a single scalar y_r and the label $y_w \in \{-1, 1\}$. The mathematical notation is shown in Equation 1.9.

$$L(y_r, y_w) = \max(0, 1 - y_r y_w) \quad (1.9)$$

- **Binary cross-entropy**: The function is used in binary classification with conditional probability outputs. Binary cross-entropy is useful when the network produces class conditional probability for a binary classification problem and It is assumed that the output layer is transformed using the sigmoid function. The loss function is defined in Equation 1.10. Label $y_w \in \{0, 1\}$ defines in which class the sample belongs to and $p(y_w)$ is the probability from the model.

$$L = -y_w \log p(y_w) - (1 - y_w) \log(1 - p(y_w)) \quad (1.10)$$

- **Categorical cross-entropy**: Similar to binary cross-entropy, the function is used when a probabilistic interpretation is required, but is used in multi-class classification. It is assumed that the output layer is transformed using the softmax function. The loss function is defined in Equation 1.11, where C is count of classes, y_w is vector of wanted probability for all classes, so y_{w_c} is wanted probability of c class and y_{w_c} is predicted probability by model for that class.

$$L = - \sum_{c=1}^C y_{r_c} \log(y_{w_c}) \quad (1.11)$$

- **Mean Square Error**: It is the average squared difference between the observed and predicted values. It is used to calculate output error in regression tasks. This is a simple function that skillfully solves „punishment“ for a very distant output from the desired output and at the same time does not focus on small deviations. The formula for the calculation is shown in Equation 1.12, where n is number of training samples in Epoch.

$$MSE(y_r, y_w) = \frac{1}{n} \sum_{i=1}^n (y_r - y_w)^2 \quad (1.12)$$

Next part of model training is calculate the gradients and adjust weights of the network by the calculated gradients.

Gradient is a vector that gives the magnitude and direction of the steepest slope [27]. One of the Loss functions is used to calculate the gradient.

Back-propagation algorithm is used to propagate gradients from the output layer through all hidden layers back to the start of the network and adjust weights in order to minimize the loss function. This is demonstrated on Figure 1.12.

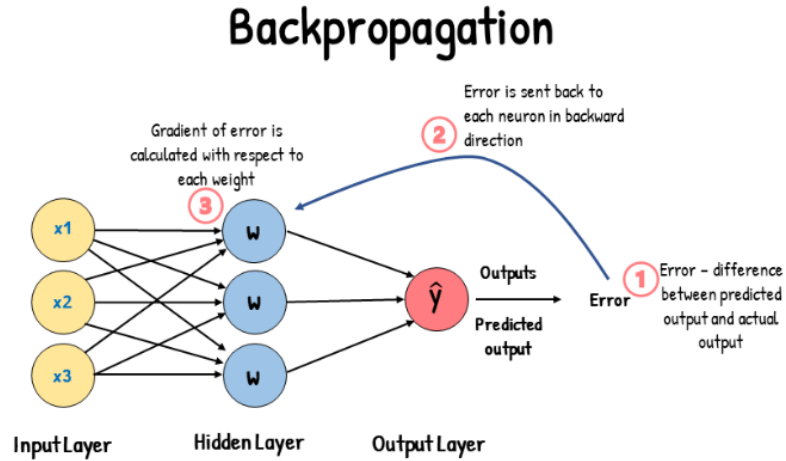


Figure 1.12: Progress of back-propagation algorithm. Image taken from [15].

The goal is to find the minimum of the loss function for all the training data by stepping down on the function surface as it is shown on Figure 1.13.

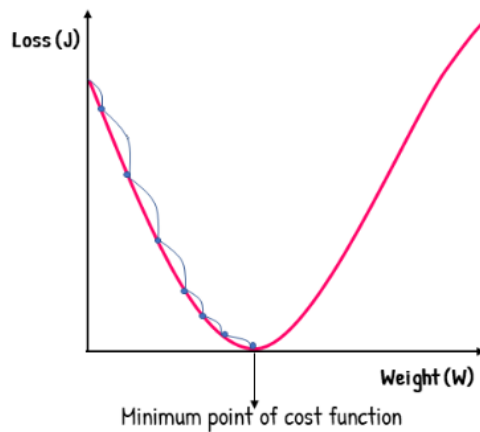


Figure 1.13: Progress of finding minimum loss. Image taken from [15].

The way the gradient changes the individual weights of neurons to reduce errors is defined by the chosen *optimization function*:

- ***Stochastic Gradient Descent***: Stochastic Gradient Descent is a general optimization algorithm. It works by repeatedly sampling a training example and computing the gradient of the error on the example with respect to the parameters. It updates the parameters in the opposite direction of the gradient of the loss function with respect to the parameters. Stochastic Gradient Descent updates the parameters after computing the gradient using a single example or a mini-batch of examples. [10]
- ***Adagrad***: The algorithm adaptively tunes learning rate for each parameter and in the process performing larger updates for rarely updated parameters. Nevertheless, Adagrad's update rule results in a vanishing (decaying) rate which forces the learning rate to monotonically decrease to a very small amount. A very small learning rate stalls the algorithm. [21]
- ***Root Mean Square Propagation (RMSProp)***: The technique maintains per-parameter learning rates and is an extension of the stochastic gradient descent algorithm, that attempts to fix the issue of vanishing (decaying) learning rates. A version of RMSProp optimization algorithm uses momentum - moving average of the squared gradients for each parameter. [21]
- ***Adam***: Adam means Adaptive Moment Estimation. It is a popular algorithm and integrates the benefits of Adagrad and RMSProp. It leverages on the moving average of past gradients to ascertain the direction of descent just as it uses the running average of past squared gradients to scale (modify) the learning rate. Adam offers a remarkable improvement on stochastic gradient descent in that it performs well in practice. It converges fast and enhances the learning speed of neural network models. [21]

1.3.7 Artificial Neural Networks for unstructured data

This types of neural networks are used to solve complex problems like text generation, speech recognition or language translation. The neural network must not only learn to solve the problem correctly, but also convert the input data into a format that it understands and can find a solution for. This category also includes the type of neural network used in Large Pre-trained Language models.

Convolutional neural network

This is neural network used mostly for solving problems include image type input's data. For process this type of data, Convolutional neural network have special types of layers on the begging of network to convert input to better format for Feed-forward network. The special types of layers are:

- ***Convolutional layer*** is supposed to apply matrix filters to the input and identify indicative local predictors in a large structure [10].
- ***Pooling layer*** resizes output matrix from the Convolutional layer to the smaller one.

These two layers are repeated at the beginning of the neural network until a vector is created that can be processed by the **Feed-forward network**, which is described in Subsection 1.3.5. This architecture is shown on Figure 1.14. Convolutional neural network is used for image classification, object detection, motion prediction. It can also be used in the processing of natural speech, where the sound stage is transformed into reasonable representations such as spectrograms and then can be processed by Convolution neural network.

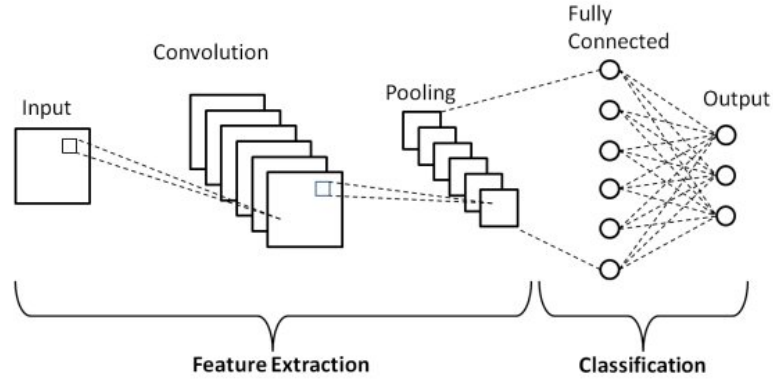


Figure 1.14: Visualization of CNN architecture. Image taken from [24].

Recurrent neural networks

The architecture of Recurrent neural network was mentioned in Subsection 1.3.5 for its use of its own output as input in the next step. Recurrent neural network, as language model, computes the likelihood of a sequence of words and predicts the next word in the sequence. Sequence of words are saved as inner state and new word is used as input to network. Input word is added to inner state and network predict next new word. Process of prediction is shown on Figure 1.15.

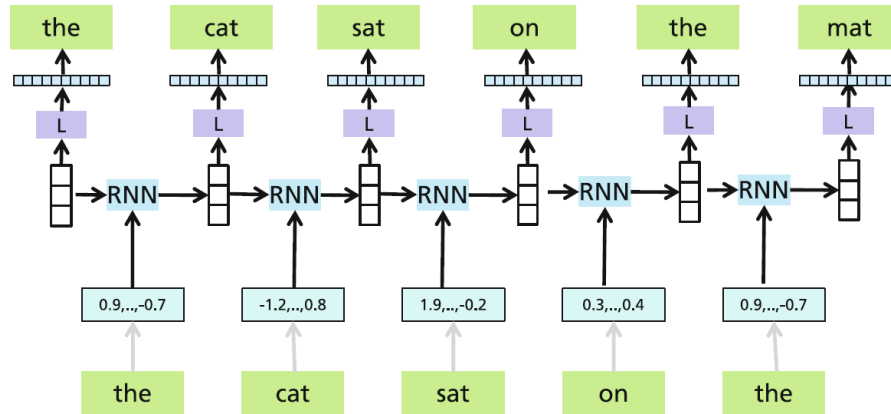


Figure 1.15: Process of prediction sequence of new words by Recurrent neural networks . Image taken from [22].

It turns out that this model has difficulties to reconstruct the relation between distant sequence elements, since gradients tend to vanish or “explode” as the sequences get longer. Therefore, new Recurrent neural network types have been developed: **Long Short-Term Memory** and **Gated Recurrent Unit**. Both of them introducing gating mechanisms that allow the model to retain information over longer sequences. [22]

Transformer

Transformer is the neural network architecture that can be used for Natural Language Processing tasks like text generation. Models based on Recurrent Neural Networks have a major limitation caused by the sequential nature of Recurrent Neural Network. The number of operations required to determine the relation between words grows with the distance between positions. The model has to store the relations between all words simultaneously in a vector, making it difficult to learn complex dependencies between distant positions. The Transformer directly computes these relations between words in parallel in one step, instead of relating distant words by a large number of computation steps. [22]

The transformer architecture is the foundation of many Large Pre-trained Language models and will be described in more detail in the following Chapter 2.

Chapter 2

Large Pre-trained Language Models

In this chapter, the architecture of Large Pre-trained Language Models, the process of converting text into tokens, and the learning process of Large Pre-trained Language Models are described. Various techniques for evaluating these models will also be discussed, some of which will be used to evaluate the models used to address the problem presented in this thesis.

2.1 Tokens

These models do not understand text representation as one long string datatype. Therefore, text need to be divide into tokens.

An essential components for this in the architectures of Large Pre-trained Language models are the encoder and decoder. The encoder processes the input text sequence and converts it into tokens, while the decoder performs the reverse operation, converting tokens back into text sequences. To accomplish this, the model utilizes a vocabulary.

2.1.1 Vocabulary

It is essentially a mapping structure where each token is associated with a unique index or identifier and the corresponding text, which a token represents. The identifier associated with the token is a reference for the model. Typically, it is represented as an integer data type and allows the model to efficiently access and manipulate tokens during training and inference.

Size of vocabulary is not infinite and depends on various factors, such as the size of the training corpus, the tokenization technique or adding own tokens, imposed on the vocabulary size. A larger vocabulary can capture a wider range of linguistic variations but may require more computational resources. [22]

2.1.2 Tokenization

Tokenization is process, where input text is divided into tokens and can be done in the following ways (For each way, there is example of tokenization of this sentence: „This is an example of tokenization.“) :

- **Tokenization based on words** divide text to separate words and each token represents one word. Example:

```
tokens = ["This", "is", "an", "example", "of", "tokenization", "."]
```

- **Tokenization based on subwords**, where token represents part of word. So for tokenization of one word is needed more than one token. Example:

```
tokens = ["This", "is", "an", "example",
          "of", "to", "##ken", "##ization", "."]
```

- **Tokenization based on characters**, where token represents smallest unit of word. Example:

```
tokens = ['T', 'h', 'i', 's', ' ',
          'i', 's', ' ',
          'a', 'n', ' ',
          'e', 'x', 'a', 'm', 'p', 'l', 'e', ' ',
          'o', 'f', ' ',
          't', 'o', 'k', 'e', 'n', 'i', 'z', 'a', 't', 'i', 'o', 'n',
          '.']
```

- **Tokenization based on n-grams**, where token represents n-number of words [22]. Example:

Bigrams (2-grams):

```
tokens = [("This", "is"), ("is", "an"),
          ("an", "example"), ("example", "of"),
          ("of", "tokenization"), ("tokenization", ".")]
```

Trigrams (3-grams):

```
tokens = [("This", "is", "an"),
          ("is", "an", "example"),
          ("an", "example", "of"),
          ("example", "of", "tokenization"),
          ("of", "tokenization", ".")]
```

Since the Large Pre-trained Language Model uses a vocabulary, in which all used tokens should be found so that the model can process them, **Byte-pair Encoding**, **WordPiece Algorithm** and **SentencePiece** are used to correctly create these tokens.

- **Byte-pair Encoding:** This method first selects all characters as tokens. Then, successively the most frequent token pair is merged into a new token and all instances of the token pair are replaced by the new token. This is repeated until a vocabulary of prescribed size is obtained. Note that new words can always be represented by a sequence of vocabulary tokens and characters. Common words end up being a part of the vocabulary, while rarer words are split into components, which often retain some linguistic meaning. In this way, out-of-vocabulary words are avoided. [22]

- **WordPiece Algorithm** also starts by selecting all characters of the collection as tokens. Then it assumes that the text corpus has been generated by randomly sampling tokens according to their observed frequencies. It merges tokens in such a way that the likelihood of the training data is maximally increased. There is a fast variant whose computational complexity is linear in the input length. [22]
- **SentencePiece** is a package containing several subword tokenizers and can also be applied to all Asian languages. All the approaches effectively interpolate between word level inputs for frequent words and character level inputs for infrequent words. [22]

As the current definition of tokenization shows, sentences with the same meaning but in a different languages will be tokenized with different tokens, and the number of tokens may also be different. Since the thesis is focused on generation medical reports in the Czech language, an example of sentence tokenization in the English language and the Czech language will be given to compare the results. For this tokenization is used tokenizer of LLaMA 3 by META, which use Byte-pair Encoding.

