CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

Text Summarization of Medical Records

Master's Thesis

Jakub Monhart

Prague, May 2024

Study programme: Open Informatics
Branch of study: Artificial Intelligence

Supervisor: Ing. Jan Drchal, Ph.D.

Acknowledgments

First and foremost, I would like to express my deep gratitude to my supervisor, Jan Drchal, for his guidance and valuable support throughout the course of working on this thesis. I would also like to thank the Institute for Clinical and Experimental Medicine (IKEM) for providing me with the data essential for this work.

My heartfelt thanks go to my close friends, whom I met during my studies. Your support and encouragement have been a constant source of motivation and joy.

Finally, I am deeply thankful to my soon-to-be wife Tereza and my family for their support and understanding. Your patience and love have been of great importance to me.

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Monhart Jakub**	Personal ID number:	**483438**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence**		

II. Master’s thesis details

Master’s thesis title in English:

Text summarization of medical records

Master’s thesis title in Czech:

Textová shrnutí medicínských záznamů

Guidelines:

The aim of the thesis is to propose and implement a method for automatic generation of hospital discharge report summary on Czech language dataset provided by Institute for Clinical and Experimental Medicine (IKEM).
Detailed instructions:
1) Research state-of-the-art methods for NLP summarization, with focus on domain specific data.
2) Analyze the Czech dataset from IKEM. Preprocess and clean the data to make it suitable for solving the task.
3) Consider another (e.g., English) text clinical datasets, or summarization datasets from a different domain, in case these are likely to improve the Czech discharhge report summarization models.
4) Train and evaluate multiple models to solve NLP summarization task on clinical data, focusing on abstractive and extractive methods utilizing transformer architectures.

Bibliography / sources:

[1] El-Kassas, Wafaa S., et al. "Automatic text summarization: A comprehensive survey." Expert systems with applications 165 (2021): 113679.
[2] Liu, Yixin, et al. "BRIO: Bringing Order to Abstractive Summarization." Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022.
[3] Tian, Shubo, et al. "Opportunities and challenges for ChatGPT and large language models in biomedicine and health." Briefings in Bioinformatics 25.1 (2024): bbad493.
[4] Straka, Milan, et al. "SumeCzech: Large Czech news-based summarization dataset." Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). 2018.

Name and workplace of master’s thesis supervisor:

Ing. Jan Drchal, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **01.02.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Jan Drchal, Ph.D.	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Declaration

I declare that presented work was developed independently, and that I have listed all sources of information used within, in accordance with the methodical instructions for observing ethical principles in preparation of university theses.

Date ....

Abstract

A vital part of healthcare is creating, processing and storing good quality medical documentation. The task is, however, time-consuming and burdens the medical care workers heavily. This thesis's goal is to explore the possible automation of part of this process. Concretely, it focuses on automatically generating the Hospitalization summarization paragraph of the Discharge report, an important medical document ensuring continuity of care. To this end, two Czech text medical datasets from two departments of the Institute for Clinical and Experimental Medicine (IKEM) are used. The datasets are analyzed and preprocessed for the task. Language models based on the Transformer architecture, pretrained on multilingual datasets are utilized. The models are further fine-tuned on the datasets, that are preprocessed for a text summarization task. Both extractive and abstractive text summarization approaches are explored. All the models are evaluated using automatic metrics. The automatic metrics show that abstractive summarization methods outperform the extractive ones on the task. Further, manual evaluation of the best performing abstractive summarization models is also conducted, showing that the models solve the task correctly on over 40% of the test samples. The manual evaluation also shows that the automatic metrics measuring the quality of the generated summary (using the summary written by the doctor as a reference) are consistent with the manually assigned quality labels, which justifies their use.

Keywords Machine Learning, Natural Language Processing, Summarization, Abstractive Summarization, Extractive Summarization, Medical Documentation

Abstrakt

Zásadní součástí zdravotní péče je vytváření, zpracování a uchovávání kvalitní zdravotnické dokumentace. Tento proces je časově náročný a značně zatěžuje zdravotnický personál. Cílem této práce je prozkoumat možnost automatizace části tohoto procesu. Konkrétně se tato práce zaměřuje na automatické generování odstavce "Průběh hospitalizace" v propouštěcí zprávě, což je důležitý lékařský dokument pro zajištění kontinuity péče. V práci jsou použity dva české textové lékařské datasety ze dvou oddělení Institutu Klinické a Experimentální Medicíny (IKEM). Datasety jsou nejdříve zanalyzovány a zpracovány do vhodného formátu. Jako základ řešení úlohy jsou využity jazykové modely založené na architektuře Transformer, předtrénované na vícejazyčných datasetech. Tyto modely jsou dále učeny na našem datasetu zpracovaném do formátu vhodného pro sumarizaci textu. Jsou prozkoumány jak extraktivní, tak abstraktivní přístupy k sumarizaci textu. Všechny modely jsou hodnoceny pomocí automatických metrik. Ty ukazují, že abstraktivní metody fungují lépe v porovnání s těmi extraktivními. Dále je provedeno manuální hodnocení nejlepších abstraktivních modelů, které ukazuje, že modely generují průběh hospitalizace správně na více než 40% testovacích vzorcích. Manuální hodnocení dále ukazuje, že automatické metriky měřící kvalitu generovaného průběhu hospitalizace (v porovnání s průběhem hospitalizace napsaného lékařem) jsou konzistentní s manuálním ohodnocením, což ospravedlňuje jejich použití.

Klíčová slova Strojové učení, Zpracování přirozeného jazyka, Sumarizace, Abstraktivní sumarizace, Extraktivní sumarizace, Zdravotnická dokumnetace

Contents

Chapter 1

Introduction

Recent advancements in Artificial Intelligence and Natural Language Processing bring a promise of revolutionizing various domains, including healthcare. A promising application of these technologies is the automation of medical documentation processing. It has been widely discussed, especially in the context of omnipresent digitalization, that medical care workers are burdened with time-consuming clerical work. Simply reducing the amount of paperwork is not possible, as processing the medical documentation and keeping good quality records is vital to ensure continuous high-quality healthcare.

One type of critical document in healthcare is the Discharge report. It gives a concise yet thorough description of the patient's hospital stay, detailing the reason for hospitalization and the patient's relevant current problems. Crafting the report is time-consuming and requires retrieving information from several medical documentation notes.

In this thesis, we explore the possibility of partially automating this task by using a Language Model to generate the Hospitalization summary paragraph of the discharge report. The hospitalization summary paragraph gives a concise story of the hospitalization and is important to ensure good continuity of patient care. While creating and processing medical text documentation is generally a challenging task requiring the domain knowledge of a medical specialist, the hospitalization summary usually contains information that can already be found in an admission report, documentation concerning examinations and procedures, or other medical notes. This makes it reasonable to formulate the automatic generation of the hospitalization summary paragraph as Text Summarization (a natural language processing task), which can be solved with a method based on a suitable state-of-the-art language model.

To experiment with this, we obtained two Czech datasets from the Institute for Clinical and Experimental Medicine (IKEM) in Prague. Each dataset is from a different department and contains samples with information about hospitalizations, including the Discharge report with the hospitalization summary. We define the task as automatically generating the hospitalization summary using the other information about the hospitalization as the source text.

To our best knowledge, there is no published work focused on using language models to generate discharge reports in the Czech language and the research on processing Czech medical documentation in general is very limited.

1.1 Thesis structure

The thesis is structured as follows. Chapter 1 introduces the goal of this thesis and the motivation behind it. In Chapter 2, we describe the summarization task as a natural language processing problem. The language models we use, together with approaches for their evaluation are introduced. We shortly present related problems and publications in Chapter 3, all concerning natural language processing tasks in the context of the medical domain. In Chapter 4, we introduce the real-world medical text datasets provided by IKEM, we show how we preprocess the data and subsequently analyze it. We describe our approach and methods for solving the task in detail in Chapter 5. In Chapter 6, we experimentally compare our methods through both automatic and manual evaluation. We summarize our results, discuss possible improvements and future work, and conclude this thesis in Chapter 7.

Chapter 2

Preliminaries

This chapter serves as the groundwork for this thesis. All the topics we discuss are concerned with natural language processing. *Natural Language Processing* (NLP) is a subfield of computer science concerned with processing and manipulating text using computer programs. In this thesis, we only consider machine learning approaches to NLP and generally call them *Language Models*. In the following text, we first define the summarization task in Sec. 2.1. Then, we explore the used language models in Sec. 2.2, covering the original Transformer architecture, upon which all modern language models are based, continuing with a description of tokenization and description of all the various models we use in the experiments in this thesis. A discussion about various techniques of inference using the language models follows in Sec. 2.3. In Sec. 2.4, we describe techniques to train large models efficiently. The chapter is closed with Sec. 2.5, describing the evaluation of language models, mainly discussing the metrics we use.

2.1 Summarization

Automatic text summarization is an NLP task that involves condensing input text into a shorter, more concise version with the goal of retaining the most important information and meaning of the original text. In this thesis, we generally denote the input text as *source* and the output text as *summary*. There are two main approaches to summarization:

- **Extractive summarization** aims to identify and extract important sentences or subsequences from the source. The resulting summary is a concatenation of such identified sentences or text spans.
- **Abstractive summarization** creates new text based on the content of the source. It is not constrained to only use sentences or text spans of the original text. Abstractive summaries can contain new words or utilize some background information from pretraining or finetuning (we explain these processes in Sec. 2.2). The goal ultimately is to output a summary a human would write. By choosing specific data for fine-tuning, the model can be trained to output summaries that conform to some specifications. The summaries in the training data might be less or more concise, technical or scholarly, etc.

A common problem with abstractive summarization (and open text generation using language models in general) is *hallucination*. Hallucination happens when the model generates content that is not inferable from the source. While hallucination in some contexts might not be problematic, it can cause the model to produce summaries that are factually incorrect, which we want to avoid. We use terminology from [6] to speak about the generated text in the context of hallucination. We say a generated summary is *faithful* if it is consistent and truthful to the source text. We denote a generated summary *factual* if it is based on fact.

2.2 Language Models

NLP has advanced in recent years mainly due to increased computing capacity and data availability. An abundance of training data is supplied by using self-supervised and unsupervised training paradigms, which allows to use large amounts of data scraped from the internet. This is contrary to a supervised paradigm, where (usually) human-annotated data are required, which are costly and time-consuming to acquire. Further, the Transformer architecture [37] allows to process words (more specifically, tokens) of its input in parallel during training, better utilizing the increased compute power. In contrast, previous mainstream architecture for NLP - Recurrent Neural Networks - prevented parallelization due to computing non-linear dependencies between input sequence elements. These advancements in NLP architectures, together with ever-increasing computing effectiveness (most notably of Graphic Processing Units allowing parallel computations), make it possible to process unprecedented amounts of text data using models with up to billions of trainable parameters during training, creating language models, that can be fine-tuned to various downstream tasks with great results. Language models with large amounts of trainable parameters that are trained to have general-purpose language manipulation capabilities are being called *Large Language Models* (LLMs) today.

Training LLMs in self-supervised and unsupervised fashion on large amounts of general domain text data is called *pretraining.* The training is carried out on multiple (tens or hundreds) GPUs, spanning weeks or even months. A pretrained language model or LLM is usually not very useful on its own. To use it on a downstream NLP task (such as question answering, text classification, or summarization), the model needs to be *fine-tunned* To fine-tune pretrained language model on a specific downstream task, only a small amount of data and computation time is needed compared to resources used for pretraining. This setting is beneficial, as the fine-tuned model utilizes the language representations learned by the model during the long and resource-heavy pretraining, allowing the use of a single pretrained model as a base model for multiple downstream tasks, not training a language model from scratch each time. Experiments have repeatedly shown that increasing the number of trainable parameters and the amount of data used during pretraining propagates to better results of the fine-tuned models, motivating the recent scaling of LLMs.

One common approach to solve an NLP task is to take a state-of-the-art (SOTA) pretrained model, create a high-quality supervised dataset for the required downstream task and fine-tune a specialized version of the language model on it. Another more recent approach called *Instruction Fine-Tuning* (IFT), fine-tunes the pretrained model on a dataset of instructional *prompts* and corresponding outputs. Such datasets are usually created manually. This type of pretraining allows the model to follow general instructions, rather than only perform specific tasks.

We now follow with a description of the base Transformer architecture followed by an introduction of tokenization. Next, we describe individual pretrained language models we use as a base of our methods. While there is an abundance of language models existing today with new ones appearing every month, most of them are pretrained (and optionally fine-tuned) on solely English data. A significant amount of language models are being trained on multilingual corpora, but those, usually, only contain a handful of most-spoken languages. To ensure good performance on our task, we limit our choice of models to the ones that are trained on multilingual datasets that specifically contain Czech pretraining data. This significantly narrows our options. Further, we need the models' weight to be publicly available to be able to use it as a starting point (checkpoint) which we fine-tune on our task-specific dataset. We first describe XLM-RoBERTa (Sec. 2.2.3) which we use for our extractive summarization method. In Sec. 2.2.4 - Sec. 2.2.6, we describe the language models (mBart, mT5 and AYA) we fine-tune for the abstractive summarization approach. Lastly, we describe Sentence Transformer in Sec. 2.2.7. We use it as one of the similarity metrics for the pseudo-labeling method with which we create training data for the extractive summarization approach.

