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 **Bc. PATRIK PRICL**
AUTOR PRÁCE

SUPERVISOR **RNDr. MAREK RYCHLÝ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

153712

Institut:	Department of Information Systems (DIFS)
Student:	**Pricl 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

Requirements for the semestral defence:
Items 1 and 2.

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:	17.5.2024
Approval date:	30.10.2023

Abstract

The thesis consider with the use of pre-trained language models for summarizing medical documentation in the form of dismissal reports.

Abstrakt

Práca sa zaoberá využitím predtrénovaných jazykových modelov na sumarizáciu zdravotnej dokumentácie do formy prepúšťacích správ.

Keywords

NLP, Text sumarization, Artificial intelligence, Large language models

Kľúčové slová

Spracovanie prirodzeného jazyka, Sumarizácia textu, Umelá Inteligencia, Predtrénované jazykové modely

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 Pricl
May 17, 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
  - 1.2.3 K-Means Clustering . . . 12
  - 1.2.4 Q-Learning . . . 12
  - 1.2.5 Artificial Neural Network . . . 13

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
- 2.2 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 Evaluation techniques . . . 37

3 Assistant for Creating Medical Reports **39**
- 3.1 Dataset . . . 39
  - 3.1.1 Entities . . . 39
  - 3.1.2 Dismissal reports . . . 41
  - 3.1.3 Data for Supervised fine-tuning . . . 42
- 3.2 Assistant API Gateway . . . 44
  - 3.2.1 API . . . 44
  - 3.2.2 Ollama . . . 44

4 Implementation and testing **45**
- 4.1 Data preparation . . . 45
  - 4.1.1 1st version of dataset . . . 46
  - 4.1.2 2nd version of dataset . . . 46
  - 4.1.3 3rd version of dataset . . . 46

4.2 Suitable model 47
4.2.1 Model‘s variants 47
4.2.2 Prompt 47
4.3 Learning 48
4.4 Inference and API communication 49
4.4.1 Inference 49
4.4.2 API communication 50

Bibliography **52**

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 [21] . 11
1.4 Visual demonstration of clustering using the K-means algorithm. Image taken from [17] 12
1.5 Visual description of a human neuron. Image taken from [6]. 13
1.6 Visualization of a perceptron. Image taken from [4] 14
1.7 Graph of Bipolar step function. 16
1.8 Graph of Binary step function. 16
1.9 Graph of Identity. 17
1.10 Graph of Sigmoid function. 17
1.11 Graph of ReLU Activation Function. 18
1.12 Visualization of individual types of the neural network based on connection. Image taken from [5] 19
1.13 Progress of back-propagation algorithm. Image taken from [12] 21
1.14 Progress of finding minimum loss. Image taken from [12] 21
1.15 Visualization of CNN architecture. Image taken from [19]. 23
1.16 Process of prediction sequence of new words by Recurrent neural networks . Image taken from [16]. 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 [16]. 28
2.4 Token embeddings for Transformers, like BERT or GPT, contains position information duo to parallel computations of Transformers. Image taken from [16]. 29
2.5 Computation of a contextual embedding for a single token "mouse" by attention mechanism. Image taken from [16]. 31
2.6 Visual representation of multi-head attention. Image taken from [16]. 32
2.7 Scheme of stacking multiple transformer layers. Image taken from [16]. . . . 33
2.8 Transformer model predicts the next token in token's sequence. Image taken from [16]. 34
2.9 Comparation between word prediction by BERT and text generation by GPT. Image taken from [16]. 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 matrices. Image taken from [10]. 37

3.1 Entity relationship diagram of given dataset. 41

4.1 Simple schema of fine-tuning process of large language model. 45
4.2 Client‘s application. 50

Introduction

The expansion of artificial intelligence within the framework of natural language processing leads to its more frequent use in various fields of work. By using artificial intelligence, people can be relieved of redundant administration by having artificial intelligence manage it by itself. Medicine is one of these sectors. In the 1st chapter of this thesis, artificial intelligence as such will be discussed. The second chapter, the architecture and learning process of Large Pre-trained Language models will be discussed. In Third chapter, the dataset is shown and how to modify it for training purpuse. In forth chapter, implementation of training of Large Pre-trained language model is shown and evaluate.

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. [22]

Based on this, we divide artificial intelligence into [11]:

- **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 [11]:

- **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.

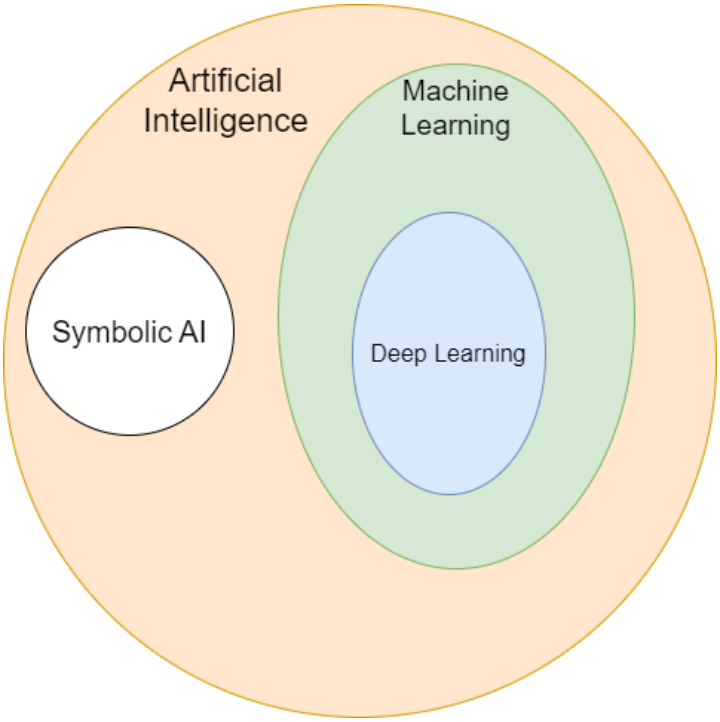

Figure 1.1: Venn diagram of Artificial Intelligence's subsets mentioned in this thesis.

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 [7]. 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]

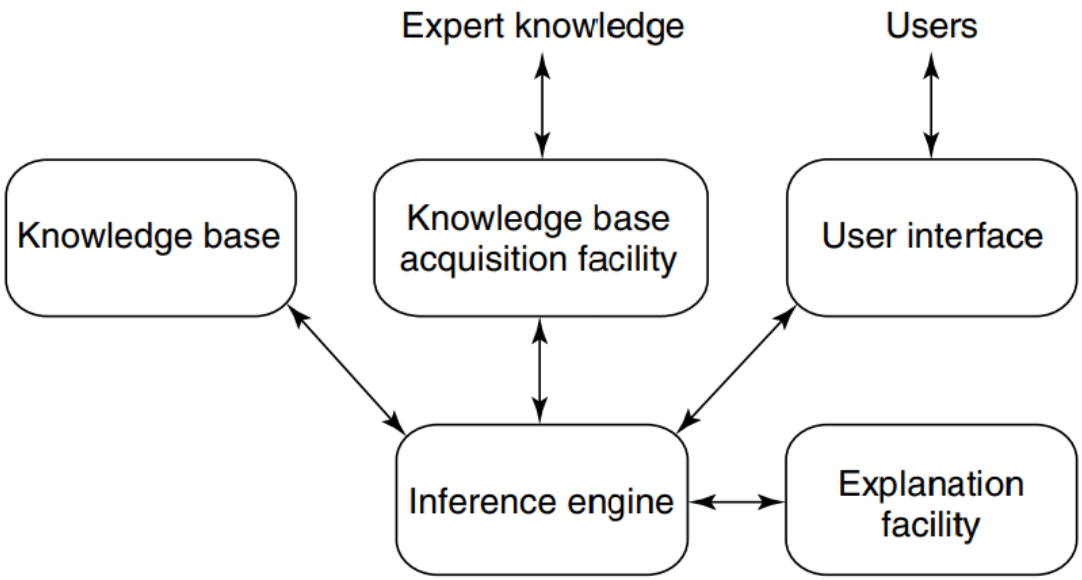

Figure 1.2: Design of Rule-based expert system. Image taken from [2].

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]

.

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.