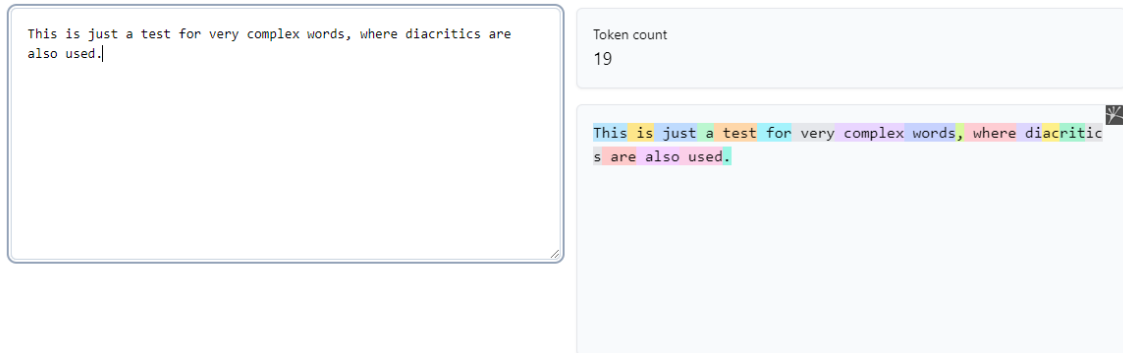


Figure 2.1: Tokenization of the sentence written in the English language.

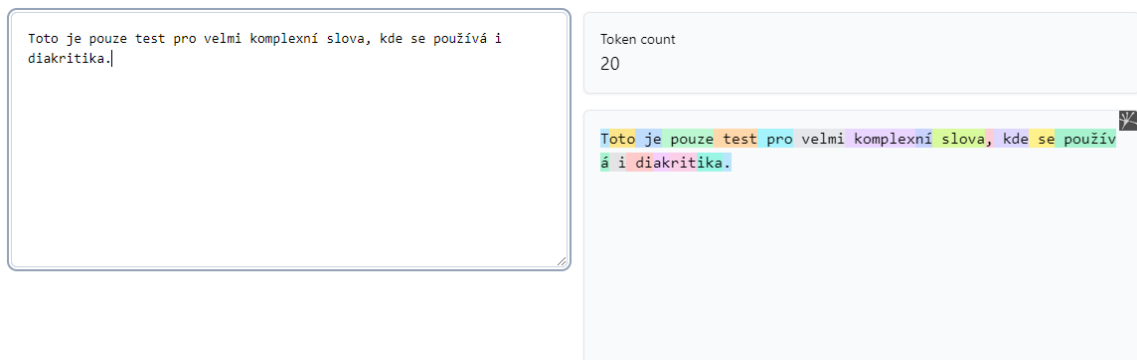


Figure 2.2: Tokenization of the sentence written in the Czech language.

As can be seen in Figure 2.1, the given sentence, that has 14 words and 2 punctuation marks, is encoded into 19 tokens. On the Figure 2.2, the given sentence, that has 13 words and 2 punctuation marks, is encoded into 20 tokens. The difference between the better encoding of these two languages lies in the fact that model is trained primarily with text in

English language and thus also its vocabulary contains tokens of English words. It cannot be concluded from this, that the given model works better with text in English language than with text in Czech language, but the use of a larger number of tokens results into a usage of larger amount of required memory.

2.1.3 Token embeddings

They represent the meaning of each word by a vector of real numbers with hundreds of dimensions and each dimension capture different aspect. Between these vectors can be computed a sort of relation between different words. Those embeddings can be categorized into two main types: *Simple embeddings* and *Contextual embeddings*.

Simple embeddings assign each token in the vocabulary a unique, fixed-length vector representation. These representations are pre-computed and remain constant throughout the duration of a task. It is typically pre-trained on large text corpora using unsupervised learning techniques. During training, the embeddings are learned by optimizing an objective function that encourages similar words to have similar embeddings. Simple embeddings capture both semantic and syntactic information about tokens. Tokens with similar meanings or usage patterns tend to have similar embeddings, which allows the embeddings to encode semantic relationships between words. They are based on global statistics of language usage obtained from the entire training corpus. As a result, they do not consider the surrounding context in which tokens appear and provide a general representation of words based on their overall usage patterns. [22]

Common approaches of Simple Embeddings [22]:

- **Word2Vec:** Word2Vec is a popular simple embedding technique that represents words as dense vectors in a continuous vector space, which learn embeddings by predicting neighboring words or context words given a target word.

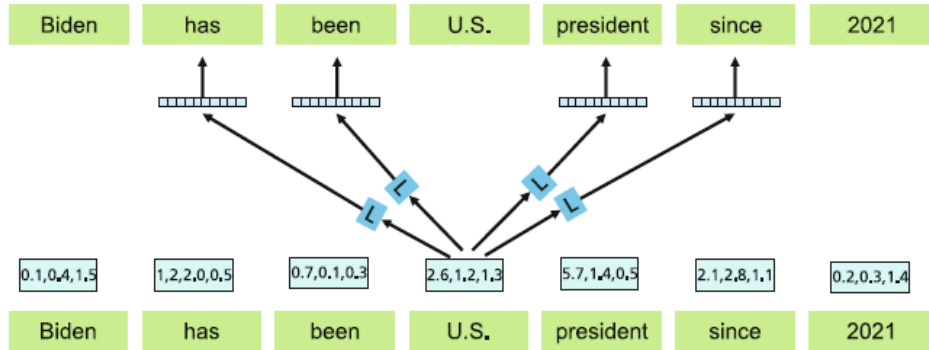


Figure 2.3: Word2vec predicts the words in the neighborhood of a central word by logistic classifier L. Image taken from [22].

- **Global Vectors for Word Representation** is another widely used simple embedding method that learns word embeddings by factorizing the co-occurrence matrix of words in the corpus. It captures global statistics of word co-occurrences to generate embeddings that reflect both semantic and syntactic similarities between words.

- **FastText** extends the Word2Vec model by representing each word as a bag of character n-grams, allowing it to capture morphological information and handle out-of-vocabulary words more effectively.

Contextual embeddings are a type of word representation in natural language processing that captures the meaning of a word in the context of a sentence or document. Unlike simple embeddings, which assign fixed representations to each word regardless of context, contextual embeddings are dynamically computed based on its context within a sentence or a document, allowing for a more nuanced understanding of word meaning. Contextual embeddings often capture bidirectional context, meaning they consider both preceding and subsequent words when computing the representation of a word. This bidirectional context enables contextual embeddings to capture long-range dependencies and semantic relationships between words within a sentence.

Large Pre-trained Language Models, which use this type of embeddings, are [22]:

- **Bidirectional Encoder Representations from Transformers**, also known as BERT, is a transformer-based model that generates contextual embeddings by pre-training on large text corpora using masked language modeling objectives. BERT representations capture bidirectional context through self-attention mechanisms.
- **Generative Pre-trained Transformer**, shortly GPT, is another transformer-based model that generates contextual embeddings by pre-training on large text corpora using autoregressive language modeling objectives. GPT representations capture unidirectional context and are suitable for generating text.

Contextual embeddings also capture information about the relative positions of words within sentences, including word order and positional relationships. The embeddings of words vary depending on their positions within the input sequence, allowing the model to understand the sequential nature of language. [22]

This is shown on Figure 2.4.

position embeddings	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
	+	+	+	+	+	+	+	+	+	+	+
segment embeddings	x_A	x_A	x_A	x_A	x_A	x_A	x_B	x_B	x_B	x_B	x_B
	+	+	+	+	+	+	+	+	+	+	+
token embeddings	$x_{[CLS]}$	x_{my}	x_{dog}	x_{is}	$x_{[MASK]}$	$x_{[SEP]}$	x_{he}	x_{likes}	x_{play}	$x_{\#\#ing}$	$x_{[SEP]}$
input tokens	[CLS]	my	dog	is	[MASK]	[SEP]	he	likes	play	##ing	[SEP]

Figure 2.4: Token embeddings for Transformers, like BERT or GPT, contains position information due to parallel computations of Transformers. Image taken from [22].

2.1.4 Special tokens

Special tokens play a crucial role in various natural language processing tasks, providing additional information or a structure to input sequences processed by models. Overview of special tokens used by BERT or GPT [22]:

- **[CLS] (Classification) Token** represents the aggregation of the input sequence for classification.
- **[SEP] (Separator) Token** separates segments of text in tasks involving multiple text inputs, such as sentence pairs or question-answering.
- **[MASK] Token** is used in masked language modeling tasks like BERT’s pre-training objective. Tokens are randomly masked during training, and the model is trained to predict them based on surrounding context.
- **[UNK] (Unknown) Token** is used as representation of out-of-vocabulary words or tokens not present in the model’s vocabulary. This token is used during inference to handle unknown tokens.
- **Beginning-of-Sentence (BOS) Token** : Marks the beginning of a sentence in tokenized sequences.
- **End-of-Sentence (EOS) Token**: Marks the end of a sentence in tokenized sequences.
- **End-of-Text (EOT) Token**: Marks the end of the entire text or a document in tokenized sequences.

2.1.5 Prompt

A prompt is a textual input provided by the user to get a response or trigger an action from the model. The prompt can include questions, instructions or any text that the model needs to analyze and generate a response based on. Prompt include even tokens, which help divide prompt into section or make prompt for model more understandable.

The prompt should be as clear and specific as possible to ensure the model understands exactly what is expected. Providing context or additional information can help the model generate a more precise and relevant response. The format of the prompt can vary, ranging from questions to commands to text snippets that serve as a basis for the model to generate a response.

Well-chosen prompts can significantly impact the quality and usefulness of the responses generated by model. Also, prompt needs to be tokenized and convert into token embeddings so model is able process the prompt.

2.2 Large Pre-trained Language Model’s Architectures

Large Pre-trained Language Models typically use transformer architecture as their backbone. Transformers have become the standard architecture for many natural language processing tasks due to their ability to efficiently capture long-range dependencies in sequences as was mentioned in Subsection 1.3.7.

Transformers rely on mechanisms like self-attention and feed-forward neural networks to process sequential data. This architecture allows them to effectively model relationships between tokens in a sequence, making them well-suited for tasks such as language modeling, text generation and machine translation.

2.2.1 Attention mechanism

The attention mechanism is a crucial component of transformer architectures in machine learning, enabling models to focus on relevant parts of input or output sequences during sequence processing. This mechanism allows models to efficiently learn long-range dependencies in sequences and capture context for better predictions.

Each word or token in the input sequence is represented by a vector. This is input representation and it is used to generate three vectors: query, key, and value. Query represents actual token and key represents other tokens in the sequence. [22]

For each token, a scalar is computed between the query and the key of each other token in the sequence. This scalar expresses the „importance“ of the query with respect to the given token. The scalars are normalized using an activation function like softmax to obtain attention weights. These weights are then used to weight the values corresponding to the respective tokens. The sum of weighted values forms the output representation for the given token as it is shown in Figure 2.5.

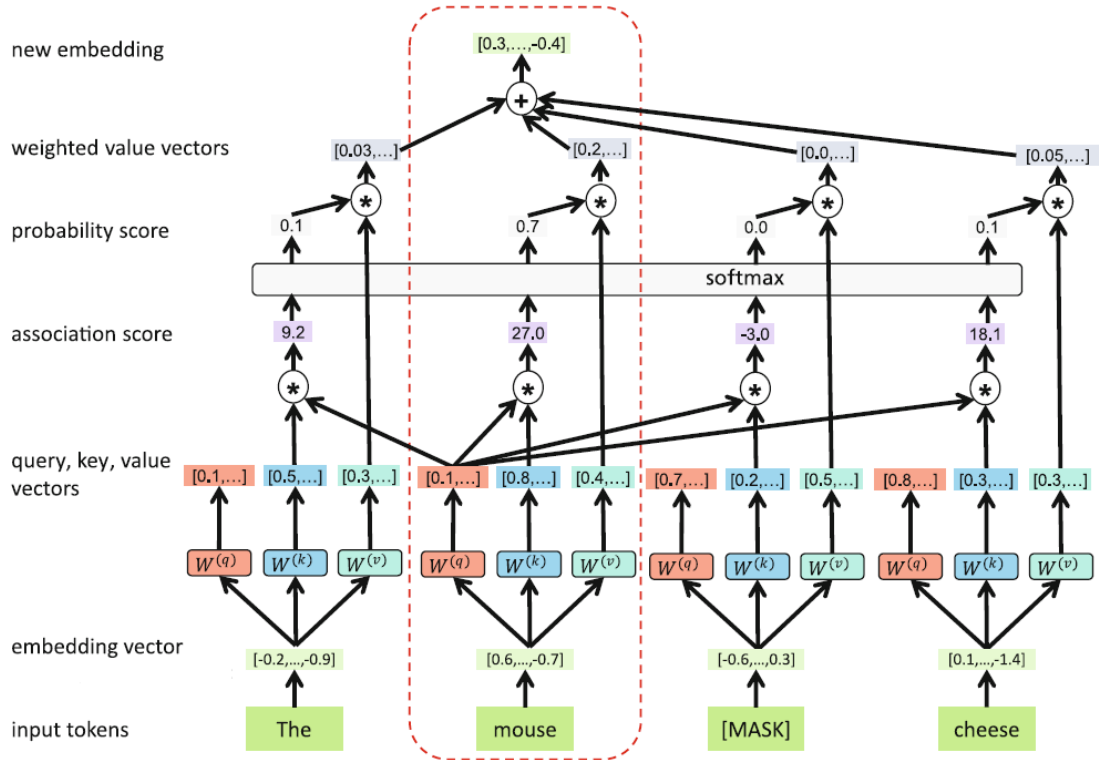


Figure 2.5: Computation of a contextual embedding for a single token “mouse” by attention mechanism. Image taken from [22].

The attention mechanism is often implemented as multi-head attention, which allows the model to focus on different aspects of the input. In multi-head attention, the query, key, and value are linearly projected into multiple spatial subspaces, and then the attention mechanism is applied to each of these subspaces. The outputs are then concatenated and linearly combined to obtain the final output. Visual representation of multi-head attention is shown in Figure 2.6.

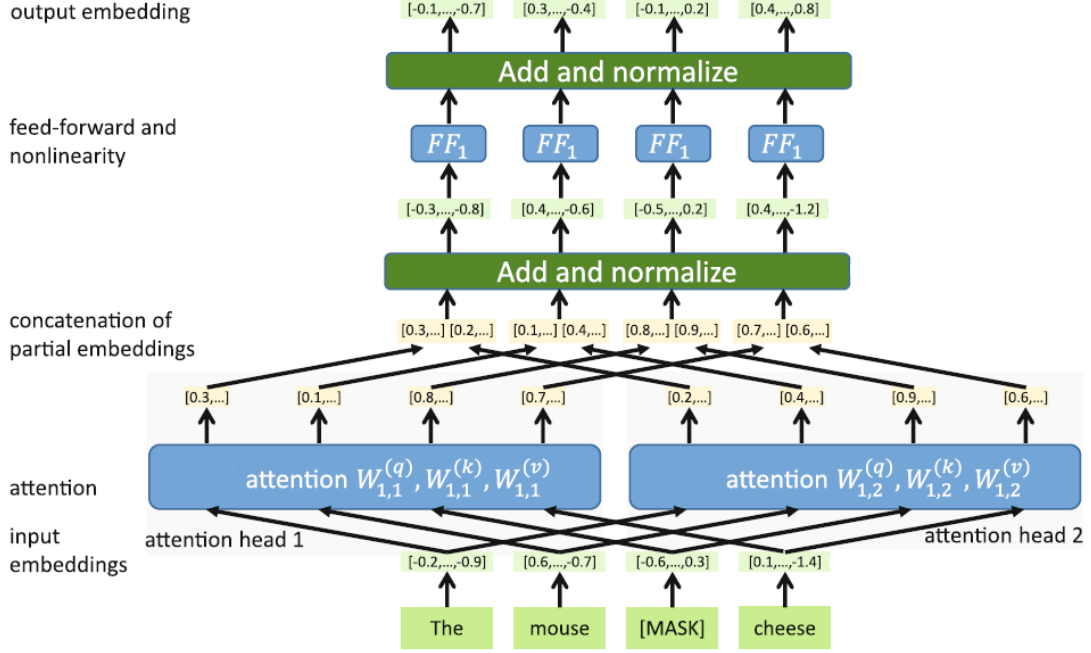


Figure 2.6: Visual representation of multi-head attention. Image taken from [22].