2.2.1 Transformer

All SOTA language models today use the Transformer architecture [37], built on top of a *self-attention* mechanism. Self-attention mechanism allows to model dependencies between elements of its input sequence irrespective of their distance and enables parallel computation.

The bare *attention* is a function mapping n queries and m key-value pairs to n outputs. Queries, keys, values and outputs are represented as rows of matrices $Q \in \mathbb{R}^{n \times d_q}$, $K \in \mathbb{R}^{m \times d_q}$, $V \in \mathbb{R}^{m \times d_v}$. The attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\mathrm{QK}^{\mathrm{T}}}{\sqrt{\mathrm{d_q}}}\right) V. \tag{2.1}$$

The Transformer uses a multi-head version of the attention function, which first creates h different linear projections of Q, K, V and applies the attention function to each of them:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{2.2}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{2.3}$$

where W_i^Q, W_i^K, W_i^V, W^O are learnable parameters. This allows for different attention heads to attend to different positions of the input sequence and was shown to achieve better performance compared to simple attention.

Self-attention is a modification, where queries, keys and values are all the elements of the input sequence. The function therefore becomes:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{2.4}$$

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V), \tag{2.5}$$

where X is a matrix of the input sequence elements.

The Transformer is composed of an *encoder* and a *decoder*, each composed of several corresponding blocks, as illustrated in Fig. 2.1. Each encoder block consists of multi-head self-attention followed by a fully connected feed-forward network, which consists of two layers with a ReLU activation function in between. A residual connection is applied to both the self-attention and the feed-forward network. After each residual connection, a layer normalization is applied. The first encoder block takes the inputs encoded in the embeddings (explained further in Sec. 2.2.2). Since self-attention is independent of the order of elements in the input, a positional signal is also added to the embedded input elements. While the original Transformer used a sinusoidal positional embedding, it is now more common to use learned position embeddings.

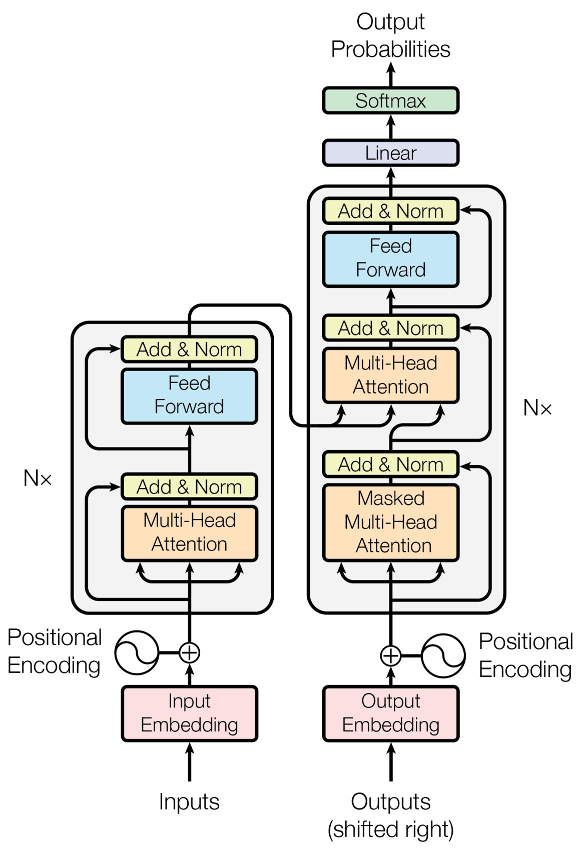

Figure 2.1: Illustration of the Transformer model architecture. Consisting of N encoders (left block) and N decoders (right block). Taken from [37]

The decoder block differs from the encoder block by also including a standard attention layer that performs multi-head attention on the output of the last encoder block (key and values), and the output of the decoder block self-attention layer (queries). The self-attention in the decoder block is masked, ensuring (together with the shifted outputs) that it can not attend to subsequent positions, making predictions for position i depend only on known outputs at positions less than i.

Language models based on the Transformers architecture utilize it either as it is, or use only the decoder or the encoder part. We describe these three options in more detail:

- **Encoder only** models use only the encoder part of the Transformer architecture. For each input token, they output a fixed-sized embedding.
- **Decoder only** models use only the decoder part of the Transformer, which is uni-directional, and the model is therefore auto-regressive. Their output can be described as:

$$P(u_i|u_0, \ldots, u_{i-1}; \Theta), \tag{2.6}$$

  where u_i is the token to be predicted at time-stamp i and $u_0, \ldots, u_{i-1}$ is the predicted output sequence so far.
- **Encoder-Decoder** models use the full Transformer architecture. As described above, the input sequence representation from the last encoder block is used as one of the inputs of the decoder blocks. The decoder then autoregressively generates an output

token sequence. This can be considered as conditional text generation. This type of architecture is also called sequence-to-sequence.

When training the decoder or encoder-decoder model to output a specific sequence, a *teacher forcing* mechanism is used. If the decoder would be used in the same way as during inference, where the previously generated output tokens are used as the decoder input, the loss function would provide too noisy learning feedback. Instead, at each time step, the true output tokens are supplied on the input of the decoder, teacher forcing the model to learn the correct output sequence.

2.2.2 Tokenization

A crucial part of language modeling is tokenization, a process of splitting a text into tokens (words or subwords). The tokens can be converted to IDs using a look-up table. In the Transformer architecture (and all language models we use in this work), each token ID corresponds to a learnable embedding (row of the input embedding matrix). A collection of all unique tokens a tokenizer (and therefore a language model) distinguishes is called vocabulary and its size is limited. This is mainly due to constraints on the size of the embedding matrix due to memory. Using limited vocabulary size means not every unique word can be represented with a different token. Sub-word tokenization ensures any text sequence can be encoded as tokens (and therefore token IDs with the corresponding embeddings). Deciding how to assign the sub-words to tokens is not a trivial task. Tokenizers are trained on the large pretraining data corpora with a focus on ensuring the most occurring words/sub-words are represented with a single token and only the less occurring words/sub-words are split into multiple tokens. The three most commonly used algorithms for training tokenizers are Byte-Pair Encoding, WordPiece and SentencePiece.

Byte-Pair Encoding (BPE) [39] works by creating a base vocabulary consisting of all characters that occur in the dataset. It then iteratively learns merge rules to create a new token from two tokens, until the vocabulary is of the desired size. BPE counts the frequency of each possible merged token pair in the dataset and picks the one that occurs most frequently. A modification of BPE, Byte-level BPE, works in the same way, but it uses bytes as the base vocabulary. This is beneficial, as the base vocabulary is only of size 256 (compare to the number of all possible Unicode characters) while ensuring every possible character can be represented using this vocabulary.

WordPiece [41] works very similarly to BPE. It also initializes the vocabulary with every character present in the dataset but differs in how it learns the merge rules. Instead of choosing the most frequent token pair to merge, it chooses the pair that maximizes the likelihood of the dataset. This is equivalent to finding a token pair, with the maximal score:

$$\text{score} = \frac{\text{freq\_of\_pair}}{\text{freq\_of\_first\_token} \times \text{freq\_of\_second\_token}}. \tag{2.7}$$

SentencePiece [33] was created to alleviate problems with processing multilingual datasets. Concretely, the previous tokenizers assume the input text uses spaces to separate words, which is not true in all languages (e.g. Chinese, Japanese, Thai). To solve this, SentencePiece treats the input as a raw input text, including the space in the set of characters to include in the base vocabulary. It then uses BPE or Unigram algorithm [32] to build the vocabulary.

Tokenizer is not used only for encoding the input text, but also for decoding the model output. Language models output probability distribution over the vocabulary for each output token. From this probability, each token is sampled (see Sec. 2.3), resulting in generated output (e.g. summary) consisting of token IDs. The tokenizer is used to decode those IDs into the generated output text.

In the following text, whenever we talk about the input or output of a language model, we automatically consider the use of the model's tokenizer for the encoding and decoding, if not specified otherwise.

2.2.3 XLM-RoBERTa

To describe XLM-RoBERTa introduced in [21], we start by introducing *Bidirectional Encoder Representations from Transformers (BERT)* [26]. When published, BERT achieved new SOTA results on several NLP tasks. Its architecture consists of a bidirectional multi-layer Transformer encoder. Authors of BERT argue that the bidirectional nature of the encoder allows the model to learn deep representations that lead to better performance on downstream tasks.

Bert uses the WordPiece tokenizer. The first token of every sequence starts with a special classification token `[CLS]`, whose final hidden state is used for the sequence representation in classification tasks. During pretraining, sentence pairs are differentiated by separating with special separation token `[SEP]` and by adding a learned embedding to every token of a sentence indicating whether it belongs to sentence `A` or `B`. Since the architecture is bidirectional, the standard language modeling objective cannot be used, as the model can see the next token it ought to predict. To overcome this, authors of [26] pretrained the model using two unsupervised tasks described below.

- **Task 1: Masked LM**
  A certain percentage (15% in the case of BERT) of the input tokens is masked at random. If a token is chosen to be masked, it is replaced by a special token `[MASK]` 80% of a time, a random token 10% of a time, and kept unchanged 10% of a time. Output representations of the masked tokens are then trained to predict the original tokens using the cross-entropy loss.
- **Task 2: Next Sentence Prediction**
  Training data for this task can be also generated from unlabeled data. Two sentences are packed in a sequence and differentiated with `[SEP]` token and sentence embedding. 50% of the time, the second sentence is the one that actually follows the first sentence in the training corpus. In the other samples, the second sentence is chosen at random. The last hidden representation of `[CLS]` token (output of final encoder block) is used to train the model to predict this binary task with a linear classifier.

The BERT model was first pretrained on these two tasks and subsequently fine-tuned on supervised data of a respective downstream task.

Authors of [29] noticed that the BERT model was severely undertrained. They dropped the Next Sentence Prediction task, increased the batch size and amount of data used for pretraining, modified hyperparameters and trained the model for more optimization steps. The trained model called *Robustly Optimized BERT Approach (RoBERTa)* and using the same architecture as BERT, achieved SOTA results again.

Both models introduced in this section were trained on purely English text corpus. To generalize the RoBERTa model for multilingual data, authors of [21] built a multilingual dataset consisting of CommonCrawl Corpus[1] in 100 languages. The resulting XLM-RoBERTa

[1] https://commoncrawl.org/

model achieved SOTA results on multilingual language tasks.

The XLM-RoBERTa model uses learned, but fixed-sized positional embedding. This limits the length of the input sequence it can process to 512 tokens.

2.2.4 mBART

We again first describe the *BART* model [27], as it precedes its multilingual version *mBART*. BART's architecture is identical to the full Transformer (the Encoder-Decoder) described in Sec. 2.2.1. It only differs by substituting the ReLU activation functions for GeLUs. BART is pretrained unsupervisedly by corrupting documents and then optimizing the model to reconstruct them by computing loss between the original document and the output of the model using the cross-entropy between the true and predicted tokens. Authors of [27] tried several mechanisms to corrupt the document, settling for two tasks that were crucial according to an ablatation study:

- **Task 1: Text Infilling**
  A certain number of text spans are sampled, with span lengths chosen at random. Each such span is replaced by a single `[MASK]` token. The model is then trained to predict the missing spans.
- **Task 2: Sentence Permutation**
  Sentences of a document are permuted randomly. The model is trained to output sentences of the document in the original order.

The multilingual version of BART, mBART, was introduced in [22]. It is again trained on a dataset extracted from Common Crawl, consisting of 25 languages. The model uses the same architecture and pretraining tasks as BART, but appends a special language token to the input and target sequence, as it is intended to be used for translation.

The mBART model again uses fixed-sized positional embedding. It can process input sequences with maximal length of 1024 tokens.

2.2.5 mT5

The *Text-to-Text Transfer Transformer (T5)* is a sequence-to-sequence model introduced in [23]. The architecture closely follows the original encoder-decoder Transformer up to some modifications. The layer normalization is applied before the residual connection. It is also simplified - activations are only rescaled and no additive bias is applied. Instead of using fixed positional embedding at each input position, T5 uses relative embeddings. In general, relative embedding produces learned embedding based on the offset between the key and query compared in the self-attention mechanism. T5 simplifies this and only uses a learned scalar as the positional embedding, which is added to the corresponding logit in attention computation. This learned positional embedding is shared across all the encoder and decoder blocks for efficiency. Pretraining is done on *Colossal Clean Crawled Corpus (C4)* created from Common Crawl. Authors of [23] used orders of magnitude more data than previous work. The dataset is also cleaned and preprocessed to contain only reasonably clean and natural English text. The pretraining task consists of leaving out random spans (approximately 15%) of the input sequence and training the model to predict them. Contrary to BART predicting the whole document, whose corrupted version is supplied as the input, T5 is trained to only predict the missing spans during the pretraining.

After the pretraining, the model was fine-tuned on several downstream tasks. All the downstream tasks were formulated as sequence-to-sequence problems (e.g., using `Translate to Czech: Hello world.` as input and teacher forcing the model to predict `Ahoj světe.`). With this representation, the model can be fine-tuned on several downstream tasks at the same time.

The T5 model also differs from previously released models by the amount of data and size of the model used. The relationship between the performance of the model on the downstream tasks and its scale in size was also explored. The general finding is that a model with more parameters pretrained on a larger amount of data for more training steps performs better.