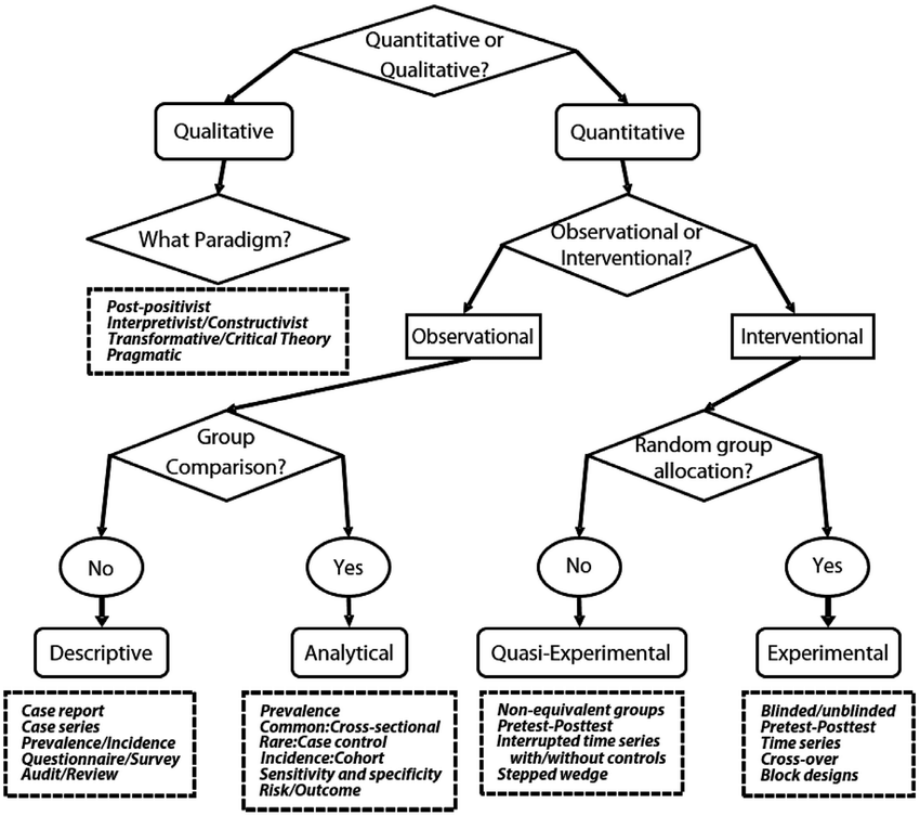

Figure 1.3: Diagram of Decision tree defining research designs. Image taken from [21]

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.

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.

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

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.

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

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

1.2.3 K-Means Clustering

This model uses unsupervised learning, which takes all the data and divides them into K clusters. The position of the cluster is mainly indicated by its centroid, which represents the point in the center of the cluster. Each cluster has its centroid positioned at the mean position of the data within that cluster. New data are included in the cluster whose center is located closest to them. The Euclidean distance or other distance function is chosen to calculate this distance. This process may visually looks like in the picture 1.4.

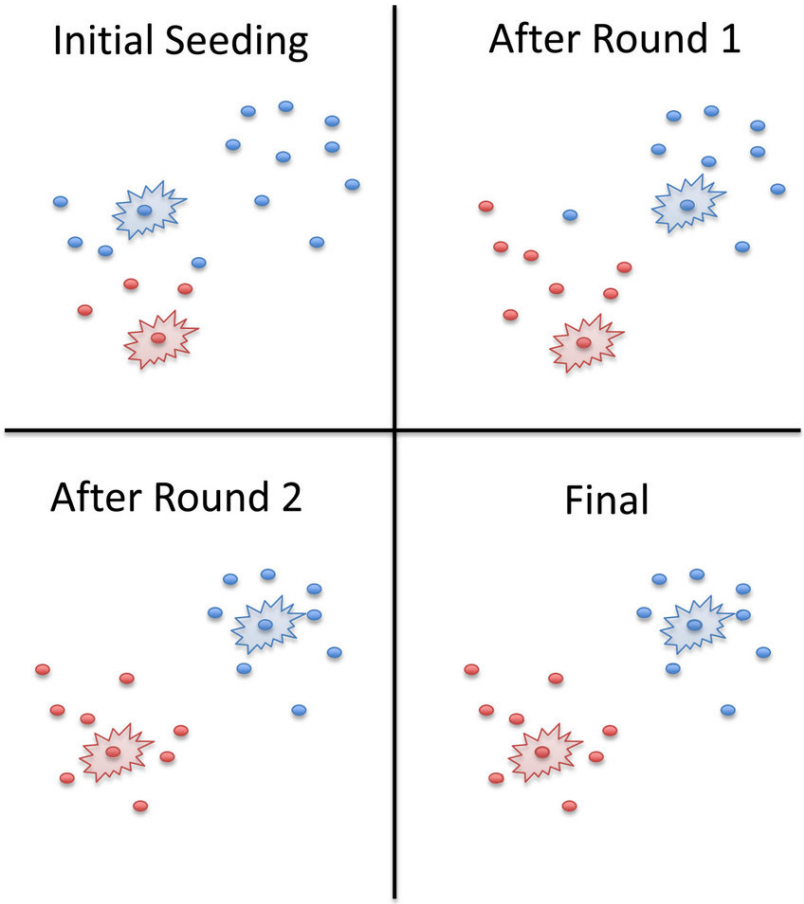

Figure 1.4: Visual demonstration of clustering using the K-means algorithm. Image taken from [17]

1.2.4 Q-Learning

Q-learning is a reinforcement learning algorithm. It is widely used to estimate an optimal strategy, where an agent needs to make decisions to maximize a reward. Usage of this type machine learning is in healthcare, autonomous cars or natural language processing.

The idea of the Q-Learning algorithm is that an agent interacts with a given environment to obtain data that have not been previously presented. The agent will then map the set

of states, actions, and rewards. This combination of state, action, and reward is called "quality". An agent concerned with the best immediate benefit would choose the action with the greatest immediate reward. However, considering the long-term impact of its actions, a better decision could be made.[20]

After the learning phase, the model creates a policy based on the quality to do the best long-term action for the current state.

1.2.5 Artificial Neural Network

Artificial Neural Networks are inspired by biology to imitate the human ability to learn. More precisely, neurons and their interconnections, which make up the brain and the entire nervous system, are a source of inspiration.

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 [24]. 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 [24]:

- 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.

,

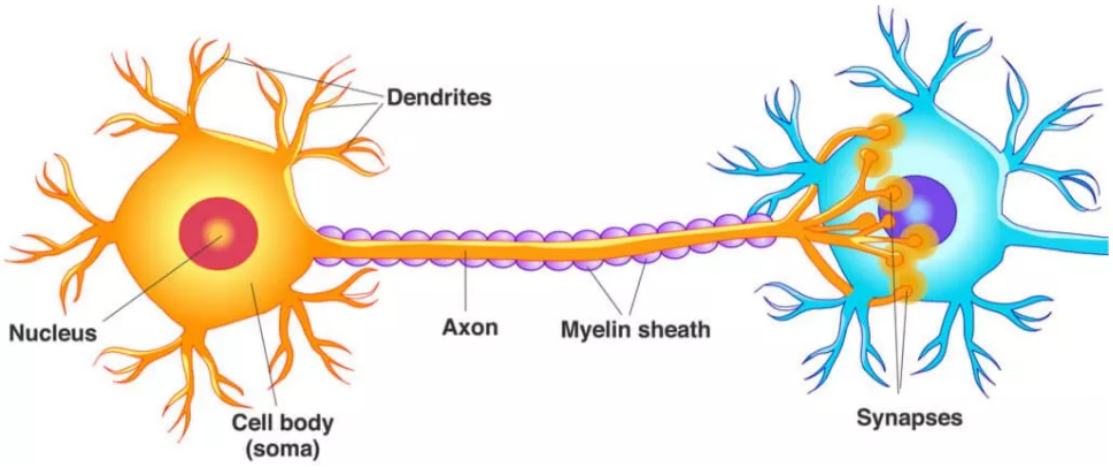

Figure 1.5: Visual description of a human neuron. Image taken from [6].

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.

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

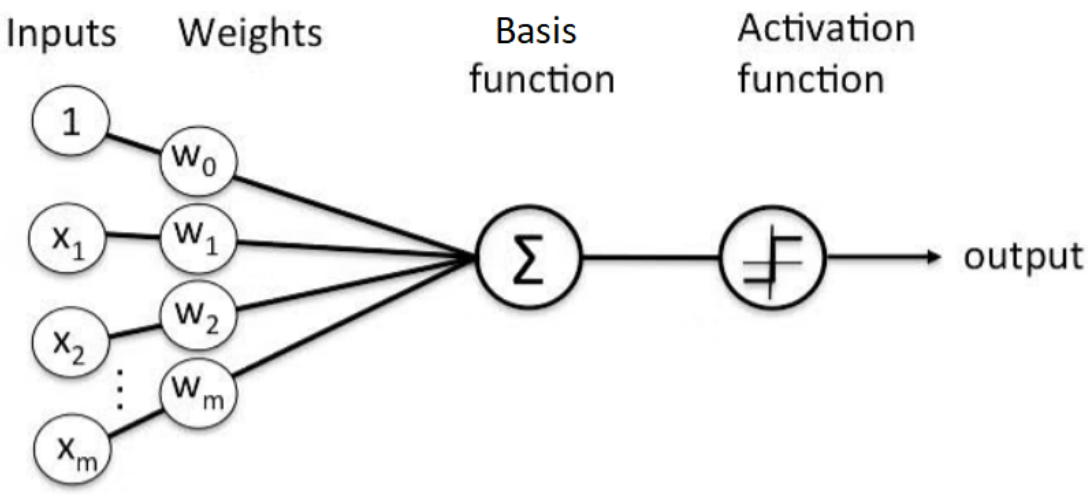

Figure 1.6: Visualization of a perceptron. Image taken from [4]

Basis function

In machine learning, two functions are defined as basis functions: the Linear basis function and the Radial basis function. [22].