Large Pre-trained Language Models typically consist of multiple transformer layers as it is shown in Figure 2.7. By stacking multiple transformer layers, the model can learn hierarchical representations of the input sequence. Lower layers capture basic features and local dependencies, while higher layers capture more abstract features and long-range dependencies.

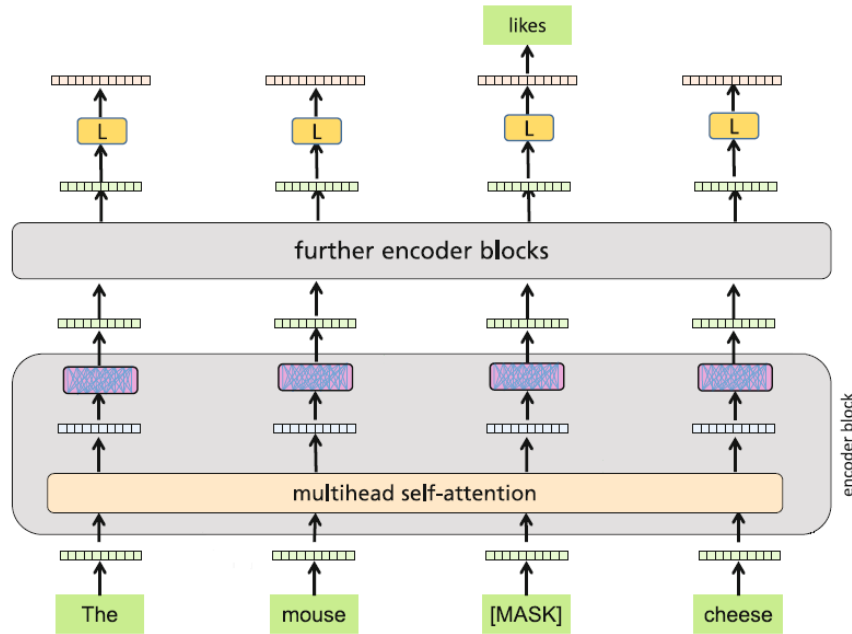


Figure 2.7: Scheme of stacking multiple transformer layers. Image taken from [22].

2.2.2 BERT

Bert is natural language processing model introduced by Google [11]. Thanks to its architecture, which was not previously used, it became an effective model in understanding the semantics and relationships within sentences. The advantage of the architecture lies in the fact that it takes into account the probability of the occurrence of a word between two words and not sequentially from one side only.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word). [11]

Another advantage is that it does not only take the sequence of words, but also the continuity of the sentences themselves.

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. [11]

BERT is also labeled as an autoencoder model whose main task is to derive context-sensitive embeddings for tokens. In each layer of BERT, the lower layer embeddings are transformed by self-attention to a new embedding. The main training task is to predict words from the input sequence, which have been replaced by a [MASK] token. This is done by using the last layer embedding of the token as input to a logistic classifier, which predicts the probabilities of tokens for this position. During training the model parameters are optimized by stochastic gradient descent. This forces the model to collect all available

information about that token in the output embedding. The first input token is the [CLS] token. During training, it can be used for next sentence prediction, where a logistic classifier with the [CLS]-embedding as input has to decide, if the first and second sentence of the input sequence belong together or not. [22]

With additional training for a specific task, the BERT model can provide output to this task through the [CLS] token.

2.2.3 GPT

GPT has transformer architecture similar to BERT. GPT model is mainly used in text generation tasks.

Model generate text by predicting the next token in a sequence based on the tokens that came before it. Since GPT generates the tokens by sequentially applying the same model, it is called an autoregressive language model and this generate text process is shown in Figure 2.8. [22]

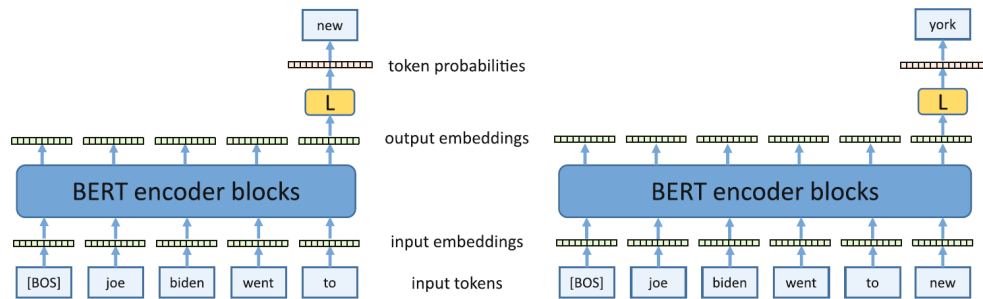


Figure 2.8: Transformer model predicts the next token in token's sequence. Image taken from [22].

This text generation process is same as text generation by recurrent neural networks or its variants. GPT uses attention mechanism described in Subsection 2.2.1, which is typical for transformer architecture. Thanks to this attention mechanism, GPT model can provide better result in text generation task than recurrent neural networks. Blocks used in GPT are referred to as decoding blocks. Difference between word prediction by BERT and text generation by GPT is shown in Figure 2.9. Text generation by GPT can be the same process as word prediction by BERT, if the BERT model predicts word at the end of a sequence.

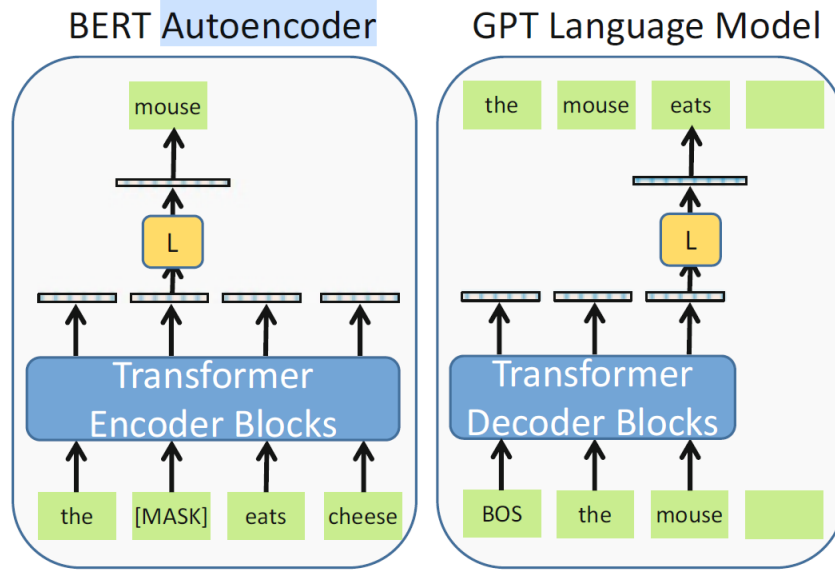


Figure 2.9: Comparison between word prediction by BERT and text generation by GPT. Image taken from [22].

2.2.4 LLaMA

LLaMA is natural language processing model introduced by Meta. It is an autoregressive language model like GPT, but LLaMA uses different type of attention mechanism than BERT or GPT.

LLaMA uses **Grouped-query attention** and BERT or GPT uses **Multi-head attention**. Instead of processing each query independently, queries are grouped into subsets based on some predefined criteria. For example, queries may be grouped based on their semantic similarity, syntactic structure, or positional proximity in the sequence. Within each query group, attention scores are computed independently using the grouped queries and all key vectors. This means that each query in the group attends to all key vectors, but only within its own group. The Difference is demonstrated in Figure 2.10.[3]

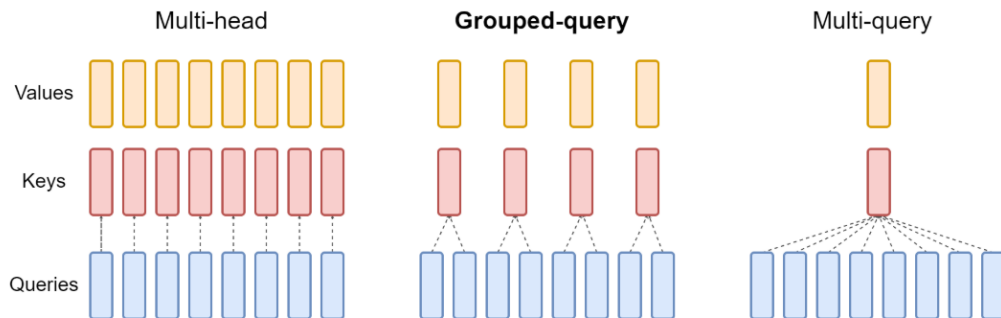


Figure 2.10: Difference between types of attention. Image taken from [3].

The main advantage of this model is that there are fewer parameters than other Large Pre-trained Language Models [32]. In addition, the benefits of Grouped-query attention should be noted, as it achieves quality close to multi-head attention while being almost as fast [3]. As a result, the model can run locally without requiring a lot of performance while still generating text with good results in a reasonable amount of time.

2.3 Learning

Large Pre-trained Language models are neural network models with a very high number of parameters, so the training such large models is computationally demanding. Due to this reason, the training is divided into 2 stages: **Pre-training** and **Fine-tuning**. The model can use the knowledge acquired during Pre-training through transfer learning and thanks to that, the performance on the fine-tuning task is much better [22].

- In **Pre-training**, the model is trained on large amount of text documents of various types and even more various contexts. Unsupervised learning is used for learning in Pre-training stage, so no manual annotation is required [22]. After completing this stage of the learning, the model generates text based on the learned probability of occurrence of a word in a given sequence of words. The model is not yet specialized for performing the given task.
- **Fine-tuning** the model is a training process for the specialization of the model, so that the model provides better results for a given task. For this stage is used far less samples compare to Pre-training. **Supervised fine-tuning** is traditional supervised learning, where instead of randomly chosen weights of the model during its initialization, the weights are given by loading the pre-trained model. **Reinforcement Learning with Human Feedback** is used to improve the results of the given model when the real person evaluates the output of the model using feedback. The model is then either „punished“ or „rewarded“ for the answer.

Parameter Efficient Fine-Tuning

It is true that a model with a large number of parameters can solve much more complex tasks more successfully than a model with a low number of parameters. But with a large number of parameters, also comes a great need for computing power.

Therefore, fine-tuning all model weights is still a very computationally demanding process. By using a PEFT, this computational complexity is reduced. PEFT, stands for Parameter Efficient Fine-Tuning. It is a set of techniques or methods to fine-tune a large model in the most compute and time-efficient way possible, without losing any performance which you might see from full fine-tuning. Really big models is almost impossible to fine-tune them without spending tens of thousands of dollars for computation resources. When it is necessary to use such big models for better performance, PEFT comes in. This is done by fine-tuning only the most important and relevant parameters in the neural network. The techniques introduce new parameters in the network or freeze the whole model except for some parts to make it easier to train the model. [23]

LoRA, which stands for Low-Rank Adaptation, is one of PEFT methods. It operates on a crucial insight. The difference between the fine-tuned weights for a specialized task and the initial pre-trained weights often exhibits “low intrinsic rank” - meaning that it can be approximated well by a matrix of low rank. A low-rank matrix has few linearly independent

columns, which means, in simple terms, that the matrix is less “complex”. One cool property of low-rank matrices is that they can be represented as the product of two smaller matrices. This realization leads to the hypothesis that this delta between fine-tuned weights and initial pre-trained weights can be represented as the matrix product of two much smaller matrices as it is shown in Figure 2.11. The key idea is to focus on updating these two smaller matrices instead of the entire original weight matrix during training, leading to improved computational efficiency. LoRA contains 2 essential parameters: **Rank (r)** and **Alpha(r_alpha)**. [20]

Parameter Rank determines the rank of the low-rank matrices that are introduced to the model. A lower rank means fewer parameters to fine-tune, leading to faster training and less memory usage. Alpha is a scaling factor that controls the magnitude of the update applied to the low-rank matrices.[20]

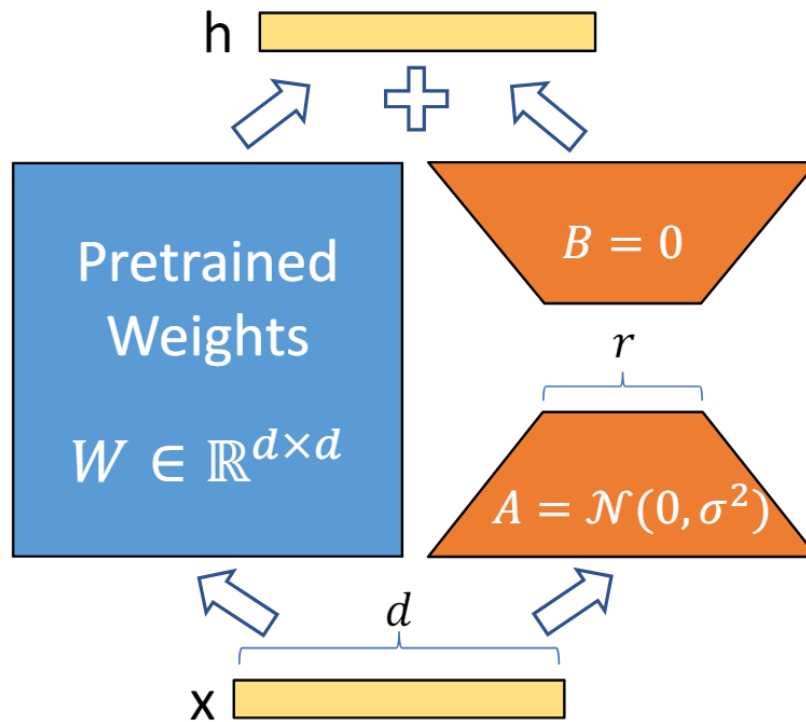


Figure 2.11: LoRA principle using initial pre-trained weights and two much smaller matrices. Image taken from [12].