To create a (massively) multilingual version of T5, authors of [18] created *mC4* - a multilingual version of C4 comprising of data in 101 languages. The resulting model, *mT5*, was pretrained on the same task and published in several model sizes. Both T5 and mT5 use the SentencePiece tokenizer. The vocabulary size of the multilingual version was increased to accommodate the need for more base tokens.

Due to the relative positional embedding, the mT5 model does not have a limitation on the length of the input size. Increasing input length, however, increases memory consumption and computational time significantly due to the attention mechanism.

2.2.6 AYA

Part of the success of the modern large language models is attributed to the Instruction Fine-tuning, which involves fine-tuning the model on pairs of *prompts* and corresponding *completions*. Fine-tuning on such data was shown to improve the helpfulness and instruction following capabilities of LLMs significantly. Most of the IFT datasets, however, are primarily English, making recent model advancements hard to reach for other (especially low-resourced) languages. This inequality motivated authors of [2] to create model AYA by curating a multilingual IFT dataset.

The largest pretrained version of the mT5 model (mT5-xxl) was used as a base model. It was further Instruction fine-tuned on a multilingual (101 languages) instruction dataset. The dataset was curated from multiple sources:

- Samples created by transforming specific multilingual datasets (e.g., QA or summarization) into instruction response pairs (as was done for fine-tuning mT5).
- Human annotated instruction samples (following the *prompt* - *completion* scheme) created by fluent speakers of various languages. Gathering a good quality multilingual IFT dataset was a large collaborative effort, which resulted in the AYA dataset [1], published in parallel with the model.
- Samples from existing English IFT datasets, which were translated into various languages.
- Samples of human-annotated prompts from ShareGPT[2] with synthetic English completions from Cohere's Command model[3]. The decision to use the completions from the Command model was made due to the ChatGPT's license which does not allow training on the outputs of their model. The English samples were then machine-translated into various languages.

Compared to previous works, the authors of AYA also significantly expanded the amount and number of types of evaluation schemes and datasets used.

[2] https://sharegpt.com/
[3] https://cohere.com/models/command

Using those various data sources of IFT samples, the authors of AYA experimented with an ablation study to obtain the optimal representation of the various languages and types of tasks in the training dataset to create a model with the best performance. Compared to other models comparable in size, also trained on multilingual instruct datasets (albeit smaller and containing fewer languages), the AYA achieved SOTA results on the proposed evaluation benchmarks.

2.2.7 Sentence Transformer

Sentence Transformer (previously called Sentence-BERT) [30] is motivated by sentence-pair NLP tasks such as semantic textual similarity. Previously, such tasks required feeding both sentences into a BERT-like model, which required many forward passes through the model to make multiple comparisons. This could have potentially been very computationally demanding if a large collection of sentences were to be considered. In Sentence Transformer, authors fine-tuned a pretrained model (originally BERT, but many other models are used as the base model today) using one of three objective functions:

- **Classification Objective Function** classifies pairs of sentences into three categories according to the *Natural Language Inference* (NLI) task (`contradiction`, `entailment`, `neutral`):

$$o = \mathrm{softmax}(W_t(u, v, |u - v|)), \tag{2.8}$$

  where u and v are embeddings of the two sentences, $(u, v, |u - v|)$ is a concatenation of the embeddings and their element-wise difference and $W_t \in \mathbb{R}^{3n\times 3}$ is a matrix of trainable parameters. A cross-entropy loss is minimized to optimize this objective function.
- **Regression Objective Function** computes cosine similarity between two sentences using their respective embeddings u and v. It is a regression task and a squared-error loss is optimized.
- **Triplet Objective Function** works by comparing three sentences: anchor a, a positive sentence (similar to anchor) p and a negative sentence (not similar to anchor) n. A triplet loss is computed as:

$$\max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0), \tag{2.9}$$

  where s_x is an embedding of sentence x and ϵ is the triplet loss margin.

The model was finetuned on several datasets using the appropriate objective function, outperforming other sentence embedding approaches on various benchmarks.

To create a multilingual Sentence Transformer, authors of [30] extended the monolingual Sentence Transformer in [24]. The original monolingual model (*teacher*) is used to create an embedding of an English sentence. The sentence is translated and a multilingual model (*student*) is trained to mimic the embedding of the English sentence from the teacher with its embedding of the translated sentence. They call this approach *multilingual knowledge distillation* and demonstrate its effectiveness in more than 50 languages.

2.3 Inference

During inference (summary generation) with generative language models, we assume autoregressive language generation. The assumption is that the probability distribution of the generated word sequence can be decomposed into the product of conditional next-word probabilities:

$$P(w_{1:T}|W_0) = \prod_{t=1}^{T} P(w_t|w_{1:t-1}, W_0), \tag{2.10}$$

where W_0 is the input text to be summarized. The length of the summary T is determined dynamically during generation by detecting that the model generates an end-of-sequence token.

There are multiple methods to approach the generation, and we describe the three most used ones. **Greedy search** is the simplest. At each timestamp t, it greedily selects word w_t with the largest probability as the next generated word:

$$w_t = argmax_w P(w|w_{1:t-1}) \tag{2.11}$$

While simple, greedy search comes with a drawback. The sampling might miss high-probability words that follow after low-probability words.

This problem can be alleviated by **Beam search**. During inference, it keeps the most likely `num_beams` of possible outputs (called hypotheses) at each timestamp. This setting ensures we generate an output sequence that is more probable than the one from greedy search.

Another approach is **Sampling**. Instead of always picking the most propable token (or n most propable tokens in case of Beam search), it samples the next word from the conditional distribution:

$$w_t \sim P(w|w_{1:t-1}). \tag{2.12}$$

This makes the generation non-deterministic. The distribution $P(w|w_{1:t-1})$ is usually tampered with to make it more likely that high-probability words are sampled. The output softmax *temperature* T can be modified:

$$\frac{e^{z_i/T}}{\sum_j e^{z_j/T}}, \tag{2.13}$$

to create a more or less sharp distribution. *Top-K Sampling* can be employed. It samples from only the top k most probable predictions. Choosing the right k can be problematic with this method, as the probability mass encompassed by the top-k most probable tokens can vary greatly. *Top-p Sampling* is more satisfactory in this regard. It chooses the smallest amount of the most probable words until the cumulative probability exceeds the threshold p.

2.4 Parameter-Efficient Fine-Tuning

Modern SOTA language models significantly increased in scale, utilizing large amounts of GPUs to train a single model for weeks or months at a time. The scale increased to such an extent that fine-tuning these models to downstream tasks can be too costly or impractical. With some models and setups, even just the inference can be too slow or memory intensive. Multiple approaches are being researched and utilized to alleviate this.

An often-used technique to decrease computation time and memory is to use more efficient float data types. The commonly used options are `fp16` (used with mixed precision training), `bf16` (available on Ampere and newer GPU architectures, also utilized with mixed precision training), or `tf32` (also available on Ampere and newer). While making it possible to do training and inference more efficiently, those techniques alone do not solve the presented problem with large-scale language models. We follow with a description of three techniques that aim to make the training or inference more efficient and accessible: *Low-Rank Adaptation of Large Language Models*, *Quantization* and *Efficient Finetuning of Quantized LLMs*.

2.4.1 LoRA: Low-Rank Adaptation of Large Language Models

Low-Rank Adaptation (LoRA) [16] focuses on making fine-tuning more efficient. During fine-tuning, traditionally, all weights of the pretrained model are updated and the fine-tuned model contains the same amount of parameters as the pretrained model. This is not only inconvenient but can also cause challenges when deploying multiple fine-tuned models for various downstream tasks. For reference, the AYA model we use has 13B parameters, but significantly larger models, like GPT-3 [20] with 175B trainable parameters, exist. Only the process of loading such models to the memory is time-consuming, considering that during deployment, the need to switch to a different fine-tuned version to process another downstream task might arise.

LoRA solves this by freezing the weights of the pretrained model and only training injected rank decomposition matrices of the model parameters. This is inspired by previous works [34] [19] that show that learned parameters of over-parametrized models (which LLMs certainly are) reside on low intrinsic dimension. Denoting parameters of the pretrained model Φ_0 and the updated parameters as $\Phi_0 + \Delta\Phi$, authors of LoRA then hypothesize that the change in model's weights during fine-tuning $\Delta\Phi$ is also of low rank. They therefore propose to represent the parameter increment with fewer parameters Θ as $\Delta\Phi = \Delta\Phi(\Theta)$, where $|\Theta| \ll |\Phi_0|$. Choosing a low rank for the representation of the weights increment, the number of parameters to train during LoRA fine-tuning can be as low as 0.01% of the original model size.

The low-rank representation of parameter update is illustrated in Fig. 2.2. Layers of

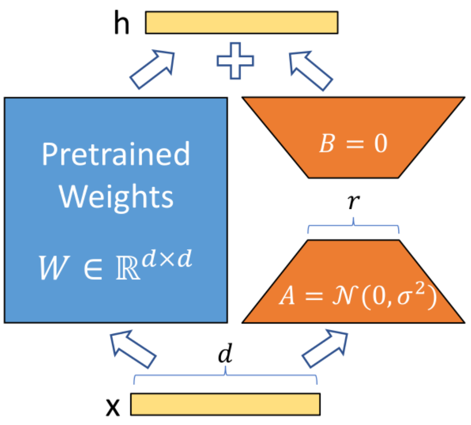
h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 2.2: Visualization of LoRA reparametrization. Taken from [16]. W denotes the original pretrained weights (denoted W_0 in (2.14)). Matrices A and B are used to represent the change in weights during fine-tuning, their initialization is denoted in the figure.

language models can be represented with matrix multiplications. LoRA represents the update of matrix weights as:

$$W_0 + \Delta W = W_0 + BA, \tag{2.14}$$

where $W_0, \Delta W \in \mathbb{R}^{d \times k}$ are the original pretrained weights and the change of the weights through fine-tuning, respectively. $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ are the matrics of the low-rank decomposition of ΔW and $r \ll \min(d, k)$ is the rank. The W_0 is frozen during training (it is not updated with backpropagation). Output of matrix multiplication (forward pass) $h = Wx$ can than be represented as:

$$h = (W_0 + \Delta W)x = (W_0 + BA)x, \tag{2.15}$$

where x is the input and h is the output of the matrix multiplication. When applied to language models (the Transformer architecture), only the attention weights W^Q, W^K, W^V and W^O (see Equation (2.3)) are optimized.

This setting reduces the memory usage during training by up to 2/3. Switching two fine-tuned models based on the same pretrained weights amounts to simply subtracting one set of the low-rank parameter updates and adding another one for a different downstream task.

Through empirical experimentation, authors of LoRA show that this proposed fine-tuning scheme does not lead to a significant decrease in the performance of the models.

2.4.2 Quantization and QLoRA: Efficient Finetuning of Quantized LLMs

Quantization focuses on representing each of the model's parameters using fewer bits, utilizing either k-bit float or integer. This is done to decrease the memory footprint and improve inference speed. There is usually a tradeoff between the quantized model accuracy and memory and run time reduction. This tradeoff and various quantization techniques are discussed, for example, in [4].

While the quantization techniques alone can reduce the memory footprint of LLMs significantly, they only work for inference and their use for fine-tuning is not straightforward. Motivated by this, authors of *QLoRA: Efficient Finetuning of Quantized LLMs* [3] combine a new quantization technique with LoRA fine-tuning approach and show that it is possible to fine-tune quantized 4-bit model without performance degradation compared to standard 16-bit fine-tuning. To combine LoRA and quantization, the authors of QLoRA introduce a novel data type and improvements to the quantization mechanism and optimizer.

2.5 Evaluation

Evaluating text generation tasks is a notoriously hard problem and an open research question. In this thesis, we use n-gram overlap metric *Rouge* and two model-based metrics *BERTScore* and *AlignScore-CS* to automatically assess the quality of the generated summary. While the first two metrics are designed to compare the generated summary with a human-written reference summary, AlignScore-CS focuses on assessing the factuality of the generated summary using the source document as a reference.

All the metrics we use output a score in the range (0, 1). We rescale it by multiplying it by 100 for better readability.

2.5.1 Rouge

Rouge, or Recall-Oriented Understudy for Gisting Evaluation [43], is an automatic metric for evaluating text similarity, used mainly to evaluate automatic summarization and machine translation. It compares the automatically produced text against a human-produced reference (or multiple references). It is a traditionally used evaluation metric in NLP research, as it is simple, computationally efficient and easily interpretable. In this work, we use the two most frequently used variants of rouge: *Rouge-N* and *Rouge-L.*

In the original paper [43], **Rouge-N** is introduced as the recall between candidate and reference summary:

$$\text{Rouge-N}_{\text{R}}(r,c) = \frac{\sum_{\text{gram}_n \in r} \text{Count}_{\text{match}}(\text{gram}_n, r, c)}{\sum_{\text{gram}_n \in r} \text{Count}(\text{gram}_n, r)}, \tag{2.16}$$

where r, c are the reference (human-produced) and the candidate (automatically generated) summaries respectively, n is the length of n-grams gram_n, $\text{Count}(\text{gram}_n, r)$ is the amount of n-grams gram_n in the reference summary r and $\text{Count}_{\text{match}}(\text{gram}_n, r, c)$ is the number of n-grams gram_n co-occurring in the reference summary r and the candidate summary c.