- ***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.

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

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 1.6 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[7]. The coordinates of this point are stored in the parameter weight. For the calculation of this distance, the Euclidean distance is used [25]. Its mathematical description is 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.

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

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 [7].

- ***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 1.8. The function is also called as Binary step function and is used in the output layer 1.2.5 of the neural network 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 change in neuron‘s weights, basically meaning that the neural network learns nothing.

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

- ***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 1.2.5, so the neuron‘s weights are changed at both outputs.

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

- ***Linear function***: This function reacts more „smooth“ than the 2 previous functions [7]. The Basic linear function is called Identity and the result of the function is the same as the input as seen in the Figure 1.9. 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.

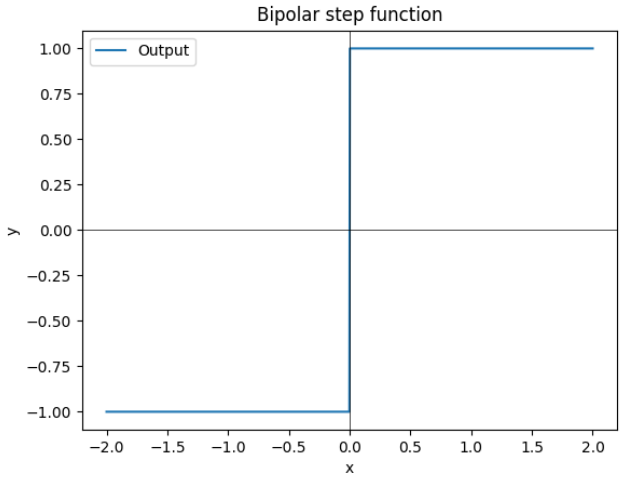

Figure 1.7: Graph of Bipolar step function.

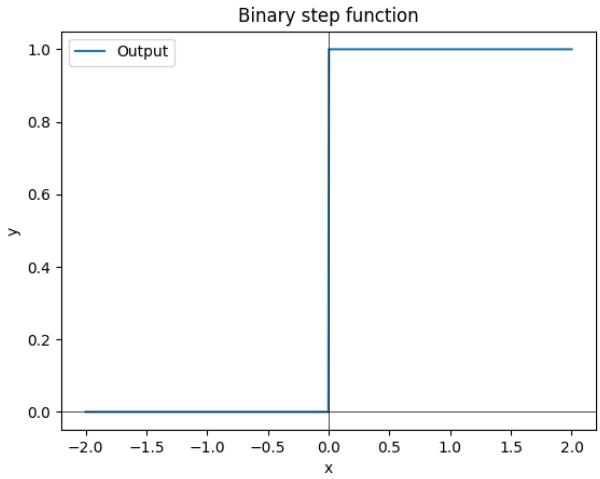

Figure 1.8: Graph of Binary step function.

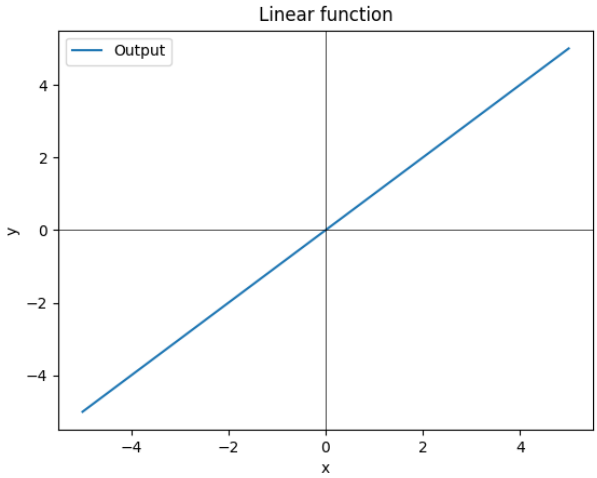

Figure 1.9: Graph of Identity.

- ***Sigmoid function***: This function is continuously differentiable and a smooth S-shaped function [23]. 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 1.10.

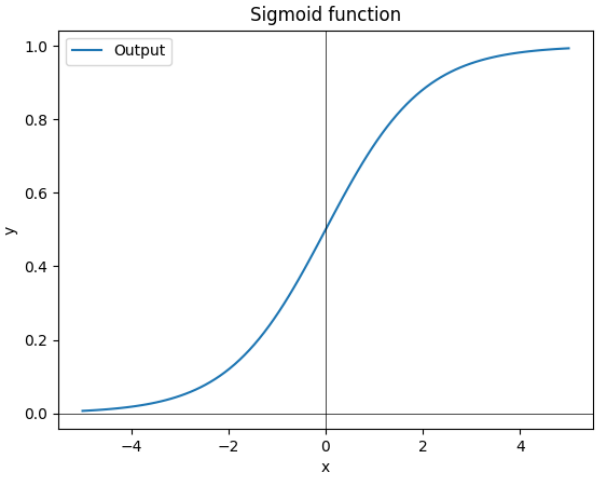

Figure 1.10: Graph of Sigmoid function.

- ***ReLU Activation Function***: ReLU stands for rectified linear unit and is a non-linear activation function which is widely used in neural network [23]. The function is based on a mathematical equation 1.7 and its behavior is illustrated in the Figure 1.11. This function is used in hidden layer 1.2.5 for various tasks from image classification to speech recognition.

$$f(x) = \max(0, x)\,. \tag{1.7}$$

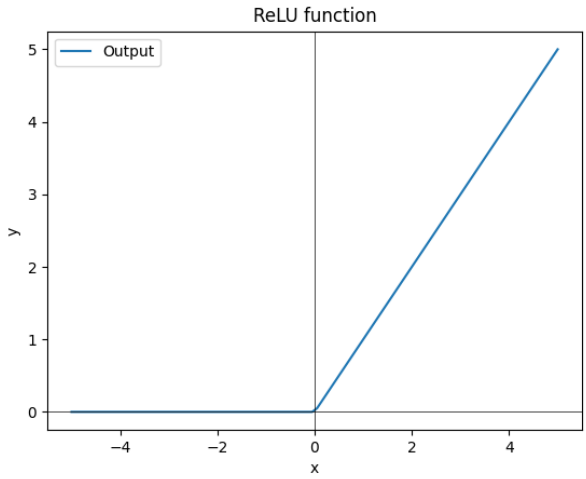

Figure 1.11: Graph of ReLU Activation Function.

- ***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.

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

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 [22]:

- ***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 1.12.

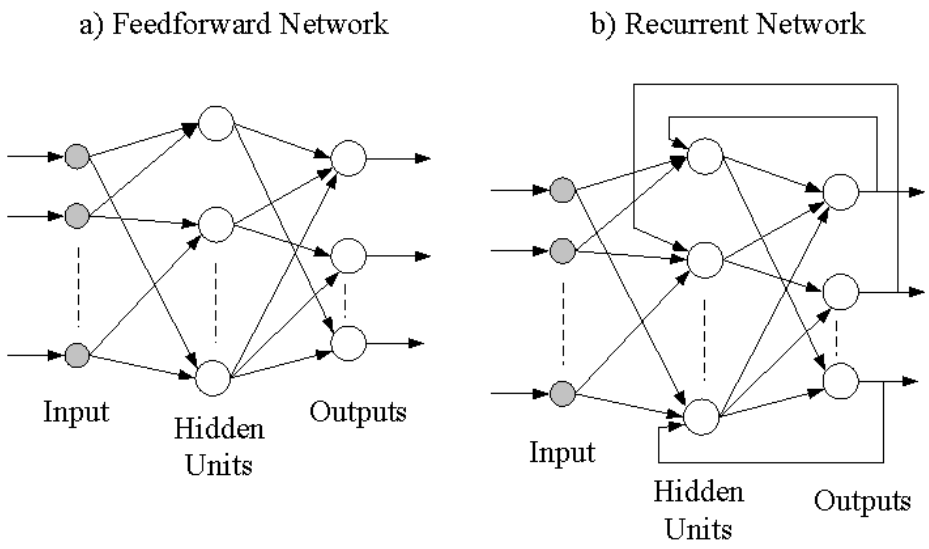

Figure 1.12: Visualization of individual types of the neural network based on connection. Image taken from [5]

Model Training

As mentioned in 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. [8]

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 [8]:

- ***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 as follows 1.9.

$$L(y_r, y_w) = \max(0, 1 - y_r y_w) \tag{1.9}$$

- ***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 as 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.

$$L = -y_w \log p(y_w) - (1 - y_w) \log(1 - p(y_w)) \tag{1.10}$$

- ***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 as 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.