2.4 Inference and evaluation

The most crucial part of training an artificial intelligence model is evaluating whether a given model produces better results than other models. If the model is successful in evaluation then it can be used to generate text for users. This generation of text by model is referred to as Inference.

2.4.1 Inference

Inference is a fundamental concept in the realm of Large Language Models. Inference is the process of generating new data or predictions based on the model’s learned patterns. It is the step where the model, after being trained on a large dataset, uses its learned knowledge to make predictions or generate responses. This process is central to the functioning of Large Pre-Trained Language Models and is what allows them to interact with users in a meaningful way. [6]

The inference process involves complex computations and algorithms. It is not just about generating any response, but about generating the most probable response based on the model’s understanding of the input. This involves a deep understanding of language, context, and the nuances of human communication. [6]

The inference parameters play a big role in the output the given model generates. These parameters are Temperature, Top_k, and Top_p.

Temperature is a parameter that adjusts the randomness of text generation by altering the probability distribution. Low Temperature leads to more predictable and conservative outputs. High Temperature leads to more varied and creative outputs.

Parameter **Top_k** limits the number of possible next words to the K-most probable words before sampling. This restricts the model’s choices to the K words with the highest probabilities.

Parameter **Top_p** selects the smallest set of words whose cumulative probability exceeds a specified threshold P.

2.4.2 Evaluation techniques

In addition to the fact that a person can evaluate the results himself, automatic methods can also be applied. A major disadvantage of this type of artificial intelligence is that automatic methods for evaluating these models lack credibility, as they would need to understand the overall meaning of the generated text.

Nevertheless, these evaluation methods prove with a large deviation whether the given model is successful or not.

These techniques are [22]:

- **BLEU** compares counts of 1-grams to 4-grams of tokens. The BLEU metric ranges from 0 to 1, where 1 means an identical output with the reference. Although BLEU correlates well with human judgment, it relies on precision alone and does not take into account recall—the proportion of the matched n-grams out of the total number of n-grams in the reference translation.
- **ROUGE** unlike BLEU is a recall-based measure and determines which fraction of the words or n-grams in the reference text appear in the generated text. It determines, among other things, the overlap of unigrams or bigrams as well as the longest common subsequence between a pair of texts.

ROUGE-1, ROUGE-2 and ROUGE-L are metrics of this method and also are used to measure success of the models in this thesis. ROUGE-1 measures the overlap of unigram between the pair of texts. ROUGE-2 determines the overlap of bigrams between the pair of texts. ROUGE-L measures the length of the longest sequence of words that is shared between both texts. This length is divided by the number of words in the reference text.

Within this thesis, the general metrics Precision, Recall and F1-score are used to display the results of ROUGE metrics. Precision focuses on the system’s accuracy. A high value means that most of the generated words are correct, but it does not indicate whether all important parts of the reference text are covered. Recall focuses on the system’s completeness. A high value means that most of the reference words are covered in the generated text, but it does not indicate how many unnecessary words are generated. F1-score combines precision and recall into a single number, providing an overall performance evaluation of the system.

- **METEOR** performs a word-to-word alignment between the translation output and a given reference translation. The alignments are produced via a sequence of word-mapping modules.

Chapter 3

Assistant for Creating Medical Reports

Creating medical reports is one of the administrative duties of the examining doctor. By creating an assistant for creating a report of this kind, it can help the doctor to create these reports faster, more easily, and it will also contribute to the elimination of errors caused by human factors.

The use of properly pre-trained language modules will speed up the development of this assistant, and through the fine-tuning process, the creation of reports will be achieved as correctly as possible.

The exact task of the assistant consists in summarizing text documents created during the patient's hospitalization to a dismissal report.

3.1 Dataset

The dataset was provided by company STAPRO s.r.o., which specializes in the development of information systems for hospitals, clinics and laboratory facilities.

3.1.1 Entities

The provided dataset contains 5 files and each file contains data and attributes of the entity it represents. So the dataset contains the data of the entities:

- ***Patient***: Provides data of patient.
- ***Department***: Provides data of department of hospital.
- ***Hospitalization***: Provides data of admission to hospital for treatment.
- ***Hospitalization_diagnoses***: It provides data on diagnoses for a given hospitalization episode.
- ***Document***: Provides data to documentation created by doctors.

Relationships between entities are shown on Figure 3.1.

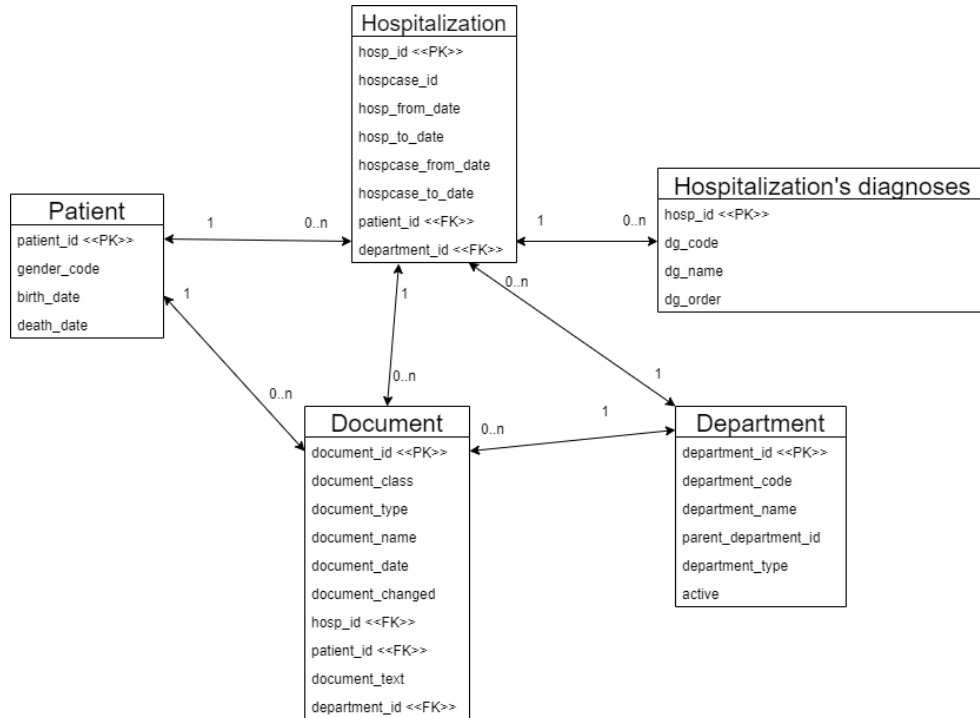


Figure 3.1: Entity relationship diagram of given dataset.

3.1.2 Dismissal reports

The dismissal report is issued when the patient is dismissed from the hospital. This is a report, that summarizes the course of a treatment and essential information from other documents related to the given treatment.

The dismissal report, in accordance with the laws of the Czech Republic, shall include [1]:

1. brief information about medical history and current illness
2. the duration and course of one-day or inpatient care describing why the patient was hospitalized and what was the result of diagnostic efforts
3. a summary of the diagnoses for which the patient was provided medical care during hospitalization
4. a record of previous treatment and the results of examinations that are essential for the provision of other health services
5. an overview of the medical procedures performed during hospitalization, which are significant for the further provision of health services, including their results and information on the complications that have occurred
6. recommendations for the provision of necessary health services, including medical rehabilitation and nursing care and recommendations for a diet regimen, medicinal products, food for special medical purposes and their dosage, and recommendations for medical devices intended for the provider who will provide other health services, and recommendations for a medical assessment report

The Preliminary Dismissal Report shall include [1]:

1. basic data on the course of hospitalization
2. a summary of the diagnoses for which the patient was provided medical care during hospitalization
3. a brief record of previous treatment, medical rehabilitation and nursing care, dietary regimen, including the indication of medicinal products, food for special medical purposes and medical devices with which the patient is equipped
4. recommendations for the next procedure in the provision of health services

This is the foundation. Furthermore, each hospital adjusts the content of the dismissal report in its own guidelines.

3.1.3 Data for Supervised fine-tuning

The provided dataset contains many entities, but the most important information is found in the entity **Document**, more precisely in its attribute **document_text**, where the text written by a doctor or other relevant person is stored, and contains information that can be used to create an assistant.

Attribute **document_class** indicates the category to which the given document belongs. These categories are shown in Table 4.2.

Further, documents from the category of dismissal report, admission report and Decours will be used to create the assistant.

Documents of admission report, saved in entity **Document**, are divided into a structures with the following subsections: Anamnesis; Measured data; Current illness; Status praesens; Status localis; Diagnostic sheet; Care plan.

From this subsections, **Current illness** provides information that can be useful in creating the course of hospitalization and at the same time does not contain a large amount of additional information that would only unnecessarily take up memory space and increase the necessary computing power.

Structure of Decours documents can be divided into **Diagnostic Summary** and **Daily Progress Notes**.

Daily Progress Notes represents an ideal part for adding input information to the model. It offers information about every single day during hospitalization and these notes are written in a concise form, which also reduces the memory and computational complexity required for running the model.

The input to the model will therefore be created by Current illness and Daily Progress Notes. Example of input is shown on Figure 3.2.

Example of input

onemocnění:

Pacient přijata k provedení diag. koronarografie před zařazením na waiting list před transplantací ledviny. Echo srdce: EF LK 60-65%, porucha diastol. relaxace LK, lehká sym. hypertrofie LK, bez lokálních poruch kinetiky, bez význam. chlopen.vad
Subj: Neguje bolesti na hrudi, dušnost, palpitace, pre/synkopy.

Denní průběh:

10.3.2022 8:40 Zapsal [REDACTED]:

S: bez obtíží

O: afebrilní, eupnoe, AS reg., dýchání alv. čisté, břicho měkké, nebol., DKK bez otoku, lýtka volná, akra prokrvená, zápěstí po skg klidné telem: SR 61/min
dimise

Figure 3.2: Example of input for model.

The label will then be created from the dismissal report document, which has the following structure: Diagnostic Summary; Anamnesis; Current illness; Status praesen; Status localis; Treatment Summary; Surgical Procedur; Laboratory Results; Measured Data; Hospital Course; Patient state at dismissal; Recommendations; Recommended Therapy.

The label is exactly Hospital Course of Dismissal Report and example is shown on Figure 3.3.

Example of label

Pacientka přijata k provedení diag. koronarografie před zařazením na waiting list před transplantací ledviny. Výkon proveden dne 9.3. s nálezem koronární tepny bez význam. stenoz (50% RIA, 60% ACD). Pacientka oběh. komp., afebrilní, propuštěna, dnes pravidelná dialýza. Odvoz s rodinou.

Figure 3.3: Example of label for model for input in Figure 3.2.

With provided input, this leads to summarization task.

Other parts of dismissal reports are already created like Anamnesis, Current illness, Status praesens, Status localis and can be copied from other documents. And for another parts like Recommendations and Recommended Therapy, the model would need to understand the context of the patient's current condition with the treatment that was performed in the past. This leads to the need to have access to the given data, which would be a problem in terms of the need for a large amount of memory and computing power.

Even if the structure of the text documents is obvious, it is only a form of guidance for doctors and it is not exactly necessary to follow this structure by them. In reality, doctors have a free hand in creating this documents, which can lead to violations of data structures. This can complicate data extraction and leads to the creation of a low-quality dataset, which subsequently leads to the creation of a low-quality assistant.

Chapter 4

Implementation and testing

This chapter encompasses the two most essential parts of this thesis. The process of obtaining a model to serve as an assistant in generating medical documentation, thus addressing the thesis problem, is illustrated in Scheme 4.26. Programming languages and libraries used for the implementation, including both the model training process and its runtime, as well as the tools enabling the model's usage, will be also introduced.

The process of obtaining a model is divided into individual steps, each accompanied by tests that demonstrate how each step influenced the acquisition of the model. These steps are as follows:

- **Data preparation:** A dataset suitable for learning the Large Pre-trained Language Model will be created from the data provided by the company STAPRO s.r.o.
- **Choose suitable model:** A suitable model will be chosen that can process the given input text so as to generate the most correct output.
- **Supervised fine-tuning of model:** Choosing the right training procedure and additional training parameters for the best possible fine-tuning of the model.

The current scheme for fine-tuning the model consists of:

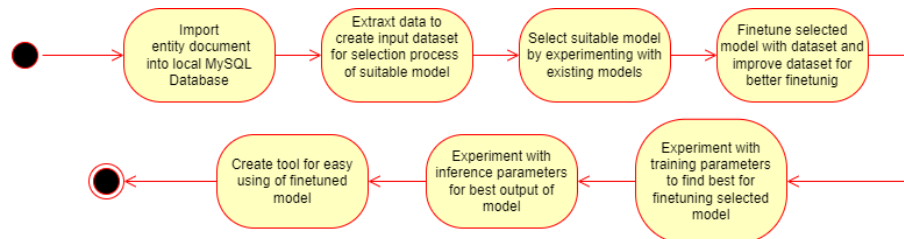


Figure 4.1: Simple schema of fine-tuning process of large language model.

4.1 Programming Languages, Libraries and Technologies

Various libraries were utilized to expedite the model creation process, with some being indispensable for the model's development. These include not only the libraries used for model creation but also those necessary for the model's operation and applications essential for model's successful use. For better and faster way to process data files provided by STAPRO s.r.o., MySQL Database system is used.

4.1.1 MySQL

MySQL is an open-source relational database management system used for storing, managing, and retrieving data. MySQL is widely utilized in various applications, from small projects to large enterprise systems, due to its speed, reliability, and ease of use. This system supports the standardized SQL for database communication, allowing users to perform operations such as creating, updating, and querying data. MySQL is renowned for its ability to handle large volumes of data and deliver fast performance, which is particularly important for web applications and services that require quick response times. [19]

4.1.2 Python

For this whole finetuning process was used python as programing language.

Python is a high-level, interpreted programming language known for its readability and ease of use. It is widely used in various fields, including web development, scientific computing and machine learning. Python has a clean and readable syntactic style, making it easy to learn and understand. It supports multiple programming paradigms, including object-oriented, procedural, and functional programming and has a rich standard library and numerous external libraries. The code is executed directly by the interpreter, allowing for quick testing and debugging. [25]

Popular Python frameworks include Flask, TensorFlow, and PyTorch and the libraries commonly used in natural language processing tasks include NumPy for numerical computations, pandas for data manipulation, transformers and peft for training and inference of Large Pre-Trained Language model.