A precision rouge metric can also be defined:

$$\text{Rouge-N}_{\text{P}}(r,c) = \frac{\sum_{\text{gram}_n \in r} \text{Count}_{\text{match}}(\text{gram}_n, r, c)}{\sum_{\text{gram}_n \in c} \text{Count}(\text{gram}_n, r)}, \tag{2.17}$$

which can be used together with the recall to compute the F-score:

$$\text{Rouge-N}_{\text{F}}(r,c) = 2\frac{\text{Rouge-N}_{\text{R}}(r,c)\text{Rouge-N}_{\text{P}}(r,c)}{\text{Rouge-N}_{\text{R}}(r,c) + \text{Rouge-N}_{\text{P}}(r,c)}. \tag{2.18}$$

We report the F-score when comparing our methods in this work (as do most recently published works on summarization) and compute the Rouge-1 and Rouge-2 versions using unigrams and bigrams respectively.

Rouge-L compares reference and candidate using their longest common subsequence. The recall, precision and F-score are computed as

$$\text{Rouge-L}_{\text{R}}(r,c) = \frac{\text{LCS}(r,c)}{|r|} \tag{2.19}$$

$$\text{Rouge-L}_{\text{P}}(r,c) = \frac{\text{LCS}(r,c)}{|c|} \tag{2.20}$$

$$\text{Rouge-L}_{\text{F}}(r,c) = 2\frac{\text{Rouge-L}_{\text{R}}(r,c)\text{Rouge-L}_{\text{P}}(r,c)}{\text{Rouge-L}_{\text{R}}(r,c) + \text{Rouge-L}_{\text{P}}(r,c)}, \tag{2.21}$$

where $\text{LCS}(r,c)$ is the length of the longest common subsequence of r and c.

The original implementation published with the paper [43] uses stemming and stop word removal as a preprocessing step. However, this preprocessing is prepared only for the English language. We instead use a Rouge-raw version introduced in [35], which is language agnostic and computes n-grams without preprocessing using stemming or stop word removal. Stop word removal is not a straightforward task for the medical text data we use, as removing some words can noticeably change the meaning of a sentence. We further tried to use a lemmatization tool for the Czech language [40], but due to often occurring abbreviations and imperfect stylization of the text, it did not perform well.

2.5.2 BERTScore

BERTScore [25] is a model-based automatic text similarity metric. It leverages preraineded encoder-only model BERT to create contextual embeddings of words (more specifically, tokens) in reference and candidate text, which it then matches using cosine similarity. It has been shown to correlate with human judgment. The metric computation is visualized in Fig. 2.3. More specifically, precision, recall and F-score metrics can be computed as follows:

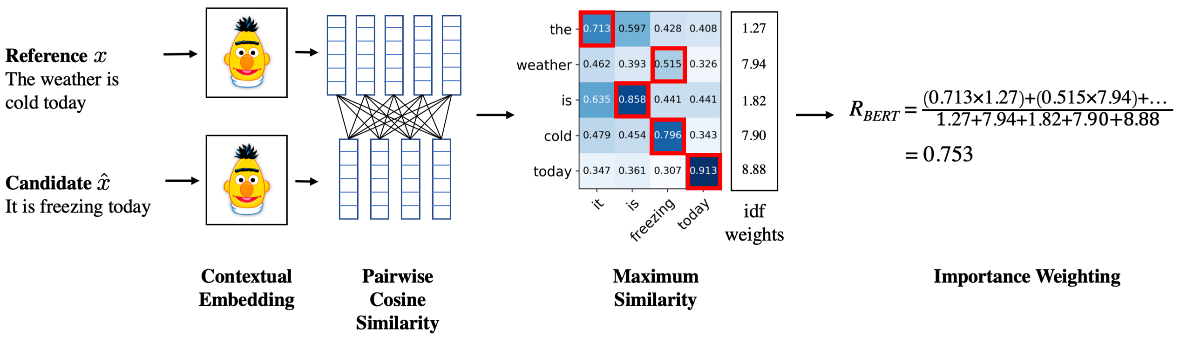

Figure 2.3: Illustration of BERTScore computation. Taken from [25]. Tokens of reference and candidate text are embedded using an encoder-only language model. The embeddings are compared in a pairwise manner using cosine similarity. The score is then computed using (2.22) - (2.24).

$$\text{BERTScore}_\text{R}(r, c) = \frac{\sum_{r_i \in r} \max_{c_j \in c} \mathbf{r}_i^\top \mathbf{c}_j}{|r|} \tag{2.22}$$

$$\text{BERTScore}_\text{P}(r, c) = \frac{\sum_{c_j \in c} \max_{r_i \in r} \mathbf{r}_i^\top \mathbf{c}_j}{|c|} \tag{2.23}$$

$$\text{BERTScore}_\text{F}(r, c) = 2\frac{\text{BERTScore}_\text{R}(r, c)\text{BERTScore}_\text{P}(r, c)}{\text{BERTScore}_\text{R}(r, c) + \text{BERTScore}_\text{P}(r, c)}, \tag{2.24}$$

where r and c is the reference and candidate summary respectively, $r_i \in r$ and $c_j \in c$ are individual tokens of reference and summary and $\mathbf{r}_i$ and $\mathbf{c}_j$ are contectual embeddings (outputs of BERT model) of r_i and c_j respectively. The embedding vectors $\mathbf{r}_i$, $\mathbf{c}_j$ are normalized, the cosine similarity is therefore simply $\mathbf{r}_i^\top \mathbf{c}_j$. The original paper also uses IDF weights to include importance weighting of individual tokens when computing the final score. This is, however, by default not used, as the corpus is usually too small to learn proper IDF weights. We use the default setting.

In comparison to Rouge, BERTScore considers the context in which words (represented by tokens) are used, making it theoretically able to recognize synonyms and infer the meaning of unknown words from the context.

Authors of BERTScore studied how the metric compares with human judgment. They used the results to choose the best-pretrained language model to use to compute the contextual embeddings of the inputs. For the Czech language, layer 9 of 'bert-base-multilingual-cased' is the recommended one to use.

2.5.3 AlignScore-CS

Another metric we use to compare the models, *AlignScore* [12], focuses on factuality. It measures factual consistency between text pairs. More specifically, its goal is to evaluate whether all information in a *claim* is contained in a *context*, ensuring that the *claim* does not contradict it. Having the metric defined as such, it is natural to use it to evaluate the factual consistency of a summary generated by a language model.

Authors of AlignScore took a holistic approach to training the metric, motivated by limitations of previous works. Previously, similar metrics were usually trained on a single NLP task (sometimes from a single domain), leading to limited generalizability. The training procedure of AlignScore unifies a wide range of data sources and tasks to train a general information alignment model. More specifically, 15 datasets spanning 7 NLP tasks were used for the training. The utilized tasks are NLI (Natural Language Inference), QA (Question answering), paraphrasing, fact verification, information retrieval, semantic similarity and summarization. The authors evaluated the metric on various large-scale evaluation benchmarks and showed that their approach substantially outperforms previous SOTA metrics. It is also on par with, or better than, metrics based on orders-of-magnitude larger language models (such as GPT-4).

The main part of the metric is the *Unified Alignment Function* trained by unifying training of the aforementioned NLP tasks. All the NLP tasks are transformed into a single text input containing a text pair: text a (the context) and text b (the claim). The Unified Alignment Function is trained with three output heads, each for a different type of the alignment label y: binary, 3-way classification (choosing between `ALIGNED`, `NEUTRAL` and `CONTRADICT`) or regressive. The output head is chosen based on the presented task. The unification of the training is visualized in Fig. 2.4.

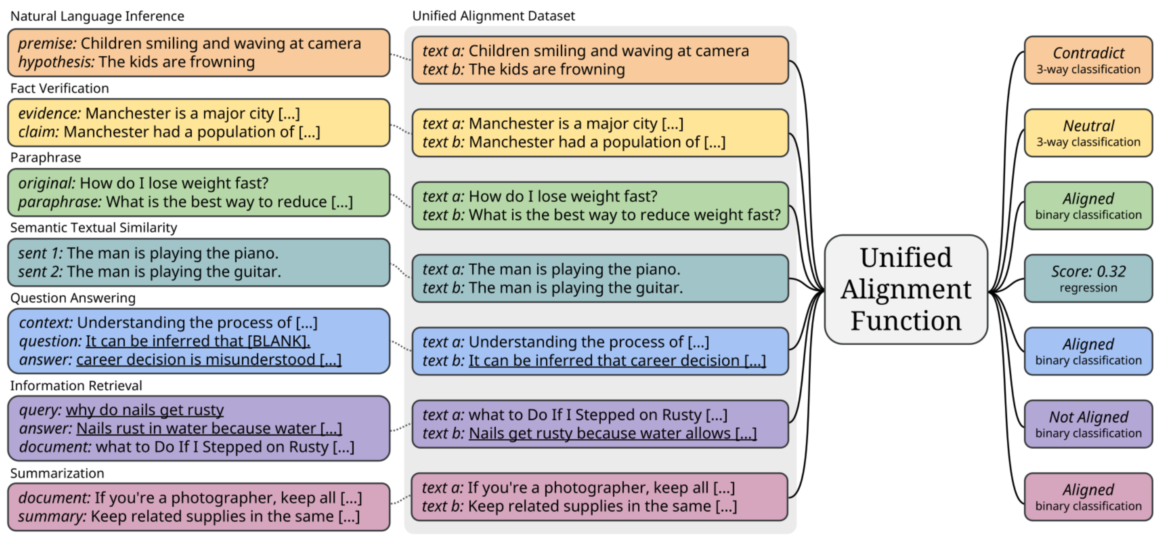
Natural Language Inference
premise: Children smiling and waving at camera
hypothesis: The kids are frowning
Fact Verification
evidence: Manchester is a major city [...]
claim: Manchester had a population of [...]
Paraphrase
original: How do I lose weight fast?
paraphrase: What is the best way to reduce [...]
Semantic Textual Similarity
sent 1: The man is playing the piano.
sent 2: The man is playing the guitar.
Question Answering
context: Understanding the process of [...]
question: It can be inferred that [BLANK].
answer: career decision is misunderstood [...]
Information Retrieval
query: why do nails get rusty
answer: Nails rust in water because water [...]
document: what to Do If I Stepped on Rusty [...]
Summarization
document: If you're a photographer, keep all [...]
summary: Keep related supplies in the same [...]
Unified Alignment Dataset
text a: Children smiling and waving at camera
text b: The kids are frowning
text a: Manchester is a major city [...]
text b: Manchester had a population of [...]
text a: How do I lose weight fast?
text b: What is the best way to reduce weight fast?
text a: The man is playing the piano.
text b: The man is playing the guitar.
text a: Understanding the process of [...]
text b: It can be inferred that career decision [...]
text a: what to Do If I Stepped on Rusty [...]
text b: Nails get rusty because water allows [...]
text a: If you're a photographer, keep all [...]
text b: Keep related supplies in the same [...]
Unified Alignment Function
Contradict 3-way classification
Neutral 3-way classification
Aligned binary classification
Score: 0.32 regression
Aligned binary classification
Not Aligned binary classification
Aligned binary classification

Figure 2.4: Unified Alignment Function Training on various NLP tasks. Taken from [12]. The figure shows examples of various NLP tasks (on the left) being transformed into the unified (text a + text b) format (in the middle). The function outputs corresponding labels based on the task (on the right).

The Unified Alignment Function is computed by computing the embedding of the input sequence (concatenation of text a and text b) using the RoBERTa language model. The contextual embedding of the classification token (beginning-of-sequence token) is then processed with one of the three individual heads for either the 3-way classification (logistic regression), binary classification (logistic regression) or linear regression. The joint loss function is computed as a sum of the losses from the individual tasks.

Instead of naively computing the AlignScore metric by passing context and claim into the Unified Alignment Function, the context is split into roughly 350-token chunks, and the claim is split into sentences. Each sentence of the claim is then evaluated against each context chunk using the alignment function. For each claim sentence, the highest alignment score is selected. The alignment scores of sentences of the claim are then averaged to compute the final factual consistency score. The computation is illustrated in Fig. 2.5.

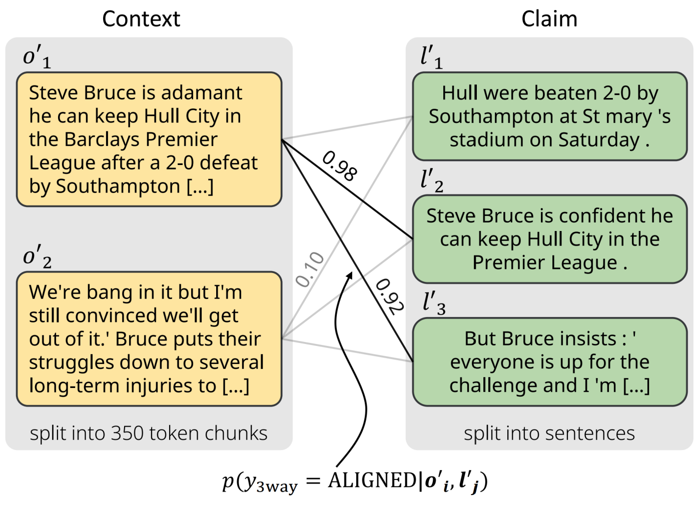
Context
Claim
o'_1
Steve Bruce is adamant he can keep Hull City in the Barclays Premier League after a 2-0 defeat by Southampton [...]
o'_2
We're bang in it but I'm still convinced we'll get out of it.' Bruce puts their struggles down to several long-term injuries to [...]
split into 350 token chunks
l'_1
Hull were beaten 2-0 by Southampton at St mary 's stadium on Saturday .
l'_2
Steve Bruce is confident he can keep Hull City in the Premier League .
l'_3
But Bruce insists : ' everyone is up for the challenge and I 'm [...]
split into sentences
0.98
0.10
0.92
$p(y_{3way} = \text{ALIGNED}|\boldsymbol{o'}_i, \boldsymbol{l'}_j)$

Figure 2.5: AlignScore visualization Taken from [12]. The figure shows the splitting process of the context into roughly 350-token chunks. Each context chunk is compared with each claim sentence. The AlignScore is computed by gathering the maximal score for each claim and then taking the average.