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

- ***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 1.12, where n is number of training samples in Epoch.

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

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 [22]. 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 1.13.

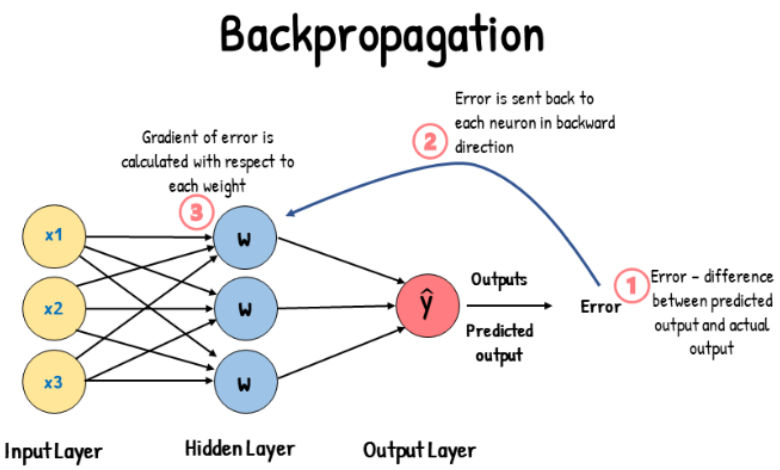

Figure 1.13: Progress of back-propagation algorithm. Image taken from [12]

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 1.14.

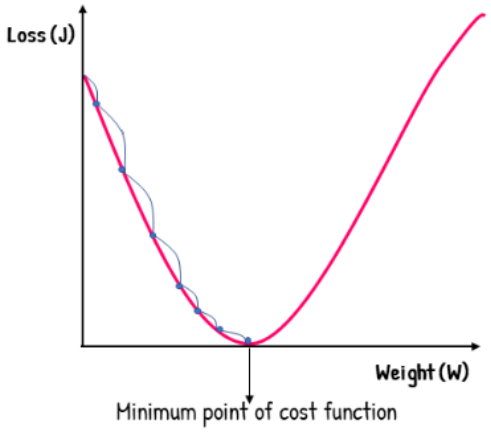

Figure 1.14: Progress of finding minimum loss. Image taken from [12]

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. [8]
- ***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. [15]
- ***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. [15]
- ***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. [15]

Neural Networks for Natural Language Processing

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.

One of this types are:

- ***Convolutional neural networks***, or CNN is neural network used mostly for solving problems include image type input's data. For process this type of data, Convolutional neural network have convolutional and pooling layers on the begging of network to convert input to better format for Feed-forward network.

  Convolutional layer is supposed to apply matrix filters to the input and identify indicative local predictors in a large structure [8].

  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** 1.2.5. This architecture is shown on Figure 1.15. 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.

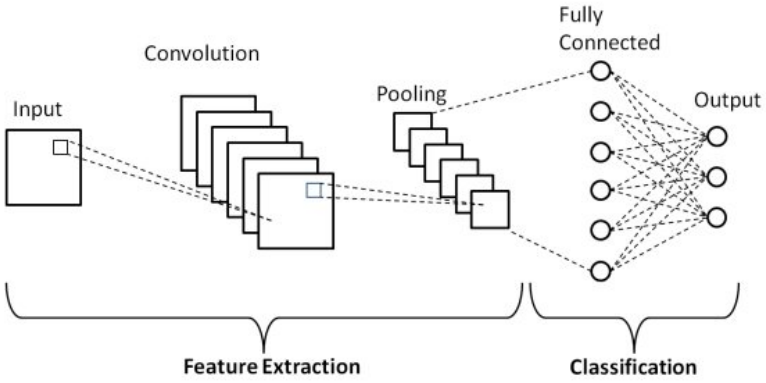

Figure 1.15: Visualization of CNN architecture. Image taken from [19].

- ***Recurrent neural networks*** was mentioned in 1.2.5 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 1.16.

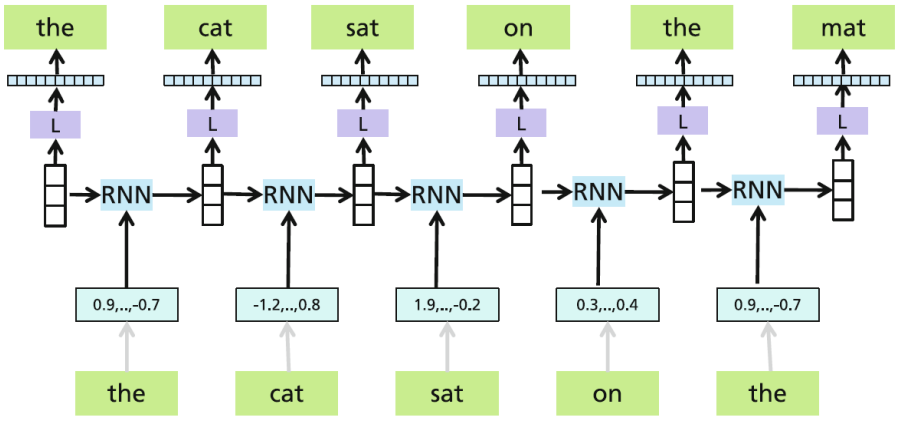

Figure 1.16: Process of prediction sequence of new words by Recurrent neural networks . Image taken from [16].

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. [16]

- ***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. [16]

Chapter 2

Large Pre-trained Language Models

In this chapter, the architecture of Large Pre-trained Language Models, the conversion of text into tokens and the learning process of neural networks of this type are described.

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. [16]

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 [16]. 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. [16]

- ***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. [16]
- ***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. [16]

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.

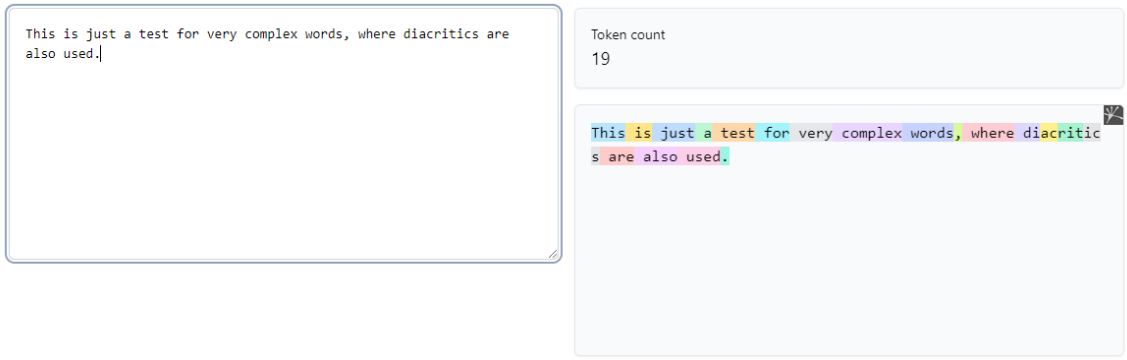

Figure 2.1: Tokenization of the sentence written in the English language.

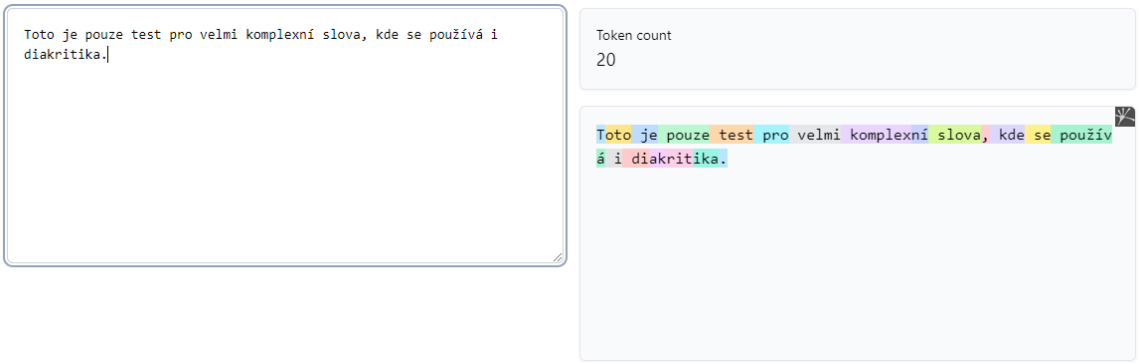

Figure 2.2: Tokenization of the sentence written in the Czech language

As can be seen in Figure 2.1, given a sentence, that has 14 words and 2 punctuation marks, it is encoded into 19 tokens. On the Figure 2.2, a 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. [16]

Common approaches of Simple Embeddings [16]:

- ***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.