4.1.3 Transformers and PEFT libraries

Transformers provides APIs and tools to easily download and train state-of-the-art pre-trained models. Using pretrained models can reduce compute costs required to train a model from scratch. These models support common tasks in different modalities, such as Natural Language Processing, Computer Vision and Audio classification. Transformers support framework interoperability between PyTorch and TensorFlow. This provides the flexibility to use a different framework at each stage of a model's life like train a model in three lines of code in one framework, and load it for inference in another. [13]

PEFT (Parameter-Efficient Fine-Tuning) is a library for efficiently adapting large pre-trained models to various downstream applications without fine-tuning all of a model's parameters because it is prohibitively costly. PEFT methods only fine-tune a small number of model parameters, which significantly decreasing computational and storage costs, while yielding performance comparable to a fully fine-tuned model. This makes it more accessible to train and store large language models on consumer hardware. [17]

4.1.4 Flask

Flask is a lightweight web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper and has become one of the most popular Python web application frameworks. Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy. [16]

4.1.5 Javascript

JavaScript is a high-level programming language commonly used to create interactive web pages. It is interpreted and runs directly in the client's web browser, allowing for dynamic content changes based on user actions. JavaScript is also widely used on server platforms. The language serves as a fundamental building block for modern web applications, enabling manipulation of the Document Object Model to alter page content without needing to refresh it. Additionally, it supports asynchronous programming through callbacks, promises, enhancing efficiency in data manipulation and server communication. [18]

4.1.6 React.js

React is a popular JavaScript library for building user interfaces, developed and maintained by Facebook. It focuses on creating single-page applications and web applications that are efficient and fast in user interaction. React employs a component-based approach to user interface development, allowing applications to be divided into independent, reusable components. These components can manage their own state and respond to changes in the application, enhancing code modularity and maintainability. One of React's key concepts is the virtual Document Object Model, which enables efficient rendering of changed parts of the page without redrawing the entire page. This improves performance and user experience during application interactions. [9]

4.2 Data preparation

As a first step is need to insert data to MySQL database from CSV file document. This file represents entity document, which was mention in Figure 3.1, and contains all needed information to create dataset to training model. MySQL is used for faster work with data thanks of create index on specific row which are most crucuel in `patient_id` and `document_type` rows. This is done by script **Data_preparation.py** implementing using Python language.

It begins by importing the necessary libraries: `pandas` for data manipulation and `mysql.connector` for connecting to the MySQL database.

Next, it defines the connection parameters for the MySQL database, which include the host, user, password, and database name. It also specifies the path to the CSV file that contains the data to be loaded, as well as the table name in the database and the primary key column.

The script contains a function, **create_connection**, which attempts to establish a connection to the MySQL database using the specified configuration parameters. If there is an error during the connection attempt, the function displays an appropriate error message and returns `None`.

Another function, **load_csv_to_mysql**, reads data from the CSV file into a `pandas DataFrame`. It then opens a cursor for executing SQL operations. For columns with date values like `document_date` and `document_changed`, it converts the values to `datetime` format. The function then creates an SQL query to create a table in MySQL, if it does not already exist, and executes this query.

Next, the function iterates over the rows of the `DataFrame` and for each row, it creates and executes an SQL query to insert the data into the table. After successfully inserting all the data, the changes are committed. The function includes an `except` block that catches

and prints any errors that may occur during execution. Finally, the function closes the cursor and the database connection.

Document Type	Occurrences
General Nursing Documentation	151611
Laboratory Examination	86135
Invasive Entries	74562
Decours	24324
Dismissal Report	15379
Consultation	14567
Consent	10339
Rehabilitation Procedures	10115
Admission Report	9182
Wound Recording	7453
Epicrisis	7068
Radiodiagnostics	5753
Scoring Schemes	4902
ADL	4101
Pressure Ulcers	3823
Falls	3223
Surgical Protocol	3004
Nurse Education	2850
Finding	1716
Childbirth Report - Child	1519
Other	1335
Childbirth Report - Mother	1060
Pathology - Biopsy	564
Nursing Transfer Report	525
Nursing History	517
Prescription	492
Medication Request	488
Social Record	363
Nutrition	284

Table 4.2: The number of occurrences of individual types of medical documentation insert into database table document.

The table 4.2 shows the type of medical documentation and its count that was uploaded to the local MySQL database. It is obvious that a higher number of Dismissal report type documentation than Admission report was uploaded. This is caused by the fact that some doctors create a preliminary dismissal report and a final dismissal report as two separate documents, and some just supplement the preliminary dismissal report and create a final dismissal report from it, and it is therefore one and the same document.

4.3 Suitable model

Part of this thesis is experimenting with existing models. This is also used for find suitable model, which helps provide better solution of problem of this thesis.

From the mentioned architectures of models in 2.2, the models with LLaMA architecture was chosen for several reasons. The first reason is the feasible requirements for computation and memory resources, as the model is available in versions with a low number of parameters, specifically 13 billion parameters and 7 billion parameters. Another reason is that the model is multilingual; it can generate text in several languages, including Czech. Additionally, the model is open-source and can be fine-tuned locally, which provides flexibility and control over its use. Lastly, the model is relatively new, having been released recently, which means it incorporates the latest advancements in the field.

With the chosen LLaMA-2 architecture, the following types of models were selected, which differ from each other through the process of retraining, and some of them have already gone through the fine-tuning process. However, this is about fine-tuning, which does not directly solve the problem solved in the framework of this thesis, and therefore even these models must be additionally fine-tuned.

Also, some models have increased content length than the original model of LLaMA-2. This is due to experimentation with models where they would take the entire medical documentation as input and not just its parts. Unfortunately, due to the hardware requirements, especially the memory of the graphics cards, which ensured the running of the model, a smaller input was used, and thus the possibility of processing larger input text remained unused.

Experimentation also took place with models of different number of model parameters. These parameters are actually the weights of the neurons of the given neural network 1.3.2. In simple terms, this means that the greater the number of these parameters, the „smarter“ answers the model generates. This is due to the fact that the generated words have a better connection with other generated words of the given text, and thus the overall text has a better impression. However, with a larger number of parameters, the hardware is more demanding, and it is mainly the small memory problem.

4-bit quantization, to reduce the model’s memory requirements, is also used when model is loaded.

These models are:

- ***togethercomputer/Llama-2-7B-32K-Instruct*** [31]: This is a model where its content length was increased from 4096 to 32000 tokens. Also, compared to other used models, it has 7B parameters, which is a lower than other models with 13B parameters.
- ***Yukang/Llama-2-13b-longlora-16k-ft*** [34]: Model has increased content length to 16000 and it is a version with 13B parameters.
- ***Yukang/LongAlpaca-13B*** [36]: It is a fine-tuned version of the ***Yukang/Llama-2-13b-longlora-16k-ft*** model. Finetuning was performed using the ***LongAlpaca-12k*** [35] dataset. The given dataset contains primarily English text, which can worsen or even make it impossible to generate text in the Czech language.
- ***meta-llama/Llama-2-13b-chat-hf*** [33]: This is the original model additionally fine-tuned for chatting. It is therefore aimed at generating whole sentences rather than creating structures or programming codes. It has a content length of 4096 and its number of parameters is 13B.

4.3.1 Dataset

Dataset for finding a suitable model is created by extracting data as it is described in Subsection 3.1.3. For this purpose is used script **Data_extraction.py** written in Python. The code connects to a MySQL database, retrieves specific records, processes these records to extract and clean text data, and saves the processed data into a json file.

The `db_config` dictionary holds the MySQL database connection parameters such as host, user, password, and database. The **create_connection** function establishes a connection to the MySQL database using the parameters in `db_config`.

The **discharge_summary_extract_label** function extracts specific sections of a discharge summary between predefined text markers.

The **discharge_summary_extract_input** function extracts the Current illness section from the admission report. The **dekurz_extract_input** function extracts the Daily Progress Notes section from the text.

The **create_training_dataset** function generates a training dataset by fetching and processing documents of a specific type and `hosp_id`. It retrieves the relevant document text, extracts and processes specific sections, concatenates text from other document types, and then inserts the processed data into array variable.

Finally, the main part of the code establishes a connection to the database, fetches documents of needed types, processes each document to create training dataset and create json file to save dataset.

Metric	Count
Total samples	5557
Content length below 4096	5235
Content length higher than 4096	322

Table 4.3: Evaluation of sample sizes based on model content length.

The Table 4.3 shows the total number of samples in the dataset used for selecting a suitable model. Their number is smaller than could be expected based on the number of occurrences of the admission report from the Table 4.2. This is due to the fact that some patients lacked a certain type of documentation or part of it. In order to create a valid sample, it was necessary that the hospitalization contained an Admission report, Dismissal report and minimally one Decourse. The Table 4.3 also contains information on how many of the given samples will be able to be actually used in relation to the smallest content length of model from the selected models. This represents the model **meta-llama/Llama-2-13b-chat-hf** with a content length of 4096 tokens.

Metric	Precision	Recall	F1-score
ROUGE-1	0.170	0.431	0.233
ROUGE-2	0.092	0.269	0.130
ROUGE-L	0.168	0.426	0.230

Table 4.4: ROUGE metrics between input and reference output of dataset.

The Table 4.4 shows the values measured using the ROUGE metrics, which is described in Section 2.4.2, between the reference output and the input of dataset for finding suitable model. The Recall is pretty high, so reference output cover a lot of information from input. On the other hand, the Precision is low, which means that the reference output contains

additional information that are no longer covered by input. This can lead to bad results of the model, more precisely to generate output with hallucinations, since it can generate additional information based on information, which is not related with it at all.

4.3.2 Prompt

Another part that must be chosen correctly is the prompt, which is described in Subsection 2.1.5. All chosen models do not have the same prompt structure. Some models also have system prompts defined, which are used to define, direct and limit the model from the point of view of its management. In this case, it is about defining the style of the text that the given model generates and limiting the use of expletives or sentences that may sound offensive.

For models togethercomputer/Llama-2-7B-32K-Instruct and Yukang/Llama-2-13b-longlora-16k-ft, the system prompt is not defined and the prompt is shown in Figure 4.2.

Examples in Figures 4.5, 4.7 and 4.9 are generated by meta-llama/Llama-2-13b-chat-hf with inference parameters - Temperature: 0.4, Top_k: 40 and Top_p: 0.9. The justification of the setted values for inference's parameters is in the Subsection 4.3.3.

```
<s>
  [INST]
    {instruction + input}
  [/INST]
```

Figure 4.2: Prompt without system prompt.

The prompt, shown in Figure 4.2, contains the beginning character of the sequence $< s >$ and the definition of the instruction, delimited by the tokens $[INST]$ and $[/INST]$. It consists of *instruction* and *input* variables.

For models Yukang/LongAlpaca-13B and meta-llama/Llama-2-13b-chat-hf, the system prompt is defined and the prompt is shown in Figure 4.3.

```

<s>
[INST]
  <<SYS>>\n
    You are a helpful, respectful and honest assistant.
    Always answer as helpfully as possible,
    while being safe.
    Your answers should not include any harmful,
    unethical, racist, sexist, toxic, dangerous
    or illegal content.
    Please ensure that your responses are socially unbiased
    and positive in nature.\n\nIf a question does not make
    any sense or is not factually coherent, explain why
    instead of answering something not correct.
    If you don't know the answer to a question,
    please don't share false information.\n
  <</SYS>> \n\n
  {instruction + input}
[/INST]

```

Figure 4.3: Prompt with system prompt.

The prompt in Figure 4.3 contains, in addition to the prompt in Figure 4.2, also system prompt, which is delimited by the tokens `<< SYS >>` and `<< /SYS >>`. System prompt in Figure 4.3 sets the model to not generate words and sentences that could sound offensive. It also set that instead of generating a false answer, model should explain why the given question seems nonsense to itself. The Yukang/LongAlpaca-13B and meta-llama/Llama-2-13b-chat-hf models were fine-tuned on this system prompt, so in order to be able to use the given model correctly, the system prompt must also be defined in the same way.

The instruction variable was first defined as it is in Figure 4.4. This is a simple instruction for the model that should direct it to summarization task.

```

instruction= '''Vytvořte souhrn o onemocnění
              a denním průběhu.\n'''

```

Figure 4.4: First definition of instruction variable.

Output of model with use of first definition of instruction variable

Sure, here's a summary of the patient's condition and daily course:

On March 10th, a patient was admitted to the hospital for coronary angiography before being placed on the waiting list for a kidney transplant. The patient has a history of heart disease, including an enlarged left atrium, diastolic dysfunction, and mild hypertrophy. They also have no localized abnormalities in their cardiac function or valve disease. The patient is experiencing symptoms such as chest pain, shortness of breath, and palpitations.

The patient's daily course from March 10th to March 11th includes:

- * On March 10th at 8:40 AM, the patient reported feeling well without any difficulties.
- * At 10:30 AM, the patient's blood pressure was measured at 120/70 mmHg.
- * At 12:00 PM, the patient's heart rate was measured at 61 beats per minute.
- * At 2:00 PM, the patient reported feeling tired and had difficulty breathing.
- * At 4:00 PM, the patient's blood pressure was measured at 130/80 mmHg.
- * At 6:00 PM, the patient reported feeling better but still had some discomfort.
- * At 8:00 PM, the patient's blood pressure was measured at 120/70 mmHg.

Overall, the patient's condition appears stable, with no significant changes over the past 24 hours. However, they are still experiencing symptoms that may require further monitoring or treatment.

Figure 4.5: Output from model for input in Figure 3.2, which has label in Figure 3.3.

As it is shown on output in Figure 4.5, the model did not summarize the input so much, but on the contrary, it expanded it when every single word of information was replaced by a whole sentence. The model also generated text with bullet points. It is structure, which do not appear in the label. Base on that, the second definition of instruction variable is defined as it is shown in Figure 4.6.

```
instruction= '''Vytvořte souhrn celými větami  
o onemocnění a denním průběhu.\n'''
```

Figure 4.6: Second definition of instruction variable.

The second definition of instruction variable in Figure 4.6, compared to the first definition, specifies to the model that the given summary should be generated as whole sentences. This definition should minimize the chance that the model will generate text with bullet points as it is shown on Figure 4.7. It can be seen in the output that even the given part, which was previously generated using bullet points, is even more concise. Still is necessary that the output from the model be generated in the Czech language and not in English, therefore the final definition of instruction variable is defined as it is shown in Figure 4.8.

Output of model with use of second definition of instruction variable

Sure, here's a summary of the patient's condition and daily progress:

On March 10th, a patient was admitted to the hospital for coronary angiography before being placed on the waiting list for a kidney transplant. The patient's echocardiogram showed mild left ventricular hypertrophy with a left atrium ejection fraction (EF) of 60-65%, impaired diastolic relaxation, and no localized abnormalities in contractility or pump function. The patient had no symptoms of heart failure, such as shortness of breath, edema, or syncope.

The patient's daily progress has been monitored since March 10th. On that day, the patient reported no difficulty breathing and had a stable vital sign reading of 61 beats per minute. The patient's temperature, blood pressure, and oxygen saturation were all within normal ranges. The patient also reported no chest pain, discomfort, or other signs of disease.

Overall, the patient's condition is stable, and there are no significant changes or developments since their admission to the hospital. They continue to have no symptoms of heart failure or other concerns.

I hope this summary helps! Let me know if you need any further assistance.

Figure 4.7: Output from model for input in Figure 3.2, which has label in Figure 3.3.

```
instruction= '''Vytvořte souhrn v českém jazyce celými  
větami o onemocnění a denním průběhu.\n'''
```

Figure 4.8: Final definition of instruction variable.