This chunking was shown to perform better than evaluating the alignment function on the sentence or document level. Further, it allows AlignScore to evaluate context-claim pairs even if the context is longer than the 512 maximal supported input sequence length of the RoBERTa model. By default, the output of the 3-way classification head (probability of the model predicting the `ALIGNED` label) is used to compute the alignment during inference, as it was shown in an ablation study to perform the best.

Until now, we discussed the original AlignScore published in [12], which was fine-tuned using only English datasets, using the RoBERTa model which itself was pretrained on an English-only corpus. To make it possible to use AlignScore for Czech data, colleagues of mine, Marian Krotil and Martin Hubal, translated the used training and benchmarking data to Czech. They used both Czech and English data to train a Czech version *AlignScore-CS* and replaced the pretrained RoBERTa with its multilingual version XLM-RoBERTa. Otherwise, they followed the training procedure from [12]. The trained metric showed similar results to the original AlignScore on both the translated and the English benchmarks.

Chapter 3

Related Work

In this chapter, we briefly discuss research related to the goal of this thesis. Text summarization is a well-researched natural language processing task. Relevant publications, however, mostly focus on summarization datasets consisting of general domain English text. Our datasets are in the Czech language and of a specific medical domain. The amount of related publications is therefore severely limited. We discuss some works, all focusing on English data, that either attempt to solve a similar task to ours, or research another summarization problem with medical data.

A similar task to ours, generating a *Brief Hospital Course* (BHC) summary, was explored in [9], [7], [13] and [14]. BHC is a succinct summary of an entire hospital encounter embedded within a discharge summary (an equivalent of the discharge report used in Czech healthcare). The task is formulated as generating a BHC summary by multi-document summarization, utilizing all kinds of medical notes describing the patient's hospital stay. Per this description, the task is similar to what we are trying to achieve with the datasets from IKEM.

Authors of [9] experiment with generating BHC summary on two datasets. MIMIC-III [38] is a dataset of US Internal Care Unit (ICU) patient admissions and contains data about 47,951 unique admissions. The other dataset, KCH, was extracted from [31]. It is a dataset consisting of 1,586 admissions of UK patients who suffered a stroke. The work experiments with both extractive and abstractive methods and also with their ensemble. The authors report better performance using the abstractive methods, compared to extractive summarization. Specifically, they use the BART model and show better performance with first pretraining it on a SumPubMed [15], a summarization dataset of English scientific medical articles. Interestingly, using the T5 model yields worse results compared to BART in their experiments. This is contrary to the results we report in Chapter 6, where the multilingual version of T5 outperforms the multilingual version of BART.

Authors of [7] and [13] study generating the whole discharge summary using the in-patient records (a type of daily notes about the patient during hospitalization). Both works conduct a data analysis to find out if the task is possible. In other words, they study if the source texts contain enough information to successfully generate the summaries. They conclude all information needed to generate the discharge summary is not always available in the text records documenting a patient's hospital stay. Physicians sometimes use a piece of information they remember or make a diagnostic conclusion only when writing the discharge summary. Other times, the information in the hospitalization summary is contained in the patient's past medical records, examination results, surgical reports or medication notes.

Further, according to [14], as much as 46% of the discharge summary is copied and pasted as is from past medical records and 36% imported from structured data sources. This leaves a relatively small portion of the discharge report to be written manually by the physician, which serves as a good motivation for automating the task.

In [11], the task of medical text summarization was studied across multiple English clinical text datasets. The study compared several LLMs, choosing GPT-4 as the best-performing option. With this model, they further conducted a study with several physicians to compare if they preferred the automatically or manually created summary, measuring *completeness*, *correctness* and *conciseness*. The results showed that the GPT-4 generated summaries are more complete and contain fewer errors compared to the human summaries.

We also want to highlight another useful application of text summarization in healthcare, further motivating the use of language models in this domain. Another vital part of healthcare is a patient-doctor conversation, which is followed by the physician writing structured notes describing the patient's health state and plan for treatment. Writing such notes is another time-consuming work medical experts need to focus on. Authors of [8] and [17] experiment with automating the process using language models to create structured notes.

All the related works focus on using English clinical datasets. While the difference in language compared to our datasets is obvious, the structure of clinical notes in different countries can also differ, making us cautious to make strong conclusions. Some of the proposed techniques are not even possible. This includes pretraining the abstractive summarization model on a large medical summarization corpus, before fine-tuning it on the downstream task (generation of the hospitalization summary), as there is no available large medical text corpus in the Czech language. Although some of the works show promising results, generally the topic of using language models in the clinical domain has not been thoroughly researched yet.

Chapter 4

Datasets

We were provided with two datasets from *Institute for Clinical and Experimental Medicine (IKEM)*. The datasets consist of thousands of samples. Each sample contains several text paragraphs containing information about a patient's hospitalization. The paragraphs are extracted from an admission report and a discharge report, both concerning a single hospitalization.

The samples are anonymized. We are only provided with a generated patient's ID. As there are patients with multiple hospitalizations in the dataset, we use the IDs to prevent cross-contamination of the training and validation set with hospitalizations of the same patients. Dates of procedures, patient ages, and all names are also redacted.

The two datasets are from different departments: Acute Cardiology and Hepatogastroenterology. In Sec. 4.1, we first describe the datasets' structure, which is identical for both. Then, we define the summarization task we are trying to solve with the data in Sec. 4.2. Next, in Sec. 4.3, we outline the process used to clean and preprocess the data. We finish with analyzing the datasets and describing how they differ in Sec. 4.4, setting ground for explaining why some methods might work better on samples from one department compared to another.

To illustrate the datasets, we show one sample from each preprocessed dataset in Appendix A.

4.1 Dataset structure

Each hospitalization sample contains two IDs and several text paragraphs:

- **Hospitalization ID**
  Generated ID unique for each hospitalization.
- **Patient ID**
  Generated ID of each patient. One patient might have gone through several hospitalizations. With this ID, we can ensure we create the training, validation, and test splits such that no two splits contain hospitalizations of the same patient.
- **Current problems** (`Nynější onemocnění`)
  During patient admission, the patient's history and current condition are considered, and a report is written. Extracted from the current problems paragraph of the admission report.
- **Admission findings** (`Objektivní nález`)

Describes results of patient examination during admission. Extracted from the admission report.

- **Admission conclusion** (`Závěr při přijetí`)
  Conclusion paragraph from the admission report.
- **Hospitalization reasons** (`Důvod hospitalizace`)
  Summary of reasons for the hospitalization. Extracted from the discharge report.
- **Procedures** (`Operace`)
  Summary of interventions performed on the patient during the hospitalization. Extracted from the discharge report.
- **Examinations** (`Vyšetření`)
  Summary of examinations performed on the patient during the hospitalization. Extracted from the discharge report.
- **Hospitalization summary** (`Průběh hospitalizace`)
  Overall summary of the hospitalization from the discharge report.

4.2 Task description

A natural way to formulate a summarization task on the dataset structure described above is to generate the *Hospitalization summary* using all the other text paragraphs as a source. The hospitalization summarization contains more concise information about the patient's admission, a summary of important findings, and the results of the interventions and medical examinations. It is also the last part of the discharge report the doctor gets to write, and it should be a reasonably concise and informative paragraph summarizing the patient's hospitalization, pinpointing the crucial information for ensuring good continuity of medical care.

A more realistic task of creating the hospitalization summary would be better defined as summarizing not only the text paragraphs provided in the dataset but also utilizing all the text medical documentation about the patient's hospitalization. This includes the patient's historical documentation, the admission report, a full report of each examination and intervention carried out during the hospitalization, and all daily notes and reports. With all such data available as the source, we could even attempt to solve the task of writing the whole discharge report, which could still be reasonably defined as a summarization task, generating the individual paragraphs of the report individually, possibly with the help of some retrieval methods.

As the focus of this thesis is on an initial exploration of using NLP methods for medical texts in the Czech language, we focus on the simplified task defined above to make the problem more tractable, considering, for example, the limitations on the length of the input sequence of the used language models.

4.3 Preprocessing the data

We preprocess the provided datasets in several ways to make them more fit for the summarization task we just defined. We first create the source input by concatenating the following paragraphs: Current problems, Admission findings, Admission conclusion, Hospitalization reasons, Procedures, and Examinations. Before the concatenation, each paragraph is prepended with its name in Czech (for example, `Důvody hospitalizace` for Hospitalization reasons). Due to the universal nature of the sequence-to-sequence language models, such source input created by concatenation and the hospitalization summarization paragraph used as the target summary could already be used for fine-tuning a language model. There are, however, several other ways we preprocess and clean the dataset to make it more suitable for our experiments.

Due to the limited input sequence length some of the language models we chose to use in our experiments can process (namely the mBart and XLM-RoBERTa), we create a subset of the datasets by filtering out samples with source input longer than the allowed length (which is 1024 tokens for both the models, after modifying the embedding layer of XLM-RoBERTa, as explained later in Sec. 5.3). Filtering shorter discharge reports also approximately coincides with filtering the reports of less complicated hospitalizations. Such hospitalizations are usually planned. The patient comes for scheduled examinations or interventions, and nothing unexpected happens during the hospital stay. The amount of interventions and examinations is usually lower than during a more complicated hospitalization. The discharge reports of such hospitalizations are usually shorter, and the hospitalization summary is more predictable and less complicated. Therefore, we hypothesize that solving the task on this filtered subset should be easier and can serve as a good stepping stone to experiment with our methods.

While the filtered dataset with shorter input sequences allows us to compare all methods we use on the same data, we also fine-tune some models on the complete unfiltered dataset. The performance might be worse due to the more complicated hospitalizations, but the complete datasets are approximately double the size of the filtered ones. As the number of samples in the filtered datasets is considerably low, and the performance of fine-tuned models generally increases with more training samples, we want to experiment with the influence of these two factors.

By empirically analyzing the dataset, we found that the hospitalization summary paragraph tends to contain some sentences with information that cannot be found in the concatenated source. Examples include information about the discharge of the patient (e.g., *Dimitován v kardiopulmonálně stabilním stavu.*, which translates to *The patient was discharged in cardiopulmonary stabilized state*), change in medication (e.g., *Z medikace vysazen Myfenax a Valcyte, navýšen Prednison.*, or *Myfenax and Valcyte was withdrawn from the medication, dosage of Prednison was increased.* in English) or some diagnostic conclusion that is included in neither the Examinations nor Procedures paragraph.

For most of these *unsourced* sentences, we should be able to find the source of information in the other text documentation concerning the hospitalization (for example, the patient's medical condition on the day of discharge could be found in the appropriate daily note). However, we speculate there might be some unsourced sentences that could not be generated, even if the model had access to all the existing relevant text medical documentation. This could happen if some of the unsourced sentences are written based on information the doctor remembers or infers (makes a diagnostic conclusion) during the discharge report writing. Generally, if the hospitalization summary paragraph contains information that cannot be found elsewhere, the task of automatically generating it cannot be achieved perfectly. We want to avoid training the model to generate such unsourced sentences, as this might cause it to hallucinate or attempt to make a diagnostic conclusion based on the available information, which is not how we formulate the task. Instead, we want the model to extract and reformulate existing information, even if that means leaving the hospital summarization paragraph incomplete, requiring further review and completion by the doctor. While it might be true that language models could generate a valid diagnostic conclusion given sufficient context and relevant information (some works do focus on this area of research, for example, [10]), it is not how we formulate the task in this thesis.

To alleviate the possible problem with hallucinations, we attempt to filter out two categories of the unsourced sentences that we empirically found to appear often in the data. Using a regex heuristic, we attempt to filter out sentences from the hospitalization summary in the Acute Cardiology dataset if they contain information about the discharge of the patient or medication modification. In the Hepatogastroenterology dataset, we decided only to filter out the discharge sentences, as filtering out the sentences about medication modification was too noisy, discarding valid sentences often. Further, it contained unsourced medication sentences less often than the Acute Cardiology dataset.

Finally, we cleaned all special characters, replaced anonymized dates with shorter sequences to safe input sequence lengths (using `DD` instead of anonymized date `XX.XX.XXXX`), and filtered out samples (hospitalizations) with empty Hospitalization summary paragraph, as we considered those to be faulty, creating unnecessary noise in the training dataset.

We gather all the patients (using the Patient ID) for each dataset and split them into train, validation, and test subsets. We then create the dataset splits by fetching all the hospitalizations of the patients in the corresponding splits.

4.4 Analysing the datasets

The samples of both datasets are of the same structure, but their content differs to some extent. Medical conditions with which patients are hospitalized are different but might overlap in some cases. During hospitalizations, patients go through different types of examinations and operations depending on the department. Different departments might have different stylistic requirements and habits, such as how concise or fluent the paragraphs must be.