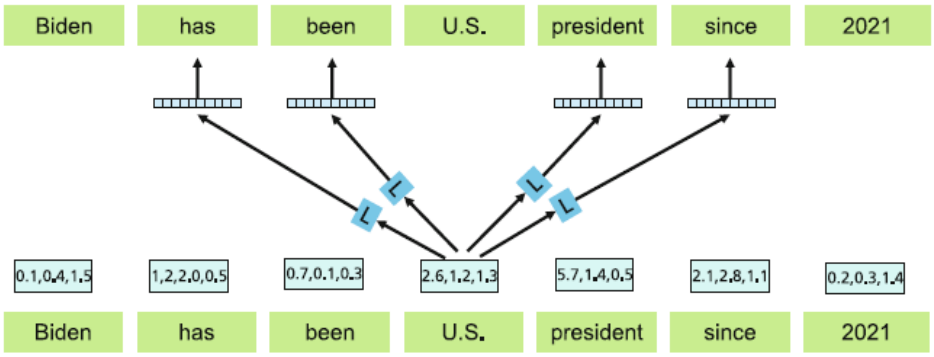

Figure 2.3: Word2vec predicts the words in the neighborhood of a central word by logistic classifier L. Image taken from [16].

- ***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 [16]:

- ***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. [16]

This is shown on Figure 2.4.

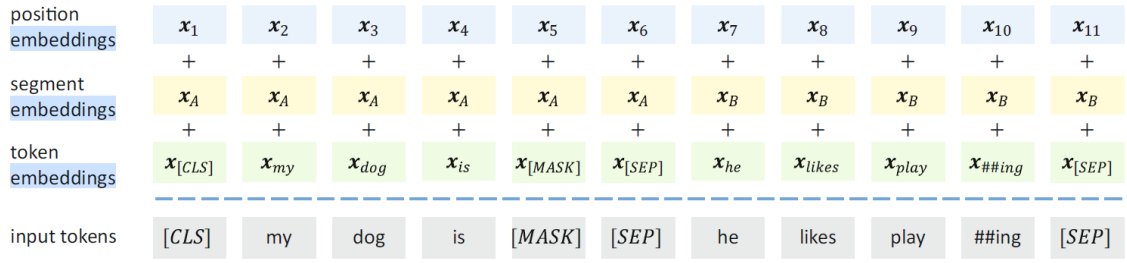

Figure 2.4: Token embeddings for Transformers, like BERT or GPT, contains position information duo to parallel computations of Transformers. Image taken from [16].

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 [16]:

- ***[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.

2.2 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 1.2.5.

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. [16]

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.

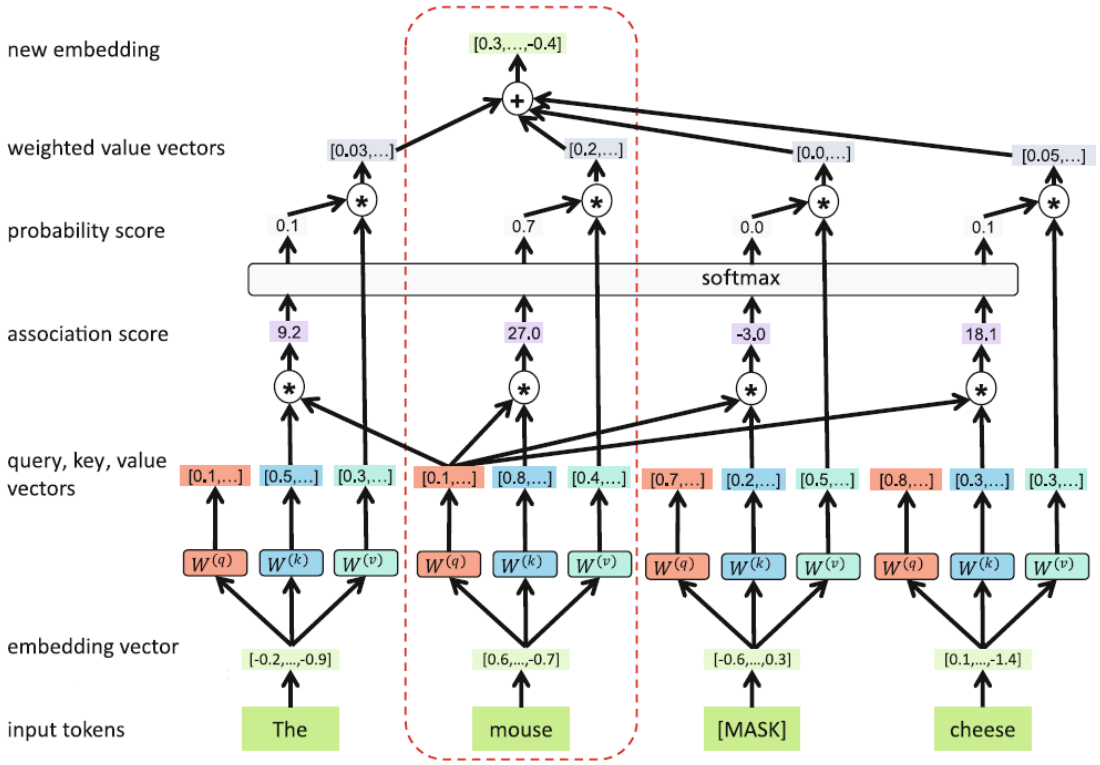

Figure 2.5: Computation of a contextual embedding for a single token “mouse” by attention mechanism. Image taken from [16].

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.

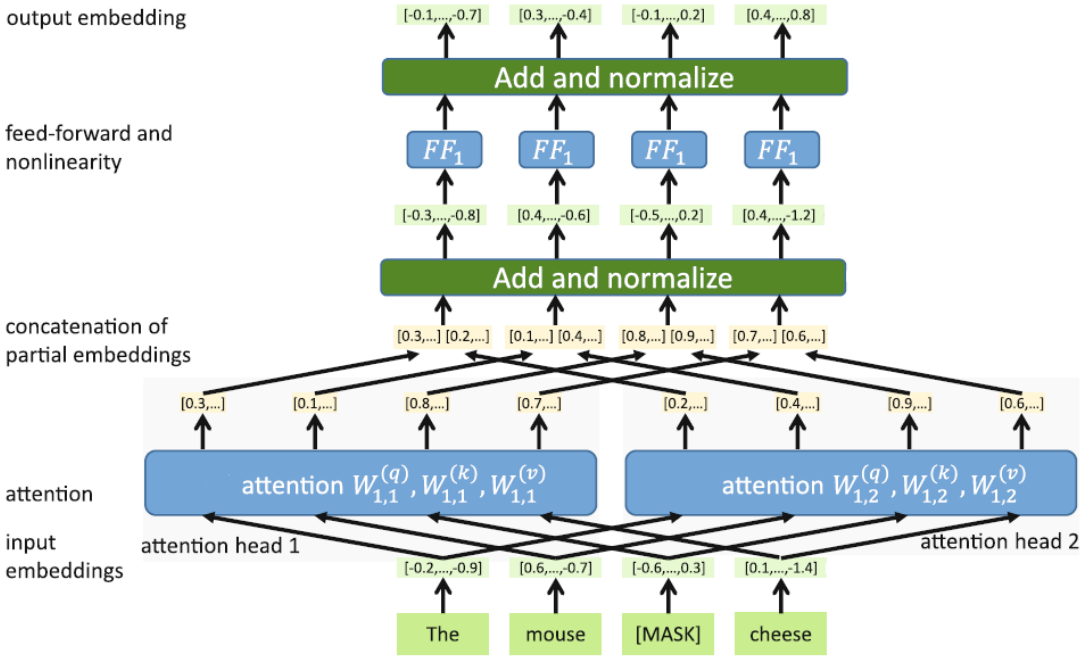

Figure 2.6: Visual representation of multi-head attention. Image taken from [16].

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.