The instruction, defined as it is in Figure 4.8, directed the model to generate output in complete sentences and in the Czech language as it is shown in Figure 4.9. Thanks to this, it is possible to determine which model has the potential to correctly generate the Hospitalization Course of Dismissal Report.

Output of model with use of final definition of instruction variable

Souhrn větami o onemocnění a denním průběhu:

Pacient byl přijat k provedení diagnózy koronárního obrátce před uvedením na čeká-
telský seznam pro transplantaci ledviny. Echo ukázalo srdeční selhání s ejektivi-
tou 60-65%, poruchu diastolníhorelaxace a lehkou hypertrofiu ledviny bez místních
poruch v kontrakci nebo kinetických. Pacient uvedl, že mu dochází k bolestem v krku,
dechu a palpitacím epizodám. Denní průběh pacienta je bez obtíže.

Figure 4.9: Output from model for input in Figure 3.2, which has label in Figure 3.3.

4.3.3 Inference and testing

Inference is provided with parameters which need to be set before generating the output. These parameters are described in Subsection 2.4.1. Default values of parameters were primarily used.

Top_k is set to 40 and **Top_p** is set to 0.9. Other values were also experimented with, for **Top_k** the values were: 70, 20 and for **Top_p** the values were: 0.7, 0.5.

The reason for choosing the default values is that the change in values was not visible on the output of the given models or models give worse outputs. Outputs were checked manually by a human and since the given model is selected based on the result of a text check by a human, it was decided to stay with the basic values of these parameters.

With the different value of **temperature** parameter, the change to output of the model was visible. This parameter is a change from the default value of 1 to 0.4, when the goal was for the model not to generate a text that is too artistic, but rather to summarize the given facts that the text should be factual.

In order to choose the right suitable model, a manual check of the output generated by individual models was also performed and compared with the reference output, and the information provided to the model at the input was also considered. Individual outputs could be categorized into 4 categories and the results are recorded in Tables 4.7, 4.10, 4.13 and 4.16. The **Copied Input** category includes outputs that contained an exact copy of the input. The **Nonsensical Summary** category includes performances that were erroneous, for example word cycling, an inappropriate structure, or a performance without text (only contained white characters). **Summary with Reservations** is the category in which performances with insufficient information were included. **Full Summary** is a category in which performances against which there were no reservations were included. In the category **Summary with Reservations** and **Full Summary**, it was recorded whether the model generated the given output in the Czech language (marked as CZ) or if it was a different language (marked as ANY). The given number in the field of the table represents a part of the whole, where the whole is 250 samples and the number of occurrences is indicated in the bracket.

togethercomputer/Llama-2-7B-32K-Instruct

From the Table 4.5 it follows that the given model decently summarized the input text. The Table 4.6 shows that the output from the model and the reference output are quite different, which can be expected since the given model is not yet fine-tuned in a doctor-like way of summarizing of Hospital Course. When a person manually checked the output of the model, it was found that the given model often copied the given input to the output. The consequence of this copying is that the values measured in the Table 4.5 are very favorable, while the result of the model itself does not correspond to this at all. Results from manual human testing are recorded in the Table 4.7. The model was able to generate a successful summarization in the Czech language in a few cases. But according to the occurrences, it was rather an exception than the constant functioning of the model. Example of output from model is shown on Figure 4.10.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.5295	0.3712	0.4180
ROUGE-2	0.4574	0.3133	0.3532
ROUGE-L	0.5287	0.3707	0.4173

Table 4.5: The ROUGE metrics between output of model and input text to model.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.1307	0.2123	0.1449
ROUGE-2	0.0683	0.1053	0.0722
ROUGE-L	0.1296	0.2106	0.1436

Table 4.6: The ROUGE metrics between output of model and reference output.

Type of Summarization	CZ	Any
Full Summary	0.028 (7)	0.104 (26)
Summary with Reservations	0.096 (24)	0.056 (41)
Nonsensical Summary	0.196 (49)	
Copied Input	0.412 (103)	

Table 4.7: Results of manual check of outputs by human.

Output of model
<p>Patient was admitted for coronary angiography before being placed on the waiting list for kidney transplantation. The patient’s echocardiogram showed an EF of 60-65%, diastolic dysfunction, and mild hypertrophy of the left atrium. There were no localized abnormalities in the heart’s kinetic function or valve disease. The patient reported no chest pain, shortness of breath, syncope, or palpitations.</p> <p>The daily course is as follows:</p> <p>10.3.2022 8:40 Doctor’s note:</p> <p>S: without difficulty</p> <p>O: afebrile, euphoric, regular sinus rhythm, clear respiratory tract, soft stool, lungs clear, skin warm, limbs free from cyanosis, blood pressure: 120/70 mmHg</p> <p>Dimensions:</p> <p>HR: 61 beats per minute</p> <p>Respiration: 20 breaths per minute</p> <p>Temperature: 36.5 degrees Celsius</p> <p>SpO2: 98%</p>

Figure 4.10: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Yukang/Llama-2-13b-longlora-16k-ft

From the Tables 4.8, 4.9 and 4.10, it is quickly understandable that this model failed in the given task. It could not generate a meaningful sentence even once, as can be seen from the table 4.10. There is no example of output because it was just random letters and text has not any meaning at all.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.0160	0.00004	0.00009
ROUGE-2	0.0000	0.0000	0.0000
ROUGE-L	0.0160	0.00004	0.00009

Table 4.8: The ROUGE metrics between output of model and input text to model.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.0120	0.0001	0.0002
ROUGE-2	0.0000	0.0000	0.0000
ROUGE-L	0.0120	0.0001	0.0002

Table 4.9: The ROUGE metrics between output of model and reference output.

Type of Summarization	CZ	Any
Full Summary	0 (0)	0 (0)
Summary with Reservations	0 (0)	0 (0)
Nonsensical Summary	1 (250)	
Copied Input	0 (0)	

Table 4.10: Results of manual check of outputs by human.

Yukang/LongAlpaca-13B

Since it is a fine-tuned model of the **Yukang/Llama-2-13b-longlora-16k-ft** model, which failed the given task as derived in Subsection 4.3.3, its results are a pleasant surprise. According to the Tables 4.11 and 4.12, it seems that the given model failed in summarization. During a manual check of the model’s output, the results of which are recorded in the Table 4.13, it was found that the model summarized the given input quite successfully, even if the language in which the output text was written was English. However, since it is necessary for the model to generate output in the Czech language, this model is not entirely suitable for additional fine-tuning either. Example of output from this model is shown on Figure 4.11, where can be seen that model used some Czech words.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.3002	0.1100	0.1487
ROUGE-2	0.2586	0.0888	0.1212
ROUGE-L	0.2999	0.1100	0.1485

Table 4.11: The ROUGE metrics between output of model and input text to model.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.0829	0.0662	0.0645
ROUGE-2	0.0457	0.0369	0.0355
ROUGE-L	0.0828	0.0662	0.0644

Table 4.12: The ROUGE metrics between output of model and reference output.

Type of Summarization	CZ	Any
Full Summary	0 (0)	0.396 (99)
Summary with Reservations	0 (0)	0.224 (56)
Nonsensical Summary	0.164 (41)	
Copied Input	0.216 (54)	

Table 4.13: Results of manual check of outputs by human.

Output of model
<p>The patient was admitted to the hospital with a diagnosis of heart failure. The doctor recommended a transplantation of the patient's ledviny, which was performed successfully. After the surgery, the patient was put on medication to reduce the symptoms of heart failure, such as shortness of breath, fatigue, and swelling. The patient was monitored closely by the doctors and nurses for any signs of complications or relapse.</p>

Figure 4.11: Output from model for input in Figure 3.2, which has label in Figure 3.3.

meta-llama/Llama-2-13b-chat-hf

This model summarized the given input as shown in the Table 4.14. From results, provided by the manual check, shown in the Table 4.10, it is clear that in part the model helped by copying the input text with a good result in the Table 4.14. On the other hand, the model surprisingly often generated text in the Czech language and its summarization was also often of high quality. According to the Table 4.8, however, the given output from the model does not match the reference output written by the doctor. Example of output from this model is show on Figure 4.9.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.5858	0.3605	0.4282
ROUGE-2	0.3967	0.2560	0.2984
ROUGE-L	0.5780	0.3557	0.4224

Table 4.14: The ROUGE metrics between output of model and input text to model.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.1554	0.2199	0.1641
ROUGE-2	0.0498	0.0759	0.0530
ROUGE-L	0.1536	0.2172	0.1620

Table 4.15: The ROUGE metrics between output of model and reference output.

Type of Summarization	CZ	Any
Full Summary	0.312 (78)	0.092 (23)
Summary with Reservations	0.236 (59)	0.084 (21)
Nonsensical Summary	0.044 (11)	
Copied Input	0.232 (58)	

Table 4.16: Results of manual check of outputs by human.

After evaluation of the performances of the models, **meta-llama/Llama-2-13b-chat-hf** is chosen as the most suitable model. Although the output was sometimes generated in English, the model generates an output in sentences, as it is necessary for generate the Hospital Course of Dismissal Report. Also the output is not generated indefinitely, rather, the model itself evaluates when it finishes. And also according to the manual check of the individual models recorded in the tables 4.7, 4.10, 4.13 and 4.16, the meta-llama/Llama-2-13b-chat-hf model is the most suitable model for fine-tuning.

4.4 Fine-tuning of selected model

For finetuning was choosed **meta-llama/Llama-2-13b-chat-hf** model as suitable model based on test in Subsection 4.3.3.

Finetuning of this model is done by script **model_training.py**, written in Python and use Transformers and Peft libraries. Functions are executed in the following sequence:

1. All necessary libraries for working with models, datasets, and fine-tuning are imported and script logs into Hugging Face account.
2. An **AutoTokenizer** is initialized for the model meta-llama/Llama-2-13b-chat-hf and a special pad_token is added.
3. An **AutoModelForCausalLM** is initialized with settings for 4-bit quantization, which reduces the model's memory requirements.
4. The dataset is loaded from a JSON file using the **load_dataset** function. The **format_instruction** function creates a prompt for the model as it is defined in Figure 4.12. The **generate_instruction_dataset** function reformats the input data into the desired format.
5. The **process_dataset** function shuffles the data randomly and reformats it using the **generate_instruction_dataset** function. The data is limited to 5000 records for training. The **gradient_checkpointing_enable** function enables gradient checkpointing, which reduces memory usage during training. The **prepare_model_for_kbit_training** function prepares the model for training with quantization.
6. The **LoraConfig** sets the parameters for Low-Rank Adaptation. The **TrainingArguments** defines the parameters for the training process, such as batch size, number of epochs, learning rate, and more.
7. The **SFTTrainer** is initialized with the model, dataset, tokenizer, and training arguments. The fine-tuning process is started using the **fine_tuning.train** function. Finally, the fine-tuned model and tokenizer are saved to the specified directory.

The dataset was divided into testing dataset and training dataset. The testing dataset contains 250 samples that are used for testing individual models, where the given results decide on the correct choice of parameters or improvements of training dataset. The same 250 samples of the testing dataset were used to select a suitable model. This process is described in the Subsection 4.3.3. Model is trained with 1 epoch and the size of the training dataset is set to 5000 samples, which could not be observed in the case of first version of dataset, described in Subsection 4.4.2, due to insufficient number of samples.

As in the case of finding a suitable model, as well as in the case of training the model, the table that records the results of the manual check is divided into categories. The **Bad output** category includes outputs that contained an exact copy of the input or errors, for example word cycling, an inappropriate structure, or a outputs are without letters (only contained white-space characters). **Summary with missing informations** is the category to which outputs with a lack of information belong. **Good Summary** is a category that includes outputs that can be used as a Hospital Course. In the category **Summary with Hallucinations**, it was recorded whether the model generated the given output with additional information, which was not found in the reference output or in the input. The given number in the field of the table represents a part of the whole, where the whole is 250 samples and the number of occurrences is indicated in the bracket. One of these tables is Table 4.19.

4.4.1 Prompt

In the training process is also used prompt defined in with slice difference. To the prompt, label is added to learn model how should it continue to generate text. This added information is just answer on input. Prompt for learning use same instruction as is defined in Figure 4.8 .

```

<s>
  [INST]
    <<SYS>>\n
      You are a helpful, respectful and honest assistant.
      Always answer as helpfully as possible, while being safe.
      Your answers should not include any harmful, unethical, racist,
      sexist, toxic, dangerous, or illegal content.
      Please ensure that your responses are socially unbiased
      and positive in nature.\n\nIf a question does not make any sense,
      or is not factually coherent, explain why instead of answering
      something not correct. If you don't know the answer to a question,
      please don't share false information.\n
    <</SYS>> \n\n
    {instruction + input}
  [/INST]
    Souhrn v českém jazyce ve formě
    celých vět o onemocnění a denním průběhu:
    {output}
</s>

```

Figure 4.12: Prompt for learning process of model.

4.4.2 Dataset improvements

LoRA method, described in Subsection 2.3, is used to reduce the computational complexity of the fine-tuning process. For the fine-tuning of the models described in this subsection, the default values of LoRA parameters are chosen. **Rank** is set to **16** and **Alpha** is set to **64**. Experimenting with different values of LoRA parameters is described in Subsection 4.4.3.

First version of dataset

As a first version of dataset for fine-tuning the suitable model is used dataset, which was also used for finding suitable model for fine-tuning. This dataset is described in Subsection 4.3.1. To be able used 250 samples for testing, the 4985 samples is used to fine-tuning the model.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.5053	0.2893	0.3503
ROUGE-2	0.3863	0.2059	0.2520
ROUGE-L	0.5032	0.2880	0.3489

Table 4.17: The ROUGE metrics between output of model and input text to model.

ROUGE	Precision	Recall	F1-score
ROUGE-1	0.2667	0.3818	0.2948
ROUGE-2	0.1511	0.2125	0.1687
ROUGE-L	0.2634	0.3773	0.2911

Table 4.18: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.056 (14)
Summary with Hallucinations	0.404 (101)
Summary with missing informations	0.232 (58)
Bad output	0.308 (77)

Table 4.19: Results of manual check of outputs by human.

Output of model
<p>Souhrn v českém jazyce celými větami o onemocnění a denním průběhu: Pacientka přijata k provedení diag. koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky normální nález, EF LK 60%-65%. Výkon i hospitalizace bez komplikací. Telemetricky sinusový rytmus, tlakově stabilní. Dimise ve stabilizovaném stavu.</p> <p>Operace: 09.03.2022 Operační diagnóza: Ischemická choroba srdečna St.p. myokard. infarktu Operační výkon: Coronarografia</p>

Figure 4.13: Output from model for input in Figure 3.2, which has label in Figure 3.3.