Particularly, we noticed that in samples from the Hepatogastroenterology dataset, sentences tend to be more fluent, and they are more often being reformulated when transcribed from the source paragraphs to the hospitalization summary. In samples of the Acute Cardiology dataset, sentences seem to be more often copied as they are when transcribed into the hospitalization summary, being reformulated only slightly and less often. We support this observation by computing the Rouge score of the true summary in Table 4.1, using the source text as the reference. We believe this might cause summarization methods to perform better

Dataset	Rouge-1	Rouge-2	Rouge-L
Acute Cardiology	**20.08**	**11.87**	**15.61**
Hepatogastroenterology	15.17	7.63	11.38

Table 4.1: Comparsion of how extractive are summaries written by a doctor. The Rouge score between the true hospitalization summary and the source text is computed for both datasets.

on the Acute Cardiology dataset, primarily when evaluated using the automatic text similarity metrics, as the model can learn to extract spans of the source while reformulating them only slightly.

We now briefly summarize both datasets' sizes in terms of the number of samples and lengths of the inputs. In table Table 4.2, we describe and compare the number of samples (hospitalizations). The Hepatogastroenterology dataset is considerably smaller. Due to its smaller size, we only use 500 patients for the validation and test split, leaving the remaining patients for training, The larger Acute Cardiology dataset allows us to use more (800) patients for validation and testing.

Split	Acute Cardiology			Hepatogastroenterology		
	#Patients	#All	#Short	#Patients	#All	#Short
Train	11015	13026	5983	3099	5203	2543
Validation	800	952	433	500	821	428
Test	800	926	421	500	851	400

Table 4.2: Size of the datasets in terms of number of patients and hospitalization samples. #All denotes the size of the complete dataset, while #Short denotes the size of the dataset with long samples filtered out.

To get a better idea about the size of the individual samples, we look at some basic statistics of the filtered Acute Cardiology dataset in Table 4.3 and the complete Acute Cardiology dataset in Table 4.4. For this, we compute the lengths of the source and summary in

		Mean	Std	Q1	Q2	Q3
Source	#Sentences	18.5	6.3	14	18	23
	#Words	340.3	65.8	297	347	391
	#Tokens	841.7	154.6	739	862	966
Summary	#Sentences	5.0	2.0	4	5	6
	#Word	68.8	27.5	50	66	84
	#Tokens	165.2	65.0	120	158	201
Token compression ratio		5.9	3.0	4.1	5.3	6.9

Table 4.3: Basic statistics for the filtered Acute Cardiology dataset

		Mean	Std	Q1	Q2	Q3
Source	#Sentences	27.0	12.9	18	24	34
	#Words	491.2	187.4	354	452	596
	#Tokens	1205.4	450.9	882	1104	1452
Summary	#Sentences	5.9	2.7	4	5	7
	#Word	86.0	38.8	59	79	106
	#Tokens	206.6	93.1	141	191	255
Token compression ratio		6.7	3.2	4.5	5.9	7.6

Table 4.4: Basic statistics for the complete Acute Cardiology dataset

the context of sentences, words, and tokens (using the mT5 tokenizer).

We compute the same basic statistics for the filtered Hepatogastroenterology dataset in Table 4.5 and the complete Hepatogastroenterology dataset in Table 4.6. Statistics of both the Acute Cardiology and the Hepatogastroenterology datasets are comparable, considering both the filtered and complete versions. The datasets consisting of all samples contain longer samples than the filtered versions, but the difference is not very large. Still, we see that the median is approximately 1000, which results in halving the size of the datasets when filtering out samples longer than 1024 tokens.

We note that the relation between the number of words and tokens is relatively small

		Mean	Std	Q1	Q2	Q3
Source	#Sentences	17.6	6.0	13	17	22
	#Words	326.0	63.3	283	332	376
	#Tokens	828.3	157.3	723	843	954
Summary	#Sentences	3.7	1.8	3	3	5
	#Word	58.5	28.6	38	52	73
	#Tokens	144.3	71.6	94	129	181
Token compression ratio		7.0	3.5	4.6	6.3	8.5

Table 4.5: Basic statistics for the filtered Hepatogastroenterology dataset

		Mean	Std	Q1	Q2	Q3
Source	#Sentences	28.1	15.7	17	24	35
	#Words	489.7	225.7	332	423	583
	#Tokens	1234.1	555.9	842	1071	1466
Summary	#Sentences	4.8	2.7	3	4	6
	#Word	79.6	45.8	47	69	101
	#Tokens	197.5	113.8	115	171	252
Token compression ratio		7.4	3.8	4.8	6.5	9.0

Table 4.6: Basic statistics for the complete Hepatogastroenterology dataset

(less than 0.5 words per token). In contrast, one token represents approximately 0.75 words on a general-domain English corpus. First, this difference is caused by using Czech (an under-represented language in the dataset used for pretraining), and second, by the nature of the medical domain text we are dealing with. Some abbreviations (e.g., abbreviations for medical conditions or drugs) are not occurring enough (if at all) in the pretraining dataset used to train the tokenizer. Those abbreviations are therefore usually represented by tokens character by character. This causes the limited input sequence length of the language models to be a greater burden than if we were to use general-domain English data, which is more efficiently represented by the tokenizer. It can also hinder the performance on the summarization task, as the abbreviations and unknown words are not presented to the attention mechanism as a single object, making it harder to learn the appropriate contextual relationships.

We also compute the ratio of the number of source tokens to the number of summary tokens, showing us how 'compressive' the model should be. This number is slightly higher for the Hepatogastroenterology dataset.

Chapter 5

Methodology

This chapter describes the summarization methods we use to solve the task presented in Sec. 4.2. We discussed some requirements for the proposed methods for the task of automatic hospitalization summary generation with IKEM. The data occurring in the hospitalization samples are of a sensitive nature, as they contain private medical information. This makes it desirable to use an on-premise solution to process such data, utilizing both open-source language models and hardware owned by the facility, instead of using LLMs provided as a service (e.g., OpenAI's GPT or Google's GEMINI). Solving the task on-premise is also preferable in terms of reliability and consistency. While companies such as OpenAI and Google can provide very reliable service (concerning speed and uptime), the LLMs they provide tend to be updated often, possibly changing their behavior and making the consistency of a solution utilizing them questionable. Naturally, the requirement to solve the task on-premise also brings the question of computational demands. While the companies providing LLMs as a service have a vast amount of computing power to run massive language models, we need to consider the scale of the models if they should be used on-premise. For this, we compare language models of various sizes, investigating whether a relationship exists between the model scale and performance on the provided datasets.

We first briefly describe generative language models that we fine-tune for the abstractive approach in Sec. 5.1. In Sec. 5.2, we explain our approach for *pseudo-labeling* the datasets for extractive summarization, which is followed by a description of the supervised extractive method in Sec. 5.3. We conclude by discussing how we implemented the methods, the data processing, and the experiments in Sec. 5.4.

5.1 Models for abstractive summarization

To find out how well the defined task can be solved using the abstractive summarization approach, we compare several generative sequence-to-sequence language models.

To experiment with the mBART model, we use the large version and fine-tune the pretrained 'facebook/mbart-large-cc25'[1] checkpoint. The model was pretrained with a goal to be later fine-tuned for translation task by appending the input sequence with a token indicating the source language (e.g., <`En`> for English) and prepending the target sequence (labels) with the target language token (e.g., <`De`>) for translating to German. To fine-tune it for a summarization task that transforms the source into the summary without changing the

[1] https://huggingface.co/facebook/mbart-large-cc25

language, we use the same language token to append the source and prepend the summary with. The resulting source-summary pair is `Long medical text<Cs>` / `<Cs>Concise summary`, as both the source and summary are in Czech (with `<Cs>` being the language token for Czech language).

To experiment with the mT5 model, we use three different sizes of the architecture, t5-small[2], t5-base[3] and t5-large[4]. The pretrained model checkpoint (not fine-tuned on any downstream tasks) is used for each model size. The mT5 model does not require any further preprocessing to be done on our dataset (apart from tokenizing it).

To include a very large language model in our experiments, we fine-tune AYA from the provided checkpoint 'CohereForAI/aya-101' [5] of the pretrained and Instruction fine-tuned model. As discussed in Sec. 2.2.6, fine-tuning the model on IFT dataset should increase the model performance on general instruction following tasks. This motivated us to try to preprocess the samples of our fine-tuning datasets by wrapping the source text with instruction, as illustrated in Fig. 5.1. We compared an AYA model fine-tuned on the instruction-wrapped

> Jsi lékař a píšeš propouštěcí zprávu po hospitalizaci pacienta. Tohle jsou informace o hospitalizaci: `SOURCE TEXT` Napiš odstavec "Průběh hospitalizace" na základě těchto informací. Používej jenom uvedené informace.

Figure 5.1: AYA model input wrapped with instruction.

source as well as on the source without the instruction and concluded it does not change the performance of the model. Due to the size of the AYA model, we had to use QLoRA (see Sec. 2.4.2) to be able to train the model due to large GPU memory requirements.

Finally, we compare the number of parameters of the language models we experiment with in Table 5.1.

Model	#Parameters
mBart	0.7 B
mt5-small	0.3 B
mt5-base	0.6 B
mt5-large	1.2 B
AYA	12.9 B

Table 5.1: Number of parameters of the language models used for the abstractive summarization approach.

5.2 Pseudo-labeling

Extractive summarization methods choose sentences (or subsequences of the source text) to be included in an extractive summary. The task is usually defined as a supervised one, and

[2] https://huggingface.co/google-t5/t5-small
[3] https://huggingface.co/google-t5/t5-base
[4] https://huggingface.co/google-t5/t5-large
[5] https://huggingface.co/CohereForAI/aya-101

a labeled dataset is needed for the training. The labels should be binary, signifying which sentences of the source text to extract. Manual annotation would be impractical and time-consuming in the case of our datasets. To enable experimenting with extractive summarization without human-annotated data, we experiment with several *pseduo-labeling* algorithms to create the labeling automatically, utilizing the target summary, which is available. This is quite a common approach, for example, used in [28]. We use similar methods but experiment with more similarity metrics. Pseudo-labeling was also used in [5], but it ¡sonly explored the use of the pairwise approach (described below).

The pseudo-labeling methods we experiment with are either pairwise or greedy. The **pairwise** approach first splits the source and summary into sentences and notes, respectively. We split the source text into sentences using SentenceSplitter[6], a text-to-sentences splitter based on a heuristic algorithm by Philipp Koehn and Josh Schroeder [42]. To split the summary text into *notes*, we first split it into sentences using the same approach but further split each sentence into sub-sentences (which we call *notes*) when we encounter the comma character. For each summary note, pairwise pseudo-labeling selects a source sentence most similar according to the used similarity metric. Multiple summary notes can be paired with the same source sentence. We parse the resulting selected source sentences as a set, ensuring there are no duplicates. During the dataset analysis, we noticed that the sentences in summaries are usually more dense in information, and assigning only one sentence from the source text to a single summary sentence would lead to information loss. This motivated us to split the summary into notes instead, as described above.

The **greedy** approach iteratively selects sentences of the source until an ending condition is met or the maximal number of sentences is selected. We set the limit to ten sentences in all our experiments, referring to Table 4.3 and Table 4.5. At each iteration, the greedy approach considers each unselected source sentence individually as a *candidate sentence* to be selected. The candidate sentence is concatenated with all previously selected source sentences (using the respective order of the sentences in the source), and this candidate extractive summary is compared with the target human-written summary using a similarity metric. At the end of each iteration, we select the candidate sentence that increases the similarity the most. Note that adding a new candidate sentence to the extractive summary can also decrease the similarity. If no candidate sentence increasing the similarity is found, or the maximum number of sentences in the extractive summary is reached, the iteration stops.

We use three similarity metrics jointly with the two approaches described above:

- **Rouge** - The metric is described in Sec. 2.5.1. We use a sum of the Rouge-1 and Rouge-2 F-scores as the similarity metric.
- **BertSCORE** - We use the F-score of the BertSCORE from Sec. 2.5.2
- **SentenceTransformer** - To compare the similarity of two sentences using the SentenceTransformer (Sec. 2.2.7), we embed both sequences using the model and use the cosine similarity of the embeddings as the metric.

Illustrative results of both pairwise and greedy pseudo-labeling methods can be found in Appendix A.

[6]https://github.com/mediacloud/sentence-splitter

5.3 Model for extractive summarization

For the extractive summarization method, we use a similar approach to [28], where they use an encoder-only BERT [26] model to embed the input sequence. As discussed before, BERT is a model trained on purely English corpus, not suitable for the Czech or another non-Enlish language. While a multilingual version, mBERT, exists, the XLM-RoBERTa, the multilingual version of the more robustly optimized RoBERTa, achieved better results on various downstream tasks. For this reason, we decided to use the XLM-RoBERTa encoder model (see Sec. 2.2.3 for detail) as a base of our extractive summarization method, as was also done in [5]. Concretely, we use the large version and load the pretrained checkpoint 'xlm-roberta-large'[7].

The method first processes the tokenized source using the encoder, outputting an embedding (a vector) for each token. Embeddings of tokens representing sentences of the source sequence are then gathered and processed using a linear sigmoid classifier, outputting binary prediction.