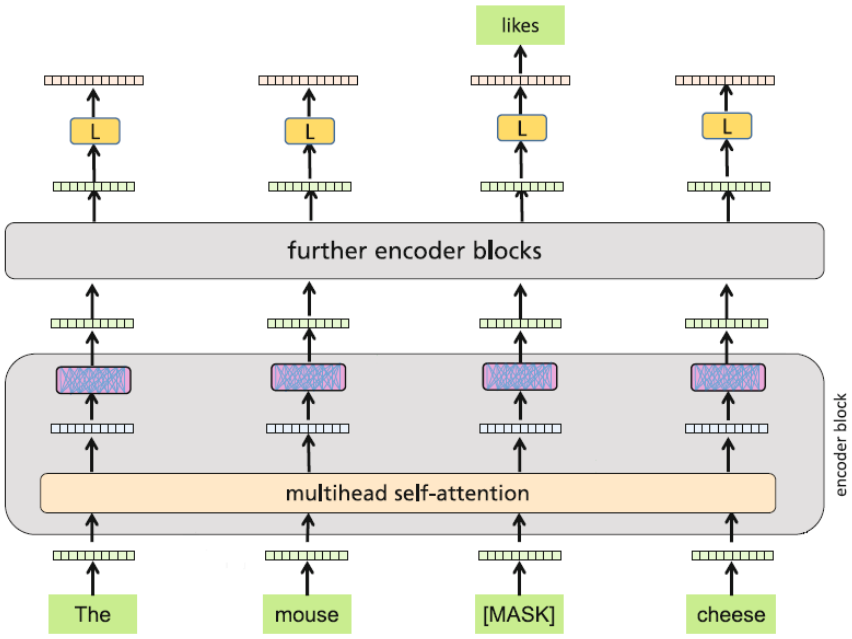

Figure 2.7: Scheme of stacking multiple transformer layers. Image taken from [16].

2.2.2 BERT

Bert is natural language processing model introduced by Google [9]. 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). [9]

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. [9]

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. [16]

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. [16]

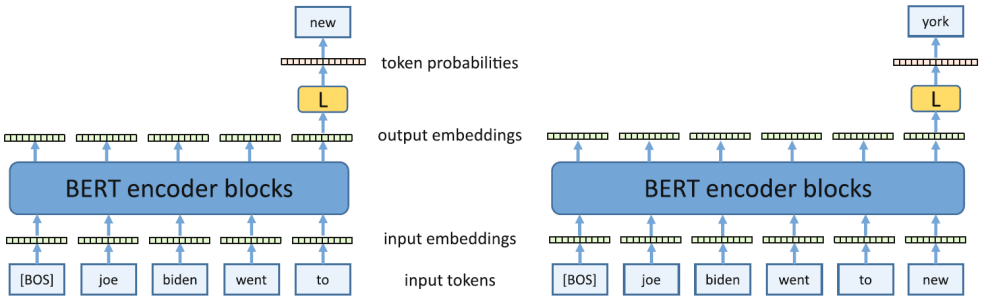

Figure 2.8: Transformer model predicts the next token in token‘s sequence. Image taken from [16].

This text generation process is same as text generation by recurrent neural networks or its variants. GPT uses attention mechanism described in 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.

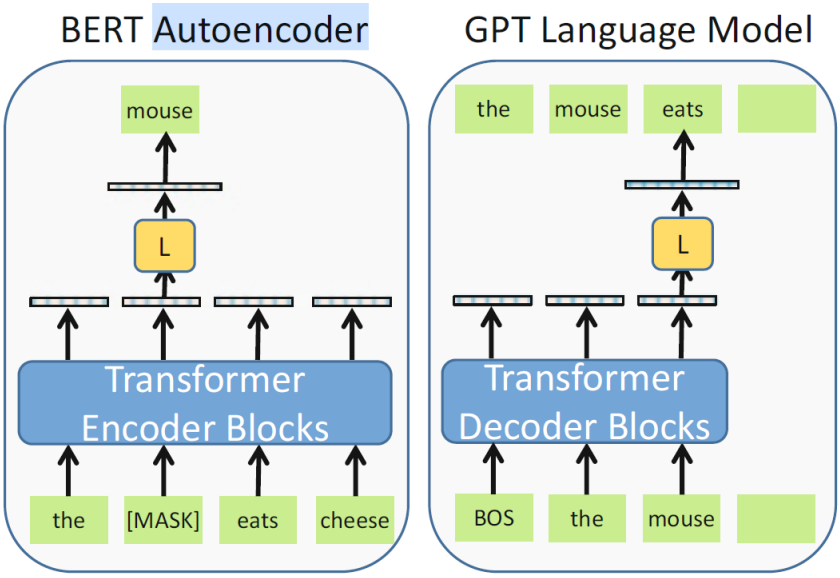

Figure 2.9: Comparation between word prediction by BERT and text generation by GPT. Image taken from [16].

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]

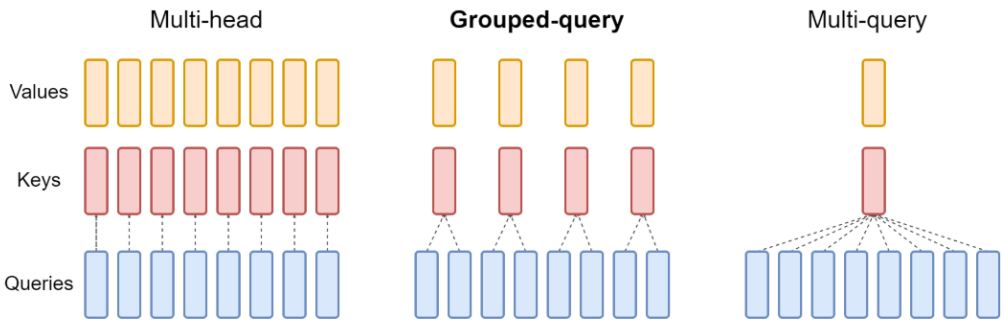

Figure 2.10: Difference between types of attention. Image taken from [3].

The main advantage of this model is that there are fewer parameters than other Large Pre-trained Language Models [27]. 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 [16].

- 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 [16]. 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. [18]

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 **Aplha(r_alpha)**. [14]

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.[14]

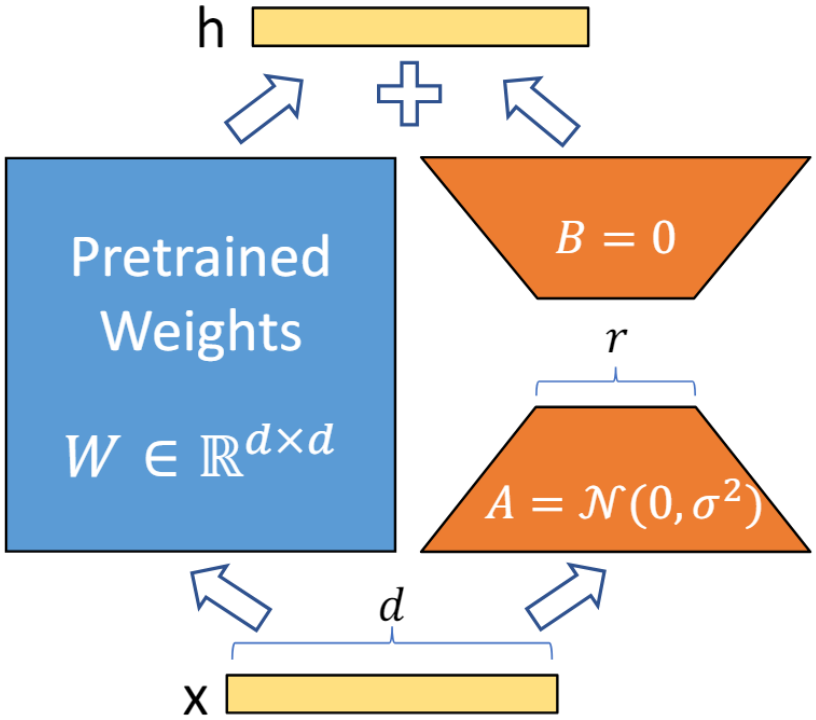

Figure 2.11: LoRA principle using initial pre-trained weights and two much smaller matrices. Image taken from [10].

2.4 Evaluation techniques

The most crucial part of training an artificial intelligence model is evaluating whether a given model produces better results than other models. 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 [16]:

- ***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.

  Different versions are used: 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.

- ***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.

Further, individual entities consist of the following attributes: For the patient entity, these are:

patient_id: Unique identifier of the patient.

gender_code: Gender (M for male, Z for female).

birth_date: The date of birth of the patient.

death_date: Date of the patient's death (added after death).

For the department entity, these are:

department_id: Unique department identifier.

department_code: Department code.

department_name: Department name.

parent_department_id: Identifier of the parent department.

department_type: Type of department (A as ambulance, L as bed, etc.).

active: Active department (1 means active).

For the hospitalization entity, these are:

hosp_id: Unique identifier of the hospitalization episode.

hospcase_id: Hospital case identifier (one or more episodes, hospital stay).

hosp_from_date: Date and time of the beginning of the hospitalization episode.

hosp_to_date: Date and time of the end of the hospitalization episode.

hospcase_from_date: Date and time of the start of the hospitalization case.

hospcase_to_date: Date and time of the end of the hospitalization case.

patient_id: Unique identifier of the patient.

department_id: Identifier of the current department, where the patient is hospitalized.