In Figure 4.13, it is obvious that the output has an incorrect form of summarization. This is due to the fact that the reference output, based on which the model was trained, contains other parts of the Dismissal Report, not just the Hospital Course, which led to model generate this incorrect form. During the manual check, however, these parts were not taken into account, and thanks to this, the correctness of the model to generate the text of the course of hospitalization could be assessed. Based on this, it was evaluated that the model also generated correct outputs, but the model most often generated output where it invented additional information as is shown in Table 4.19. Therefore, reducing the number of outputs that contains incorrect information is the goal of further improvement of the dataset.

Second version of dataset

In this version of dataset was done some improvements as anonymizing doctor names and precise extraction of Hospital course from dismissal report.

Script for creating of first version of dataset is described in Subsection 4.3.1. For second version of dataset is the **discharge_summary_extract_label** function improved to isolate and clean the portion of the text that describes the course of hospitalization, excluding certain subsections and irrelevant information. Also the **replace_doc** function is added to anonymize doctor names in the text by replacing them with the word „doctor“.

Metric Description	Count
Total samples	5557
Content length below 4096	5266
Content length higher than 4096	291

Table 4.20: Evaluation of sample sizes based on model content length.

These changes were not only reflected in the higher quality output from the model, but also affected the number of samples with which the model can be trained as can be seen in the Table 4.20, when compared with the Table 4.3. The main credit goes to the improvement of the label extraction.

Metric	Precision	Recall	F1-score
ROUGE-1	0.5097	0.2751	0.3393
ROUGE-2	0.4018	0.1968	0.2487
ROUGE-L	0.5075	0.2737	0.3377

Table 4.21: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.2740	0.3898	0.3088
ROUGE-2	0.1584	0.2400	0.1819
ROUGE-L	0.2708	0.3860	0.3055

Table 4.22: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.096 (24)
Summary with Hallucinations	0.384 (96)
Summary with missing informations	0.336 (84)
Bad output	0.184 (46)

Table 4.23: Results of manual check of outputs by human.

Output of model

Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu:
Pacientka přijata k provedení selektivní koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky normální nález na koronárních tepnách. Výkon proběhl nekomplikovaně, zápěstí po intervenci klidné. Telemetricky sinusový rytmus, bez záchytu arytmií. Dimise do ambulantní péče ve stabilizovaném stavu. Odvoz vlastní.

Figure 4.14: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Based on the results from manual check, which are showed in Table 4.23, the model generates less bad outputs. It is caused by a cleaner label and thus the model's output structure is clearer. There were also several outputs that summarized the Hospital course in a high quality without making up information or without missing essential information. Also, the Table 4.22 shows a greater similarity between the generated outputs by model and the reference outputs than in the previous version of the dataset, which has results in the Table 4.18.

Third version of dataset

The third and final version of the dataset is improved by deleting abbreviations. These abbreviations appear in the text together with their full word form. However, the model does not have to be trained that it is the same thing from the point of view of meaning, and the model can conclude that it is two different things. Therefore, replacing these abbreviations with their full word form leads to better results because the model now knows that it is the same thing.

The `replace_other_words` function is designed to replace specific abbreviations in a given text with their corresponding full words. This help in making the text more understandable by expanding commonly used abbreviations. Evaluation of sample sizes based on model content length is the same as it was in the case of the second dataset, these result are shown in the table 4.20.

Metric	Precision	Recall	F1-score
ROUGE-1	0.1890	0.4216	0.2453
ROUGE-2	0.1086	0.2615	0.1421
ROUGE-L	0.1874	0.4182	0.2432

Table 4.24: ROUGE metrics between input and reference output of dataset.

It can be seen that the changes in the dataset had a positive effect, as the results of the ROUGE metrics between input and reference output improved, when comparing the Tables 4.24 and 4.4. There is still low Precision value in the table 4.24 and thus the model will probably still generate a large amount of summarization with hallucinations.

Metric	Precision	Recall	F1-score
ROUGE-1	0.5406	0.2230	0.2991
ROUGE-2	0.4503	0.1698	0.2301
ROUGE-L	0.5389	0.2225	0.2983

Table 4.25: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.3334	0.3093	0.3075
ROUGE-2	0.2129	0.1997	0.1973
ROUGE-L	0.3303	0.3069	0.3049

Table 4.26: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.124 (31)
Summary with Hallucinations	0.372 (93)
Summary with missing informations	0.304 (76)
Bad output	0.2 (50)

Table 4.27: Results of manual check of outputs by human.

Output of model
<p>Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení selektivní koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky normální nález, t.č. bez význ. stenozy. Výkon i hospitalizace bez komplikací. Telemetricky sinusový rytmus, tlakově stabilní. Dimise ve stabilizovaném stavu.</p>

Figure 4.15: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Although, based on the comparison of the example in Figure 4.15 and the example in Figure 4.14, it shows that the model trained with the second version of the dataset generates better output.

Third version of the dataset has the highest number of good summaries compared to the first and second versions of the dataset. Changing the abbreviations to the full word form had a positive effect on the model, if only the number of good summaries is taken into account. On the other hand, there was also an increase in erroneous statements compared to the second version of the dataset. This version of the dataset, based on the results from the Tables 4.5, 4.6, 4.7, is used in the next steps of training and experimentation.

4.4.3 Experiments with LoRA training parameters

The aim of this experimentation is to find the values of the Rank and Alpha parameters, which leads to models with better results by using LoRA method. LoRA with Rank = 16 and Alpha = 64, used in the Subsection 4.4.2, train a model that creates behavior with some hallucinations and thus behaviors with some fabrication of additional false information. The solution to this can be to use the original model to a greater extent and thereby reduce the

effect of fine tuning. This is done by decreasing the value of the Alpha parameter. Also, experimenting with other values of LoRA parameters is evaluated in this subsection.

Rank 16, Alpha 16

Metric	Precision	Recall	F1-score
ROUGE-1	0.4072	0.2746	0.3042
ROUGE-2	0.3076	0.1880	0.2132
ROUGE-L	0.4051	0.2729	0.3024

Table 4.28: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.3474	0.3330	0.3305
ROUGE-2	0.2282	0.2189	0.2179
ROUGE-L	0.3446	0.3306	0.3280

Table 4.29: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.148 (37)
Summary with Hallucinations	0.268 (67)
Summary with missing informations	0.384 (87)
Bad output	0.236 (59)

Table 4.30: Results of manual check of outputs by human.

Output of model
<p>Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení selektivní koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky normální nález na koronárních tepnách, EF 60%. Výkon proběhl nekomplikovaně, zápěstí po intervenci klidné. Telemetricky po celou dobu hospitalizace sinus. rytmus, bez záchytu arytmii. Pacientku propouštíme v KP kompenzovaném stavu. Odvoz vlastní.</p>

Figure 4.16: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Rank 4, Alpha 16

Metric	Precision	Recall	F1-score
ROUGE-1	0.4533	0.1800	0.2441
ROUGE-2	0.3818	0.1394	0.1914
ROUGE-L	0.4515	0.1795	0.2434

Table 4.31: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.2775	0.2438	0.2447
ROUGE-2	0.1747	0.1596	0.1574
ROUGE-L	0.2738	0.2417	0.2423

Table 4.32: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.028 (7)
Summary with Hallucinations	0.08 (20)
Summary with missing informations	0.136 (34)
Bad output	0.752 (188)

Table 4.33: Results of manual check of outputs by human.

Output of model
Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení diag. koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky bez význ. stenozy, řešení jako FIS, subj. bez obtíží. Telemetricky sinusový rytmus, tlakově stabilní. Dimise ve stabilizovaném stavu.

Figure 4.17: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Rank 4, Alpha 4

Metric	Precision	Recall	F1-score
ROUGE-1	0.3684	0.1546	0.2092
ROUGE-2	0.3227	0.1250	0.1711
ROUGE-L	0.3664	0.1540	0.2082

Table 4.34: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.1759	0.1636	0.1591
ROUGE-2	0.1065	0.1009	0.0985
ROUGE-L	0.1744	0.1623	0.1579

Table 4.35: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.028 (7)
Summary with Hallucinations	0.028 (7)
Summary with missing informations	0.108 (27)
Bad output	0.836 (209)

Table 4.36: Results of manual check of outputs by human.

Output of model
<p>Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení diag. koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky bez stenozы, EF LK 60-65%. Echokardiograficky EF LK 60-65%, porucha diastol. relaxace LK, lehká sym. hypertrofie LK, bez lokálních poruch kinetiky, bez význam. chlopen.vad. Telemetricky sinusový rytmus, bez záchytu arytmii. Pacient propuštěna ve stabilizovaném stavu.</p>

Figure 4.18: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Model, trained by LoRA method with Rank set to 16 and Alpha set to 16, provide more good summaries based on manual check results shown in Table 4.30 and compare to Tables 4.33, 4.36. The reason is that setting the Rank parameter to a value of 4 means that the model will not learn the summarization sufficiently. Also, example in Figure 4.16 is better than example in Figure 4.15. However, there is also more frequent generation of bad output.

4.5 Inference of fine-tuned model

In this section, the results of experimentation with different values of the inference parameter of the model will be shown. Based on them, the parameters with which the tool for demonstrating the functionality of the model will run the fine-tuned model are selected. This approach and its implementation will also be described in this section.

4.5.1 Experiments with Inference parameters

The experiments refer to the parameters Top_p and Temperature. These parameters are described in the Subsection 2.4.1.

Temperature 1.0, Top_p 0.9

Metric	Precision	Recall	F1-score
ROUGE-1	0.2300	0.1022	0.1300
ROUGE-2	0.1323	0.0560	0.0718
ROUGE-L	0.2281	0.1013	0.1290

Table 4.37: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.1638	0.1617	0.1531
ROUGE-2	0.0720	0.0689	0.0669
ROUGE-L	0.1609	0.1591	0.1505

Table 4.38: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.0 (0)
Summary with Hallucinations	0.476 (119)
Summary with missing informations	0.068 (17)
Bad output	0.456 (114)

Table 4.39: Results of manual check of outputs by human.

Output of model

Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient s ICHS, s rizikovou anamnézou (sílnou rodinnou anamnézou - strýc s rizikovou anamnézou - CHSS), objednan na cestu RZP jako těžká polycystická ledvina. Po příjmu provedeno echokardiografické vyšetření, kde známky systolické dysfunkce. Provedena nekomplikovaně skg s nálezem význ. postižení RIA do 30% s lehkou restenozou distální RC, jinak normální nález na koronárních tepnách, EF 55%. Telemetricky sinusový rytmus, tlakově stabilní, pacient oběhově kompenzován. Obtíže klidové, spontánní rekoronarační terapie, tlaková mediika. Dimise do ambulantní péče ve stabilizovaném stavu.

Figure 4.19: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Temperature 0.7, Top_p 0.9

Metric	Precision	Recall	F1-score
ROUGE-1	0.3230	0.1209	0.1679
ROUGE-2	0.2518	0.0858	0.1206
ROUGE-L	0.3202	0.1199	0.1664

Table 4.40: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.1890	0.1673	0.1661
ROUGE-2	0.0978	0.0920	0.0889
ROUGE-L	0.1854	0.1642	0.1630

Table 4.41: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.008 (2)
Summary with Hallucinations	0.44 (110)
Summary with missing informations	0.144 (36)
Bad output	0.408 (102)

Table 4.42: Results of manual check of outputs by human.

Output of model
<p>Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení selektivní koronarografie před zařazením na waiting list před transplantací ledviny. Výkon nekomplikovaně, s nálezem - 70% stenoza distální ACD, řešeno PCI s impl. DES, trvá 90% stenoza proximální RIA, řešeno PCI s impl. DES, jinde bez význ. stenozy či restenozy. Po nekomplikovaném průběhu dimise ve stabilizovaném stavu. Rozhovor s pacientem: Informována o výsledku SKG, souhlasí s výkonem.</p>

Figure 4.20: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Temperature 0.1, Top_p 0.9

Metric	Precision	Recall	F1-score
ROUGE-1	0.3151	0.1235	0.1687
ROUGE-2	0.2666	0.0970	0.1336
ROUGE-L	0.3141	0.1231	0.1682

Table 4.43: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.1818	0.1515	0.1550
ROUGE-2	0.1130	0.0993	0.0988
ROUGE-L	0.1793	0.1501	0.1533

Table 4.44: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.028 (7)
Summary with Hallucinations	0.0 (0)
Summary with missing informations	0.256 (64)
Bad output	0.716 (179)

Table 4.45: Results of manual check of outputs by human.

Output of model
Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení selektivní koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky normální nález na koronárních tepnách, EF 60%. Výkon proběhl nekomplikovaně. Dimise do ambulantní péče ve stabilizovaném stavu.

Figure 4.21: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Temperature 0.4, Top_p 0.5

Metric	Precision	Recall	F1-score
ROUGE-1	0.3009	0.1234	0.1654
ROUGE-2	0.2530	0.0982	0.1320
ROUGE-L	0.2995	0.1229	0.1647

Table 4.46: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.1874	0.1522	0.1588
ROUGE-2	0.1158	0.1006	0.1017
ROUGE-L	0.1855	0.1512	0.1575

Table 4.47: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.056 (14)
Summary with Hallucinations	0.072 (18)
Summary with missing informations	0.208 (52)
Bad output	0.664 (166)

Table 4.48: Results of manual check of outputs by human.

Output of model

Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení selektivní koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky koronární tepny bez význ. stenozy, EF 60%. Výkon proběhl nekomplikovaně, zápěstí po intervenci klidné. Dimise ve stabilizovaném stavu.

Figure 4.22: Output from model for input in Figure 3.2, which has label in Figure 3.3.

Temperature 0.4, Top_p 0.2

Metric	Precision	Recall	F1-score
ROUGE-1	0.3048	0.1184	0.1619
ROUGE-2	0.2571	0.0928	0.1278
ROUGE-L	0.3033	0.1180	0.1614

Table 4.49: The ROUGE metrics between output of model and input text to model.

Metric	Precision	Recall	F1-score
ROUGE-1	0.1796	0.1471	0.1524
ROUGE-2	0.1115	0.0948	0.0961
ROUGE-L	0.1777	0.1457	0.1508

Table 4.50: The ROUGE metrics between output of model and reference output.

Type of Summarization	Occassions
Good Summary	0.052 (6)
Summary with Hallucinations	0.008 (2)
Summary with missing informations	0.268 (67)
Bad output	0.672 (168)

Table 4.51: Results of manual check of outputs by human.

Output of model

Souhrn v českém jazyce ve formě celých vět o onemocnění a denním průběhu: Pacient přijata k provedení selektivní koronarografie před zařazením na waiting list před transplantací ledviny. Koronarograficky normální nález na koronárních tepnách, EF LK 60%. Výkon proběhl nekomplikovaně. Dimise ve stabilizovaném stavu.

Figure 4.23: Output from model for input in Figure 3.2, which has label in Figure 3.3.

From the testing that was documented in this section, the parameters that were used in the generation of the results in the Section 4.4.3 were selected for inference. It is based on results from manual check by human for each of tested combination of parameters. Parameters used in Section 4.4.3 provide better results. These are the parameters - Temperature: 0.4, Top_p: 0.9 and Top_k: 40.