Before passing the source input sequence to the encoder, we need to prepare the input to be able to obtain a representation of the individual sentences. We preprocess the source by wrapping each sentence with a BOS (beginning of a sequence) token $\langle s \rangle$ and EOS (end of sequence) token $\langle /s \rangle$. When the tokens of the input sequence are embedded using the encoder, we take all the embeddings of the BOS tokens and use those as the representations of the sentences. The process is visualized in Fig. 5.2. All the $E_{\langle s \rangle}$ embeddings are gathered and

Source

This is sentence 1. This is sentence 2.

Preprocessed source

$\langle s \rangle$This is sentence 1.$\langle /s \rangle\langle s \rangle$ This is sentence 2.$\langle /s \rangle$

Tokens

$T_{\langle s \rangle}, T_{This}, T_{is}, T_{sentence}, T_1, T_{\langle /s \rangle}, T_{\langle s \rangle}, T_{This}, T_{is}, T_{sentence}, T_2, T_{\langle /s \rangle}$

Tokens embeddings

$E_{\langle s \rangle}, E_{This}, E_{is}, E_{sentence}, E_1, E_{\langle /s \rangle}, E_{\langle s \rangle}, E_{This}, E_{is}, E_{sentence}, E_2, E_{\langle /s \rangle}$

Figure 5.2: Visualization of the input sequence processing by the extractive method.

processed with the linear classifier:

$$\hat{y} = \sigma(WE_{\langle s \rangle} + b), \tag{5.1}$$

where σ is a sigmoid function and W, b are learnable parameters. The loss is computed as a binary cross entropy against the target pseudo-label y.

[7]https://huggingface.co/FacebookAI/xlm-roberta-large

During inference, both [28] and [5] score the input sequences using the linear classifier. They then choose the top k sentences with the highest score. This is unsuitable for our datasets, as choosing the right k is problematic. While the final summaries usually have 4-6 sentences, the crucial information might be scattered throughout several more or fewer source sentences, making this inference setting impractical. Instead, we used the final linear layer as the classifier it was trained to be. During inference, we compute the score for each source sentence. If it is higher than 0.5, we include it in the extractive summary.

The XLM-RoBERTa encoder model has a limited input sequence length of 512 tokens. This limitation is caused by the fixed size of the positional embedding layer. As our datasets with short samples contain samples with up to 1024 tokens, we need to modify the encoder to be able to process them. To allow for longer inputs, we concatenate two copies of the original learned positional embedding layer, creating new positional embedding that allows the encoder to process sequences of double the length. As the source text in our datasets is structured into individual paragraphs, we believe the positional embeddings over large distances are not too important. Only the local positional relationships need to be encoded correctly. Also, this new positional embedding layer is further trained during the fine-tuning process, which should help it accommodate to the changed setting.

We researched other multilingual encoder-only models that could be used in similar settings to train an extractive summarization method. As the research is now mainly focused on training large decoder-only models, which do not easily allow to get embeddings of the input tokens, the XLM-RoBERTa remains the best-performing option. Since doubling the positional embedding layer, an undocumented and not researched modification, is not an ideal solution to the limited input sequence length, we see this limitation as one of the drawbacks in pursuing extractive summarization for the automatic discharge report generation task. While we can experiment with it on the filtered datasets we created, we ought to use more source data to generate the hosptilazition summary to solve the task more thoroughly, which means longer input sentences, rendering the extractive approach unpractical.

5.4 Implementation

We use the Transformers library[8] to implement all our experiments. It provides implementations of all the SOTA language models, together with distributing their pretrained parameters. This allows us to experiment with multiple models using the same data loading and training procedures. The library is implemented in Python and allows several machine learning libraries (PyTorch, TensorFlow, and JAX) to be used interchangeably as the 'back-end'. We use the PyTorch [9] version.

Data preprocessing was mainly implemented using Python and its built-in libraries (regex, string processing). The Datasets[10] library was used to work with the preprocessed data. It also allows efficient storage of tokenized versions of the datasets, which speeds up the training process by removing the need to tokenize the text sequences during each iteration.

To compute the SentenceTransformer embeddings and BERTScore and AlignScore metrics, we used the respective implementations[11] [12] [13]. We adapted the implementation of the

[8]https://github.com/huggingface/transformers
[9]https://pytorch.org/
[10]https://github.com/huggingface/datasets
[11]https://github.com/UKPLab/sentence-transformers/tree/master
[12]https://github.com/Tiiiger/bert_score
[13]https://github.com/yuh-zha/AlignScore

Rouge-raw score computation from the code provided with the SumeCzech dataset[14].

All experiments were tracked using Weights & Biases [15] service, allowing us to monitor the training and compare various runs during hyperparameter tuning.

As training the models is a resource-heavy process, we utilized the RCI[16] cluster, to which we were provided access by the supervisor. We used the computational nodes with NVIDIA A100 GPUs, providing 40GB of GPU memory each. To allow training large models such as AYA, we used QLoRA implemented by its authors [3] in following the GitHub repository[17], accessible trough HuggingFace's PEFT library[18].

[14]https://ufal.mff.cuni.cz/sumeczech
[15]https://wandb.ai
[16]https://rci.cvut.cz/
[17]https://github.com/artidoro/qlora
[18]https://github.com/huggingface/peft

Chapter 6

Results

In this chapter, we describe various experiments we conducted to evaluate the proposed methods. In Sec. 6.1, we first compare the pseudo-labeling approaches. Then, we train the extractive summarization method on both the filtered datasets and evaluate it. To compare the abstractive summmarization methods, we conduct several experiments in Sec. 6.2. First, we fine-tune all the abstractive summarization models on the filtered datasets. To see how the performance changes if we use the complete dataset, we fine-tune a subset of the methods that can process longer input sequences. We compare all the fine-tuned models using automatic evaluation metrics from Sec. 2.5. Finally, in Sec. 6.3, we conduct a manual evaluation to see how reliably the automatic metrics can asses the model performance on the presented task.

In Appendix B, we also show examples of the summaries generated by an abstractive model concerning the manual evaluation.

6.1 Extractive summarization

In this section, we experiment with our extractive summarization method. We first evaluate the pseudo-labeling approaches proposed in Sec. 5.2. Then, we experiment with the extractive method itself, as introduced in Sec. 5.3, trained using annotations from the selected pseudo-labeling approach.

For the extractive model, we first evaluate the classifier in terms of Precision, Recall and F-score. The classifier outputs a binary label for each source text sentence, indicating whether or not the sentence should be extracted as part of the extractive summary. We compare this output to the ground truth labels from the corresponding pseudo-labeling algorithm and compute:

$$\text{Precision} = \frac{\#\text{Correctly extracted sentences}}{\#\text{Sentences extracted by model}}, \tag{6.1}$$

$$\text{Recall} = \frac{\#\text{Correctly extracted sentences}}{\#\text{Sentences extracted by pseudo-labeling}}, \tag{6.2}$$

$$F\text{-score} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6.3}$$

As with the text similarity metrics, we multiply the classifier metrics by 100 for better readability.

Next, we concatenate the extracted sentences to create an extractive summary, which we compare to the corresponding target summary and compute the text similarity metrics

Rouge and BERTScore. We report those for both the model and the used pseudo-labeling approach, denoted as Model and Oracle in the table, respectively. The oracle's performance also serves as an upper limit on the performance of the model.

We first tested several hyperparameter settings for all the extractive models we experimented with and evaluated them on the validation dataset split. We chose the model that achieved the highest recall. Generally, the model that achieved better recall also achieved better precision (and F-score in turn) than a worse model, up to negligible difference. Also, we argue recall is the better metric to optimize, as it is intuitively better to include more of the relevant details in the extractive summary, even if we include some sentences that might not be relevant. We used the held-out test dataset split to evaluate the best model and compute the metrics we then report.

6.1.1 Pseudo-labeling

To evaluate the pseudo-labeling approaches utilizing different similarity metrics, we create an extractive summary by extracting the pseudo-labeled sentences. We then compare these extractive summaries with the corresponding human-written summaries. We report the results in Table 6.1 (Acute Cardiology dataset) and Table 6.2 (Hepatogastroenterology dataset). The results are computed using the validation dataset split.

Method		Rouge-1	Rouge-2	Rouge-L	BERTScore
	Rouge	47.55	33.10	40.16	77.89
Pairwise	BERTScore	**48.24**	**34.98**	**41.93**	**78.91**
	SentenceTransformer	45.28	31.86	38.90	77.63
	Rouge	36.61	27.99	32.64	57.29
Greedy	BERTScore	**51.46**	**41.12**	**47.62**	**81.25**
	SentenceTransformer	45.79	31.94	38.94	77.47

Table 6.1: Comparison of extractive summary created by pseudo-labeling with human-written summaries on the Acute Cardiology dataset (validation split).

Method		Rouge-1	Rouge-2	Rouge-L	BERTScore
	Rouge	**39.99**	23.99	33.04	75.62
Pairwise	BERTScore	39.49	**24.34**	**33.36**	**76.51**
	SentenceTransformer	37.93	22.76	31.65	75.61
	Rouge	31.10	20.00	26.28	55.90
Greedy	BERTScore	**41.93**	**28.08**	**37.18**	**78.74**
	SentenceTransformer	37.95	22.72	31.30	75.22

Table 6.2: Comparison of extractive summary created by pseudo-labeling with human-written summaries on the Hepatogastroenterology dataset (validation split).

Generally, we can see that the Acute Cardiology dataset seems to be more extractive than the Hepatogastroenterology dataset. We believe this might be caused by the differences in fluency and amount of stylistic rewriting between the datasets, as described in Sec. 4.4.

Considering the automatic evaluation metrics, BERTScore seems to outperform the other two similarity metrics. However, for the pairwise approach, the performance of the different similarity metrics does not differ too significantly.

We also manually compared the results of the pseudo-labeling approaches side by side on multiple samples. We concluded that while the greedy approach using BERTScore performs best on the automatic evaluation, it tends not to include sentences that contain some crucial information for generating the summary. Such sentences, while containing an important atomic piece of information, might contain a lot of other subsequences with different atomic information that are not included in the summary. The similarity metric then tends not to select it. To illustrate this, let us consider Fig. 6.1. While the sentence from the source text

Sentence from the source text

Není volná tekutina v pekrikardu, jen lem tuku kolem PKS z parasternálních a hrotových projekcí, max. 3-5 mm.

Corresponding note in Hospitalization summary

Vyloučen perikard. výpotek.

Figure 6.1: Illustration of a problem with the greedy pseudo-labeling approach.

contains information (`Není volná tekutina v pekrikardu`) that is important to generate the corresponding summary note, it also contains other information, which causes the similarity during greedy sentence selection to be too low to extract this sentence. A naive solution might be to also split the source sentences into notes, as we do with summaries. That did, however, discard too much of the context information and achieved sub-optimal results during our initial exploration of these methods.

While the pairwise methods improve this, they create extractive summaries that include redundant information. Both discussed problems are inherent to the nature of the extractive summarization approach.

Based on the results in Table 6.1 and Table 6.2, we decided to use the pseudo-labels from the pairwise and greedy approach utilizing BERTScore as the similarity metric in the following experiments.

6.1.2 Acute Cardiology dataset

We trained two models on the filtered version of the Acute Cardiology dataset. One was fine-tuned on labels generated by the pairwise pseudo-labeling approach, and the other on the labels generated by the greedy pseudo-labeling approach.

Both models were trained using the Adam optimizer [36]. We used 500 warmup steps with a maximum learning rate of 3e-5 and a linear decay schedule afterward. A 0.2 dropout was applied to the input of the linear classifier layer during training. The model trained on the pairwise pseudo-labels was optimized using batch size 8, whereas the model trained on the greedy pseudo-labels used batch size 16. We trained both models for 10 epochs, evaluating every 200 epochs and saving the best checkpoint according to the validation loss.

We report the classifier performance in Table 6.3 and the resulting extractive summary evaluation in Table 6.4. According to both Rouge and BERTScore, both models perform similarly. As expected, they do not exceed the oracles' performance. By evaluating the classifier, however, we see the model trained using pairwise pseudo-labels performs better according to the F-score. The recall of the model trained on the greedy pseudo-labels is lacking compared to the model trained with pseudo-labels from the pairwise approach. This indicates it usually selects only a few easiest sentences, probably the ones describing the reason for the patient's admission, which are almost always present in the hospitalization summary paragraph.

Model	Precision	Recall	F-score
Pairwise	66.76	**57.28**	**62.08**
Greedy	**73.06**	36.46	48.64

Table 6.3: Evaluation of the extractive methods on the filtered Acute Cardiology dataset. We report the classifier metrics, precision, recall, and F-score, all computed on the test dataset split.

Pseudo-labeling	Method	Rouge-1	Rouge-2	Rouge-L	BERTScore
Pairwise	Oracle	48.24	34.98	41.93	78.91
	Model	**44.31**	32.27	39.50	**77.27**
Greedy	Oracle	51.46	41.12	47.62	81.25
	Model	42.85	**35.10**	**41.01**	75.83

Table 6.4: Evaluation of the extractive methods on the filtered Acute Cardiology dataset. We report the text similarity metrics, Rouge and BERTSCore, computed on the test dataset split. The metrics are computed for the oracle (the summary created by using the labels from pseudo-labeling directly) and the summary generated by the trained model.

6.1.3 Hepatogastroenterology dataset

Again, we trained two extractive models on the Hepatogastroenterology datasets using only the filtered version with short samples. Both models were trained with the same hyperparameter settings, using the Adam optimizer with a maximum learning rate of 3e-5 and batch size 4. The warmup was set to 500 steps with linear learning rate decay afterward. The models were trained for 10 epochs, stopping each 200 optimization steps to evaluate and save the best checkpoint according to the loss on the validation dataset split.