For hospitalization's diagnoses entity:

hosp_id: Unique identifier of the hospitalization episode.

dg_code: Code of diagnose.

dg_name: Name of diagnose.

dg_order: Order of diagnose.

For document entity:

document_id: Unique document identifier.

document_class: Document classification.

document_type: Document type.

document_name: Name of the document.

document_date: The date and time to which the document relates.

document_changed: The date and time of the last modification of the document.

hosp_id: Hospitalization identifier.

patient_id: Unique patient identifier.

document_text: Document text.

department_id: Executive department.

Relationships between entities are shown on 3.1.

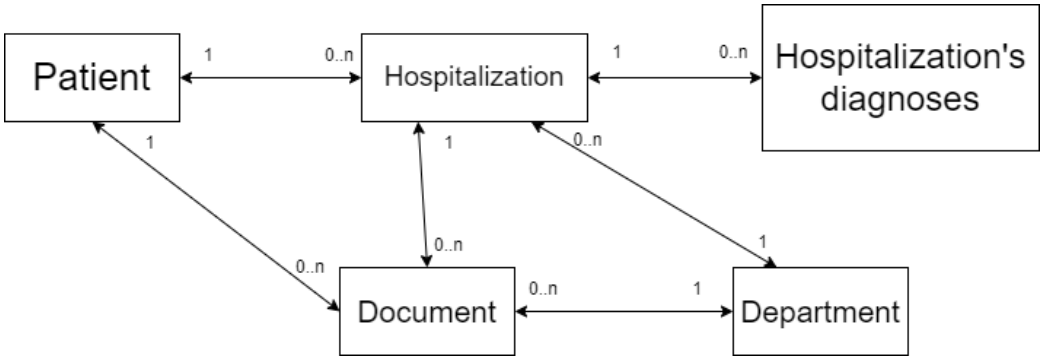

Figure 3.1: Entity relationship diagram of given dataset.

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 roast

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. Some of these categories are as follows:

ADL, ***Nursing Anamnesis***, ***Decubitus***, ***Decours*** , ***Education by Nurse***,

Epicrisis, ***Wound Documentation***, ***Hospitalization Report***,

Hygiene Regimen, ***Invasive Entries***, ***Consultation***, ***Laboratory Tests***,

Medication Request, ***Surgical Protocol***, ***General Nursing Documentation***,

Abortions, ***Prescription for Medical and Orthopedic Aid***,

Nursing Transfer Report, ***Admission Report***, ***Discharge Report***,

Autopsy Cover Sheet, ***Radiodiagnostics***, ***Prescription***, ***Social Record***,

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

Documents of dismissal 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 to:

- ***Diagnostic Summary***
- ***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 and example looks like this:

```
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 doktor:
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
```

The label will then be created from the dismissal report document, which has the following structure:

- ***Diagnostic Summary***
- ***Anamnesis***
- ***Current illness***
- ***Status praesens***
- ***Status localis***
- ***Treatment Summary***
- ***Surgical Procedure***
- ***Laboratory Results***
- ***Measured Data***
- ***Hospital Course***

- ***Pacient state at dismissal***
- ***Recommendations***
- ***Recommended Therapy***

The label will be exactly Hospital Course of dismissal report and example looks like this:

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

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 of course 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 complicates data extraction and leads to the creation of a low-quality dataset, which subsequently leads to the creation of a low-quality assistant.

3.2 Assistant API Gateway

In order to be able to use the assistant without the need to have it running locally, it needs to run on servers and use API to access it.

3.2.1 API

API is a set of rules and protocols that allows different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information.

API structure can be divided into:

- ***Headers***: Provide metadata about the request, such as authentication tokens.
- ***Parameters***: Include data sent to the API in the URL or body of the request.

To ensures, that only authorized users can access certain endpoints, this methods can be used: API keys, OAuth tokens, and JWT.

3.2.2 Ollama

It is a framework for setting up and running large language models such as Llama 2, providing engineers with the tools to build and execute these models locally. It addresses the need for a flexible and extensible environment to work with advanced machine learning models, offering a range of functionalities from API interactions to application lifecycle management. [13]

Chapter 4

Implementation and testing

According to the scheme 4.2, the implementation is divided into the following parts:

- ***Data preparation***: In this part, 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.
- ***Learning and set the correct parameters***: Choosing the right training procedure and additional training parameters for the best possible fine-tuning of the model.
- ***Set an API Gateway***: Setting an API Gateway and creating an application that can communicate with a fine-tuned model using an API.

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

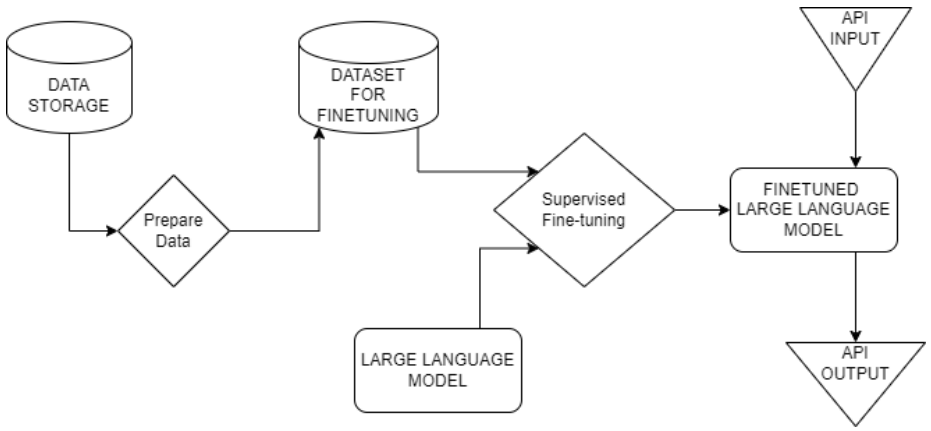

Figure 4.1: Simple schema of fine-tuning process of large language model.

4.1 Data preparation

The selected suitable model, where its selection is justified in 4.2 , has a maximum content length of 4096 tokens. Therefore, samples of these datasets will be evaluated in terms of

their content length. A new dataset,where all samples already meet the required content length, is created from the given dataset. These samples are covered by the Prompt, which is discussed in 4.2.2, and the Prompt is also included in the content length. This evaluation and creating new dataset is executed by the ***eval_dataset.py*** script written in Python.

3 versions of dataset were created, all of which consist of dismissal report, admission report and Decours parts as is described in 3.1.3. These versions were created by gradually improving previous version and removing noises.

4.1.1 1st version of dataset

1st version of dataset is just a combination of Current illness from admission report and Daily Progress Notes from Decours as input and Hospital Course from dismissal report as label. Without addictional changes like anonymizing doctor‘s name.

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

Table 4.1: Evaluation of 1st dataset.

4.1.2 2nd version of dataset

The 1st version of dataset has not only Hospital Course from dismissal report as label, but also other parts of the dismissal report. This is due to the fact that not all dismissal reports have the same structure. It may be due to the fact that the doctors can adjust it for themself.

In 2nd version of dataset is improved label to just contains Hospital Course and doctor‘s name are anonymized, but only those that could be identified based on their location in the structure. Doctor‘s name can still be found in freely written text.

Metric	**Count**
Total samples	5557
Content length below 4096.	5265
Content length higher than 4096.	292

Table 4.2: Evaluation of 2nd dataset.

4.1.3 3rd version of dataset

In the 3rd version, the occurrence of abbreviations in the text written by the doctor was analysed. Doctors do not only write abbreviations that specifically denote types of diseases or drugs, but also abbreviate words that are commonly used in communication. But sometimes these words are in full form and sometimes in abbreviated form. This can lead to the fact that the model will take them as words with different meanings, which can lead to worse results. These words have been adjusted to the full form.

Also label still contained other parts of the dismissal report, so label is improved improved again to just contains Hospital Course.

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

Table 4.3: Evaluation of 3rd dataset.

4.2 Suitable model

From the mentioned models in 2.2, the LLaMA model was chosen due this reasons:

- ***Computation and memory requirements***: Feasible requirements for computation and memory resources by using versions with low number of parameters. This versions are 13 bilions parameters and 7 bilions parameters.
- ***Multilingual***: The model can generate text in several languages. One of these languages is Czech language.
- ***Open-source***: It is open-source and can be fine-tuned locally.
- ***New***: It is a relatively new model, recently released

4.2.1 Model‘s variants

There were tested other multiple types of this model. Each of this model is pre-trained on different dataset and some have different number of trainable parameters.

These models are:

- ***togethercomputer/Llama-2-7B-32K-Instruct*** [26]
- ***Yukang/Llama-2-13b-longlora-16k-ft*** [29]
- ***Yukang/LongAlpaca-13B*** [30]
- ***meta-llama/Llama-2-13b-chat-hf*** [28]

After evaluating the performances of the models, **meta-llama/Llama-2-13b-chat-hf** is chosen as the most suitable model. Although the output was generated in English, the model generates an answer 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.