Output of ChatGPT 3.5

Souhrn Onemocnění a Denního Průběhu

Onemocnění

Pacient byl přijat k provedení diagnostické koronarografie před zařazením na čekací listinu pro transplantaci ledviny. Echokardiografické vyšetření srdce ukázalo ejekční frakci levé komory (EF LK) mezi 60-65 %, poruchu diastolické relaxace levé komory, lehkou symetrickou hypertrofii levé komory, bez lokálních poruch kinetiky a bez významných chlopenních vad. Subjektivně pacient neguje bolesti na hrudi, dušnost, palpitace a předsynkopální či synkopální stavy.

Denní Průběh

10.3.2022 8:40 Zapsal DOKTOR:

Subjektivně (S): Pacient je bez obtíží.

Objektivně (O): Pacient je afebrilní, má eupnoi (normální dýchání), poslechové nálezy na srdci (AS reg.) jsou pravidelné, dýchání je alveolární a čisté. Břicho je měkké, nebolestivé. Dolní končetiny jsou bez otoků, lýtka jsou volná, akra (konečky prstů) jsou dobře prokrvená. Zápěstí po skg (pravděpodobně po zákroku) je klidné. Tělesná teplota (telemetry): Sinusový rytmus (SR) 61/min.

Pacient byl propuštěn (dimise).

Figure 4.24: Output from ChatGPT 3.5 for input 3.2, which has label as 3.3.

For comparison, the text was also generated by the GPT 3.5 model. The final fine-tuned LLaMA model tries to imitate the performance of a doctor, as is shown in Figure 4.16. It is not detailed and it can be seen that some factual information does not fit.

GPT, on the other hand, expanded the input text more and did not summarize it as expected, as it is shown in Figure 4.24. On the other hand, since it is not fine-tuned to imitate the summarization as the doctor does, it did not even lead to him inventing the information.

4.5.2 Demonstration tool

This tool consists of two parts. It is a client-side web application and a server-side application. The server-side application has the task of running the selected model and processing inputs from the client for the model. The Client-side application offers the setting of the model type that should generate the text as well as the type of output text the user wants. Communication between these two applications is done through an API, where the sequence of individual requests and responses is shown on Figure 4.25. The fact that the server communicates exclusively only with the given client application is done by using an API key.

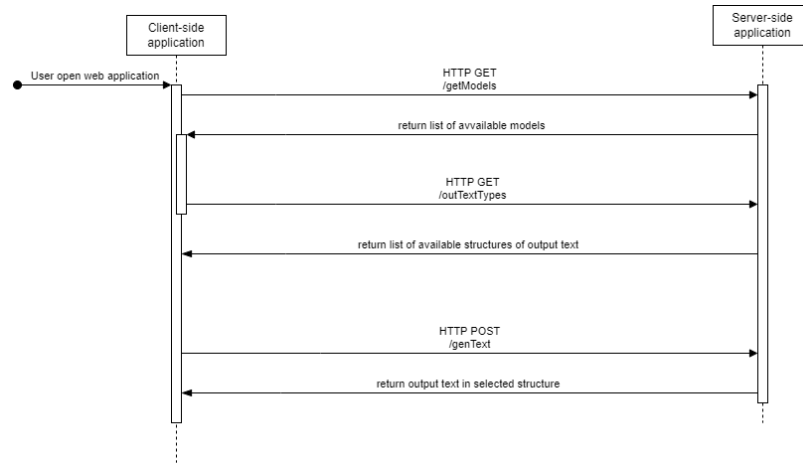


Figure 4.25: Sequence diagram of API communication between client-side application and server-side application.

Client-side application

The client application is written in Javascript and uses the React.js library to implement the necessary components. The user has 2 menus available. One to choose a model that is available on the server and can generate text. the second menu serves to select the content of the displayed text. In the case of the second menu, the user can set that the output text will contain only the Hospital course generated by the model, or the output text will also contain other parts of the Admission Report that should appear in the Dismissal report.

Next, the application provides a text field for entering the entire Admission Report. Such a field is also available for the Decor entry. However, just one hospitalization can contain more than just one Decors document, the user has the option to add additional text fields for the Decors entry. The resulting text received from the server application is displayed in the text field under the „Generate text“ button. This button sends a request to the server to generate the output text, and this request contains all the necessary inputs that the model needs.

The screenshot shows a web application interface. On the left is a sidebar with the heading "Vyberte model" and a button labeled "Llama2_13B_v1". The main area has a heading "Vyberte typ výstupního textu" and a button labeled "Příběh hospitalizace". Below this are four text input fields. The first field is labeled "Příjmová správa". The second and third fields are both labeled "Dekurz". The fourth field has two buttons above it: "Přidat pole" and "Generovat text".

Figure 4.26: Client's application.

Server-side application

The server part is written in Python and the Flask framework is used for its implementation. The server-side application offers a list of available models and types of resulting text structures. In addition, this application ensures that the fine-tuned model is up and running. The inference parameters are fixed to the values that had the best results in the experiments in the Subsection 4.5.1. From the inputs from the client-side application, the appropriate data is extracted for input into the model. This extraction takes place using functions that were also used to create the dataset for model training and are described in subsections 4.3.1 and 4.4.2.

Conclusion

As part of this thesis, it was possible to create a model that, to a certain degree of success, creates part of the medical documentation in the Czech language. Several models that would be suitable for the given task were tested, and the best of them, which generated text in the Czech language, was additionally fine-tuned. By choosing the right parameters of the training process or inference, it was possible to improve the success of generating the Hospital Course of Dismissal Report correctly compared to what was generated by the not finetuned model. Several possible combinations of individual parameters were examined and the tests resulted in a choice that led to the gradual improvement of the model.

It was successfully implemented a demonstration tool consisting of two applications that simply provides the result of the finetuned model to user.

The success of the model in generating a correct speech is relatively low for its use in practice by doctors. Another possibility to increase the success in correctly generating the Hospital Course of Dismissal Report by model, is to use a model with a higher number of parameters, which is more computationally demanding and therefore was not used in this thesis. Another possibility is a dataset that will contain better data, especially regarding the label written by the doctor. The current dataset contains many grammatical errors and strange sentence constructions, as well as a large number of medical abbreviations, which complicate the correct fine-tuning of the model to generate the correct output. The last but not least improvement is to provide into the model more documentation. Thanks to this, the model would have more information so that it can more accurately generate output, but this again requires more memory on the graphics cards on which the calculation takes place.

Another possible way to improve the correctness of output is the combination of artificial intelligence methods. As part of the thesis, it was planned that expert systems could be used in the framework of combining methods. The right combination of an expert system with a Large Pre-Trained Language Model can lead to the generation of other parts of the Dismissal Report, where the model definitely needs medical knowledge. In such a case, this medical knowledge would be provided by an expert system and the Large Pre-Trained Language Model would generate it in text form.

Bibliography

- [1] *Vyhláška č. 98/2012 Sb.* [Ministerstvo zdravotnictví České republiky]. 2012. Dostupné online: <https://www.zakonyprolidi.cz/cs/2012-98>.
- [2] ABRAHAM, A. Rule-based Expert Systems. In: SYDENHAM, P. H. and THORN, R., ed. *Handbook of Measuring System Design*. ISBN: 0-470-02143-8: John Wiley & Sons, Ltd., 2005, p. Place the page numbers here. Oklahoma State University, Stillwater, OK, USA.
- [3] AINSLIE, J., LEE THORP, J., JONG, M. de, ZEMLYANSKIY, Y., LEBRÓN, F. et al. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. Available at: <https://arxiv.org/abs/2305.13245>.
- [4] BANOULA, M. What is Perceptron: A Beginners Guide for 2023. *Simplilearn*. May 2021. Accessed: 2024-05-17. Available at: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron>.
- [5] CHANDRA, R. Problem Decomposition and Adaptation in Cooperative Neuro-Evolution. january 2012. DOI: 10.26686/wgtn.16996780. Available at: https://openaccess.wgtn.ac.nz/articles/thesis/Problem_Decomposition_and_Adaptation_in_Cooperative_Neuro-Evolution/16996780.
- [6] CONTENT EDITOR. *What is Inference: LLMs Explained*. February 2024. Accessed: 2024-07-22. Available at: <https://www.chatgptguide.ai/2024/02/29/what-is-inference-llms-explained/>.
- [7] COTOIA, A. Axon Terminal - The Definitive Guide | Biology Dictionary. *Biology Dictionary*. May 16 2020. Accessed: 2024-05-17. Available at: <https://biologydictionary.net/axon-terminal/>.
- [8] FLASINSKI, M. *Introduction to Artificial Intelligence*. 1st ed. Cham: Springer Nature, 2016. ISBN 3319400223.
- [9] GEEKSFORGEEKS. *ReactJS Introduction*. 2024. Accessed: 2024-07-22. Available at: <https://www.geeksforgeeks.org/reactjs-introduction/>.
- [10] GOLDBERG, Y. *Neural network methods for natural language processing*. San Rafael: Morgan Claypool Publishers, 2017. Synthesis lectures on human language technologies. ISBN 978-1-62705-298-6.
- [11] HOREV, R. BERT Explained: State of the art language model for NLP. *Towards Data Science*. 2018. Accessed: 2024-05-17. Available at:

- <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>.
- [12] HU, E. J., SHEN, Y., WALLIS, P., ALLEN ZHU, Z., LI, Y. et al. LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv.org*. Ithaca: Cornell University Library, arXiv.org. 2021. ISSN 2331-8422.
 - [13] HUGGING FACE. *Transformers*. 2024. Accessed: 2024-07-22. Available at: <https://huggingface.co/docs/transformers/index>.
 - [14] IBM DATA AND AI TEAM. *Understanding the different types of artificial intelligence* [<https://www.ibm.com/blog/understanding-the-different-types-of-artificial-intelligence/>]. 2023. Accessed: October 16, 2023.
 - [15] KALIRANE, M. *Gradient Descent vs. Backpropagation: What's the Difference?* January 2023. Accessed: 2024-05-17. Available at: <https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/>.
 - [16] KUSHAL DAS. *Introduction to Flask*. 2022. Accessed: 2024-07-22. Available at: <https://pythonforundergradengineers.com/flask-introduction/>.
 - [17] MANGRULKAR, S. and PAUL, S. *Parameter-Efficient Fine-Tuning using PEFT*. February 2023. Accessed: 2024-07-22. Available at: <https://huggingface.co/blog/peft>.
 - [18] MDN WEB DOCS. *What is JavaScript? - Learn web development*. 2024. Accessed: 2024-07-22. Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.
 - [19] MYSQL DOCUMENTATION TEAM. *MySQL 8.4 Reference Manual: 1.2.1 What is MySQL?* 2023. Accessed: 2024-07-22. Available at: <https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html>.
 - [20] NIEDERFAHRENHORST, A., HAKHAMANESHI, K. and AHMAD, R. Fine-Tuning LLMs: LoRA or Full-Parameter? An In-Depth Analysis with LLAMA-2. *Anyscale Blog*. September 2023. Online publication, Accessed: 2024-05-17. Available at: <https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-llama-2>.
 - [21] OKEWU, E., ADEWOLE, P. and SENNAIKE, O. Experimental Comparison of Stochastic Optimizers in Deep Learning. In: *Computational Science and Its Applications – ICCSA 2019*. Springer International Publishing, 2019, vol. 11623, p. 704–715. Lecture Notes in Computer Science. ISBN 9783030243074.
 - [22] PAASS, G. and GIESSELBACH, S. *Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media*. 1st ed. Springer Nature, 2023. Artificial Intelligence: Foundations, Theory, and Algorithms. ISBN 3031231899.
 - [23] PATEL, P. *Guide to fine-tuning LLMs using PEFT and LoRa techniques* [Webpage]. 2023. Available at: https://assets-global.website-files.com/640f56f76d313bbe39631bfd/64ac7af09311957ab6dbecc1_Peft%20finetuning.png.

- [24] PHUNG, V. and RHEE, E. A deep learning approach for classification of cloud image patches on small datasets. *Journal of Information and Communication Convergence Engineering*. january 2018, vol. 16, p. 173–178. DOI: 10.6109/jicce.2018.16.3.173.
- [25] PYTHON.ORG. *What is Python? Executive Summary*. 2024. Accessed: 2024-07-22. Available at: <https://www.python.org/doc/essays/blurb/>.
- [26] ROUT, C. and ALDOUS, C. How to write a research protocol. *Southern African Journal of Anaesthesia and Analgesia*. october 2016, vol. 22, p. 101–107. DOI: 10.1080/22201181.2016.1216664.
- [27] RUSSELL, S. J. S. J. *Artificial intelligence : a modern approach*. Third edition; Authorized adaptation from the United States edition 2010th ed. Boston ; London: Pearson, 2016. Prentice Hall series in artificial intelligence. ISBN 978-1-292-15396-4.
- [28] SHARMA, S., SHARMA, S. and ATHAIYA, A. Activation Functions in Neural Networks. *International Journal of Engineering Applied Sciences and Technology*. 2020, vol. 4, no. 12, p. 310–316. ISSN 2455-2143. Published Online April 2020 in IJEAST (<http://www.ijeast.com>).
- [29] STEVE, P. *Ludské telo: Ilustrovaný sprievodca štruktúrou, funkciami a poruchami [The Human Body]*. Ikar, a. s., 2008. ISBN 978-80-551-1731-7.
- [30] SUNDARARAJAN, N., SARATCHANDRAN, P. and LI, Y. *Fully Tuned Radial Basis Function Neural Networks for Flight Control*. Boston, MA: Springer US, 2002. The International Series on Asian Studies in Computer and Information Science. ISBN 1441949151.
- [31] TOGETHERCOMPUTER. *Togethercomputer/Llama-2-7B-32K-Instruct · Hugging Face*. 2024. Accessed: 2024-05-17. Available at: <https://huggingface.co/togethercomputer/Llama-2-7B-32K-Instruct>.
- [32] TOUVRON, H., LAVRIL, T., IZACARD, G., MARTINET, X., LACHAUX, M.-A. et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. Available at: <https://arxiv.org/abs/2302.13971>.
- [33] TOUVRON, H., MARTIN, L., STONE, K., ALBERT, P., ALMAHAIRI, A. et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. Available at: <https://arxiv.org/abs/2307.09288>.
- [34] YUKANG. *Yukang/Llama-2-13b-longlora-16k-ft · Hugging Face*. 2024. Accessed: 2024-05-17. Available at: <https://huggingface.co/Yukang/Llama-2-13b-longlora-16k-ft>.
- [35] YUKANG. *Yukang/LongAlpaca-12k · Hugging Face*. 2024. Accessed: 2024-07-14. Available at: <https://huggingface.co/datasets/Yukang/LongAlpaca-12k>.
- [36] YUKANG. *Yukang/LongAlpaca-13B · Hugging Face*. 2024. Accessed: 2024-05-17. Available at: <https://huggingface.co/Yukang/LongAlpaca-13B>.