4.2.2 Prompt

The prompt is used to better define the input so that the model can understand it. It contains special tokens and defines the instruction of what the model should do.

Instruction is define as:

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

Prompt template for training of model:

```
<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>
```

Prompt template for inference of model:

```
 <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]
```

4.3 Learning

All 3 versions of the dataset were used to train the **meta-llama/Llama-2-13b-chat-hf** model.

To reduce the computational complexity and required memory, the quantization technique is used. It is about reducing the number of bits by which the parameters of the model are represented. Quantization from the BitsAndBytes library is used and reduction to 4bit.

For the additional necessary performance, the LoRA learning technique is used. For Models, which are learned on 1st version of dataset and 2nd version of dataset, are used

this LoRA parameters r=16, lora_alpha=64. This leads to not quiet good results and evaluation was done purely by manually checking the result.

Next, the model was trained with the 3rd version of the dataset, which has 5000 samples and training was set on 3 epochs. LoRA parameters was changed to r=16, lora_alpha=16. It led to better results, where the model did not give false information so often.

Last version of a model was evaluated using the ROUGH metric. The result over 250 samples that were not part of the training dataset of this model:

Metric	**Score**
rouge1	0.3520777
rouge2	0.2225495
rougeL	0.2886016
rougeLsum	0.3039186

Table 4.4: ROUGE Scores

4.4 Inference and API communication

4.4.1 Inference

The model starts with pre-set parameters. One of these parameters is the temperature, which determines to what extent the model generates text using chance. This leads to the fact that the model makes up a lot, but too low again leads to the fact that the model generates text with information that was learned with a similar input but should not have been generated with the current input. Temperature is set on 0,4.

Example of generated text by trained model:

```
Pacient byl přijat k provedení koronarografie pro atypické bolesti na hrudi.
Koronarograficky normální nález na koronárních tepnách, EF LK 60%.
Výkon proběhl nekomplikovaně, zápěstí po intervenci klidné.
Telemetricky po celou dobu hospitalizace sinus. rytmus bez záchytu arytmií.
Pacienta propouštíme v KP kompenzovaném stavu. Odvoz vlastní.
```

And text written by doctor:

```
Pacient byl přijat k provedení selektivní koronagrafie pro atypické bolesti na hrudi.
Výkon nekomplikovaně, s nálezem - normální nález na koron.
tepnách, EF 60%. Vyloučena kardiální etiologie obtíží.
Telemetricky trvá eufrekvenční fibrilace síní.
Závěrem dimise do ambulantní péče ve stabilizovaném stavu.
```

Text generated by GPT3.5 using same input:

```
Pacient byl odeslán k provedení koronarografie (SKG)
z ambulance kvůli bolestem na hrudi. Echo z 22.
dubna ukázalo ejekční frakci levé komory (EF LK) 55 %,
bez poruchy kinetiky, s mírnou mitrální regurgitací.
Levá síň (LS) má 50 mm, hraniční velikost pravé komory (PK)
a mírná trikuspidální regurgitace.
```

```
Pacient udává, že se asi rok zadýchává při práci
a při chůzi do druhého patra, ale nezastavuje.
Bolesti na hrudi mívá nejčastěji večer v klidu,
které popisuje jako sekundové tlaky, někdy píchání
a řezání na hrudi, bez vazby na polohu či námahu.
Občas podobné potíže zažívá i při práci.
Neuvádí bušení srdce, otoky ani pre/synkopy.

Denní průběh:
Dne 30.4.2022 v 8:34 zapsal doktor:
Subjektivně se pacient cítí dobře, má pouze nachlazení, rýmu a kašel.
Objektivně je afebrilní, kardiopulmonálně kompenzovaný,
orientovaný a spolupracující, dýchá normálně, bez ikteru a cyanózy.
Srdeční akce pravidelná, bez šelestu, dýchání sklípkové, bez výpotků.
Břicho klidné, nebolestivé, dolní končetiny bez otoků a známek trombózy.
Telemetrie: sinusový rytmus, bez arytmie.
Plán: dimise.
```

For comparison, the text was also generated by the GPT 3.5 model. The fine-tuned LLaMA model tries to imitate the performance of a doctor. It is not detailed, but 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. 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.

4.4.2 API communication

For backend implementation is used Python framework Flask. And for secure reason is used API key for communication between Client and server. Server after request provided by Client starts inference with input data from Client.

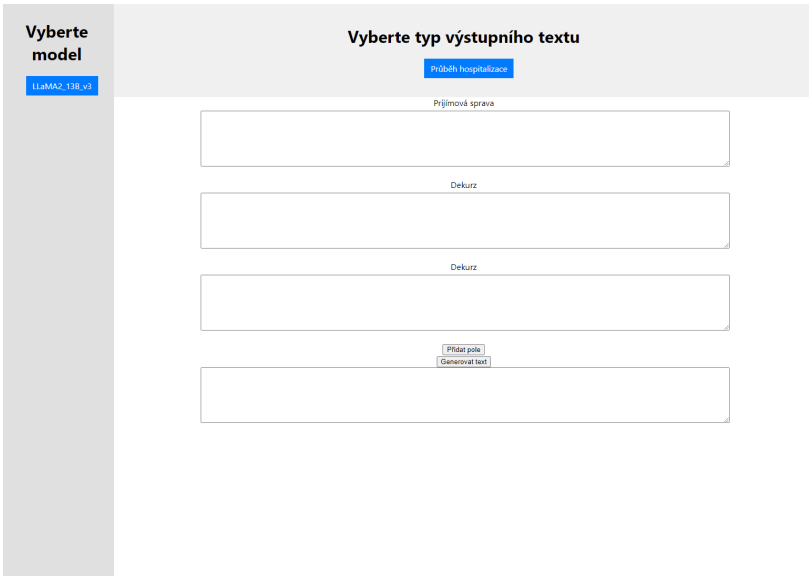

Figure 4.2: Client‘s application.

Conclusion

The model is quite good at generating text in Czech. When examining the generated output, it was found that it was correcting grammatical errors. The main problem consists in not supplying the necessary data for the input as would be needed for the given output. That's why I think it leads to adding text to the output that shouldn't be there. The necessary data is, however, difficult to extraxt automatically, since doctors do not always follow the structure of the document. Another slightly smaller problem lies in the abbreviations, which need to be replace with whole words. And last but not least is the architecture of the model. It would be necessary for the model to „understand" more the context in the given text. A model with 70B parameters or other architectures would probably handle it better. Duo the data anonymization, the datasets and trained models are not publicly available.

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.

[4] Banoula, M. What is Perceptron: A Beginners Guide for 2023. *Simplilearn*. May 2021. Available at: https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron.

[5] Chandra, R. *Problem Decomposition and Adaptation in Cooperative Neuro-evolution*. Dissertation.

[6] Cotoia, A. Axon Terminal - The Definitive Guide | Biology Dictionary. *Biology Dictionary*. May 16 2020. Available at: https://biologydictionary.net/axon-terminal/.

[7] Flasinski, M. *Introduction to Artificial Intelligence*. 1st ed. Cham: Springer Nature, 2016. ISBN 3319400223.

[8] 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.

[9] Horev, R. BERT Explained: State of the art language model for NLP. *Towards Data Science*. 2018. Available at: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[10] 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.

[11] 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.

[12] Kalirane, M. *Gradient Descent vs. Backpropagation: What's the Difference?* January 2023. Available at: https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/.

[13] Mutable.ai. *Auto Wiki by Mutable.ai — High-Quality Generated Code Documentation* [https://mutable.ai]. May 2024. Available at: https://mutable.ai.

[14] 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. Available at: https://www.anyscale.com/blog/fine-tuning-llms-lora-or-full-parameter-an-in-depth-analysis-with-llama-2.

[15] 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.

[16] 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.

[17] Page, J., Liechty, Z., Huynh, M. and Udall, J. BamBam: Genome sequence analysis tools for biologists. *BMC research notes.* november 2014, vol. 7, p. 829. DOI: 10.1186/1756-0500-7-829.

[18] 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.

[19] 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.

[20] Ris Ala, R. *Fundamentals of Reinforcement Learning.* Cham: [b.n.], 2023. ISBN 3031373448.

[21] 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.

[22] 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.

[23] 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).

[24] Steve, P. *Ludské telo: Ilustrovaný sprievodca štruktúrou, funkciami a poruchami [The Human Body].* Ikar, a. s., 2008. ISBN 978-80-551-1731-7.

[25] 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.

[26] 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.

[27] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A. et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023.

[28] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A. et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. 2023.

[29] 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.

[30] Yukang. *Yukang/LongAlpaca-13B · Hugging Face*. 2024. Accessed: 2024-05-17. Available at: https://huggingface.co/Yukang/LongAlpaca-13B.