Results of the extractive methods trained on the Hepatogastroenterology dataset are reported in Table 6.5 and Table 6.6.

Model	Precision	Recall	F-score
Pairwise BertSCORE	**61.42**	**48.05**	**53.92**
Greedy BertSCORE	60.15	44.03	50.84

Table 6.5: Evaluation of the extractive methods on the filtered Hepatogastroenterology dataset. We report the classifier metrics, precision, recall, and F-score, all computed on the test dataset split.

Pseudo-labeling	Method	Rouge-1	Rouge-2	Rouge-L	BERTScore
Pairwise	Oracle	39.49	24.34	33.36	76.51
	Model	**33.31**	20.03	28.74	**73.73**
Greedy	Oracle	41.93	28.08	37.18	78.74
	Model	32.06	**20.08**	**28.91**	72.03

Table 6.6: Evaluation of the extractive methods on the filtered Hepatogastroenterology dataset. We report the text similarity metrics, Rouge and BERTSCore, computed on the test dataset split. The metrics are computed for the oracle (the summary created by using the labels from pseudo-labeling directly) and the summary generated by the trained model.

Considering the text similarity metrics, the situation is similar to the results on the Acute Cardiology dataset. Both models perform similarly, not exceeding the oracles' performances. The models perform worse on the Hepatogastroenterology dataset compared to the performance on the Acute Cardiology dataset. Interestingly, evaluating the classifier, it is consistently better according to all the metrics when using the labels generated by the pairwise pseudo-labeling approach. The performance gap between the pseudo-labeling approaches used is, however, quite narrow.

6.1.4 Conclusion

While the models seem to predict part of the extractive summary correctly, the results (especially compared to abstractive approaches that follow) are not very promising. The training and evaluation are further complicated due to the imperfect process of automatic label generation through the proposed pseudo-labeling approaches. Correctly splitting the source and summary into well-defined subsequences is also problematic, and the text similarity metrics are not fine-tuned to the medical-domain data we use.

Due to the unpromising results and the discussed complications with the extractive method and pseudo-labeling, we decided, upon discussion with the supervisor, not to explore the extractive approach further in favor of the abstractive summarization.

6.2 Abstractive summarization

In this section, we evaluate and compare the abstractive summarization methods described in Sec. 5.1. First, we fine-tune the models using the filtered versions of the datasets. This makes it possible to compare all the methods proposed in this thesis, together with the extractive ones as well. Next, we fine-tune two sizes of the mT5 model on the complete versions of the datasets to explore how the performance changes.

For all models, we first experimented with several hyperparameter settings. For each model, we chose the setting with which the model performed best on the validation dataset split. We measured the performance using the automatic text similarity metrics (Rouge and BERTScore). The ranking of the models was consistent across all the metrics. It did therefore not happen, that one model was better than other considering one metric, but worse considering another metric. This consistency is also apparent in the results reported in the tables below.

Apart from choosing optimal hyperparameters for the training, abstractive summarization (as a language generation task) performance is also influenced by the inference settings (see section Sec. 2.3). We experimented with several settings for the inference. Using beam search was shown to have the highest impact on the performance of the trained models. On the contrary, using sampling (without beam search) decreased the performance. When combining the beam search with sampling, the performance was also lower compared to using beam search only. By modifying the temperature, top-k sampling, or top-p sampling so that the sampling probability was sharper (originally, the most probable output tokens were even more likely to be sampled), we were able to match the performance of the beam search alone. We argue this is expected due to the nature of the data. The structure of the hospitalization summary is predictable and similar for each hospitalization. Based on these findings, we use beam search with five beams in all our experiments below without sampling, always choosing the most probable output token.

During the training of every model, we frequently evaluate it on the validation dataset split, saving the best-performing checkpoint according to the validation loss.

6.2.1 Filtered dataset

In this section, we report the performance of the abstractive summarization methods on the filtered versions of both the Acute Cardiology and Hepatogastroenterology datasets.

We used the following hyperparameter settings for the models trained on the Acute Cardiology dataset. All the models were optimized using the Adam optimizer [36] and trained for 20 epochs (apart from AYA). The learning rate decayed linearly after warmup steps (if there were any). The mBART model was trained with learning rate 3e-5 with batch size 4 and 2000 steps of warmup. Both mT5-small and mT5-base were trained using learning rate 1e-3 (as recommended by the paper that introduced it) and no warmup. The mT5-small was optimized using batches of 16 samples, while the mT5-base used batches of 64 samples. To train the mT5-large model, we used a lower learning rate 3e-4 and optimized the model using batches of 4 samples, with no warmup. Finally, the AYA model was trained for 10 epochs only (as it converged early and took over 10 hours to train), with a learning rate 1e-3, batches of 4 samples, no warmup. We used LoRA rank 4 and dropout 0.05 during the QLoRA fine-tuning.

We report the results of the abstractive summarization methods achieved on the filtered Acute Cardiology dataset in Table 6.7. The performance was measured on the held-out test dataset split, which was used neither during training nor hyperparameters tuning.

The mT5-base model performed best according to all the automatic similarity metrics. Considering BERTScore, the margin is relatively small. We also note that mT5-base performs best according to the factuality metric AlignScore-CS. Overall, all the models perform similarly, apart from mBART, whose performance is noticeably worse. We also note that all the abstractive models score higher than the extractive models on the corresponding dataset (see also Table 6.4). The results show, according to all the mT5 models, that the performance is

Model	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore-CS
extractive	44.31	32.27	39.50	77.27	—
mBART	49.63	36.95	46.31	79.76	65.03
mT5-small	53.21	40.90	50.24	81.20	66.47
mT5-base	**54.13**	**41.95**	**51.10**	**81.48**	**68.50**
mT5-large	52.72	40.36	49.67	80.87	67.61
AYA	53.23	40.62	49.86	81.16	67.47

Table 6.7: Evaluation of abstractive models trained on the filtered Acute Cardiology dataset. We report text similarity metrics Rouge and BERTScore and factuality metric AlignScore-CS. The models were evaluated on the filtered test dataset split. We also provide the results of the extractive method trained on the pairwise pseudo-labels for reference.

not influenced by the scale of the model. The AYA model also performed similarly, showing no effect of increased performance by using the Instruction Fine-Tuning.

For the models fine-tuned on the Hepatogastroenterology dataset, we used the following hyperparameter settings. All the models were trained for 20 epochs again (apart from AYA). The models were optimized using the Adam optimizer with a linearly decayed learning rate after an optional warmup. The mBART model was trained with the same setting as on the Acute Cardiology dataset, using learning rate 3e-5, batch size 4, and warmup of 2000 steps. Both mT5-small and mT5-base models were trained with learning rate 3e-4, batch size 8, and no warmup. The mT5-large mode was trained using learning rate 3e-5, batch size 8 and 1000 steps of warmup. To fine-tune the AYA model, we used learning rate 1e-3 with batch size 4. We trained the model for 10 epochs with no warmup steps, using the same LoRA setting during QLoRA fine-tuning as before.

The performance of models fine-tuned on the filtered Hepatogastroenterology dataset is reported in Table 6.8. We report the performance achieved on the held-out test dataset split.

Model	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore-CS
extractive	33.31	20.03	28.74	73.73	—
mBART	38.67	22.31	34.38	76.63	53.33
mT5-small	44.78	28.78	40.96	79.06	55.05
mT5-base	44.40	28.39	40.66	78.90	57.92
mT5-large	**44.96**	**29.04**	**41.31**	**79.26**	**60.88**
AYA	43.95	27.64	39.64	78.82	59.09

Table 6.8: Evaluation of abstractive models trained on the filtered Hepatogastroenterology dataset. We report text similarity metrics Rouge and BERTScore and factuality metric AlignScore-CS. The models were evaluated on the filtered test dataset split. We also provide the results of the extractive method trained on the pairwise pseudo-labels for reference.

The relative performance is similar to the performance of the models on the Acute Cardiology dataset, apart from the models achieving generally worse results. The mBART model is again outperformed by all the other models. The mT5 family of models seems to perform best on the data, but it is hard to decide which size is the best. The mT5-large model achieves the best text similarity results with a small margin (mT5-small being close behind).

It is also the best model according to the factuality metric AlignScore-CS. As before, all the abstractive models perform better than the extractive ones (see also Table 6.6).

Conclusion

We fine-tuned and compared all the abstractive models on the filtered versions of the datasets. The abstractive methods performed better than the extractive ones. On both datasets, mBART was outperformed by the other models. The performance of the mT5 models and the AYA model are very similar. We believe this might be caused by imperfections of the dataset, mainly the occurrence of the *unsourced* sentences discussed in Sec. 4.3. Our hypothesis is the model learns to generate the summary by extracting the information from the source text, that is usually present in the target summary. The model is, however, not able to generate a better summary that would be more similar to the target summary, as the needed information is not present in the source. This creates a performance boundary that the model cannot exceed even if we use larger models that tend to perform better than smaller ones on most benchmarks. We also note that the filtered datasets contain quite small amounts of samples, which might also make it hard for the model to learn to properly generalize. We explore the performance of the trained models in more detail during the manual evaluation in Sec. 6.3.

6.2.2 Complete dataset

We discussed in Sec. 4.3 that by filtering out the samples with long source input sequences, we are approximately filtering out more complicated hospitalizations. We hypothesized that it should therefore be easier (the models should perform better) to solve the task of automatically generating the hospitalization summary. To experiment with this, we trained abstractive summarization models on the complete datasets containing all the samples. While the complete datasets contain some more complicated cases, the overall dataset size (in terms of samples) is approximately double the size, which might also affect the performance, as using more training samples traditionally correlates with better performance.

As the complete datasets contain samples with longer inputs (up to 3000 mT5-base tokens), we were only able to fine-tune mT5-small and mT5-base models. The mT5-large and AYA required too much memory, causing the training process to fail. The mBART model is only able to process input sequences shorter than 1024 tokens and could not be used for the complete dataset.

Both models trained on the Acute Cardiology dataset were optimized using the Adam optimizer with learning rate 1e-3 with linear decay and no warmup. The mT5-small model was trained for 20 epochs with batch size 16. The mT5-base model was trained only for 10 epochs with batch size 32.

We report the models' performance on the complete Acute Cardiology dataset in Table 6.9. Both models achieve similar performance. According to the Rouge metrics, the models perform worse compared to when they were trained and evaluated on the filtered version of the dataset (see Table 6.7). This confirms that the longer (possibly more complicated) samples make it harder for the model to solve the task. On the other hand, the model-based metrics, BERTScore and AlignScore-CS do not indicate a strong performance decrease.

We further evaluate the models trained on the complete version of the Acute Cardiology dataset using the filtered version of the dataset in Table 6.10. The results show a performance

Model	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore-CS
mT5-small	51.23	38.90	47.74	80.41	68.83
mT5-base	51.49	39.12	47.84	80.52	67.90

Table 6.9: Evaluation of abstractive models trained on the complete Acute Cardiology dataset. The models were evaluated on the **complete** test dataset split. We report text similarity metrics Rouge and BERTScore and factuality metric AlignScore-CS.

Training data	Model	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore-CS
Complete	mT5-small	55.25	43.05	52.28	81.93	**69.70**
	mT5-base	**55.52**	**43.43**	**52.55**	**81.95**	67.98
Filtered	mT5-small	53.21	40.90	50.24	81.20	66.47
	mT5-base	54.13	41.95	51.10	81.48	68.50

Table 6.10: Comparison of abstractive models trained on the filtered and complete Acute Cardiology dataset. The models were evaluated on the **filtered** test dataset split. We report text similarity metrics Rouge and BERTScore and factuality metric AlignScore-CS.

increase over the models trained solely on the filtered dataset (see Table 6.7). This supports our hypothesis that the dataset consisting of more samples could increase the models' performance.

On the complete Hepatogastroenterology dataset, we fine-tuned both models using the Adam optimizer for 20 epochs and linearly decayed learning rate without warmup steps. The mT5-small model was trained with learning rate 1e-3 and batches of 16 samples. The mT5-base model was trained using learning rate 3e-4 and batch size 32.

We evaluated the models on the complete test split of the full Hepatogastroenterology dataset, and we report the results in Table 6.11. The performance is significantly worse

Model	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore-CS
mT5-small	38.16	23.45	34.25	76.44	54.11
mT5-base	39.05	24.12	34.97	76.80	58.53

Table 6.11: Evaluation of abstractive models trained on the complete Hepatogastroenterology dataset. The models were evaluated on the **complete** test dataset split. We report text similarity metrics Rouge and BERTScore and factuality metric AlignScore-CS.

compared to the models trained and evaluated on the filtered version of the dataset (see Table 6.8), again confirming that the samples with longer source text are more complicated for the model.

We again evaluated the models trained on the complete Hepatogastroenterology dataset on its filtered version and report the results in Table 6.12. While the performance margin is lower, we see the same pattern as with the Acute Cardiology dataset. The model trained on the complete data outperforms the model trained on the filtered datasets when we compare them both on the filtered dataset test split. The mT5-large model trained on the filtered version of the datasets, however, still outperforms both the models trained on the complete dataset according to the factuality metric AlignScore-CS

Training data	Model	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore-CS
Complete	mT5-small	44.33	28.43	40.66	79.06	54.21
	mT5-base	**45.48**	**29.52**	**41.71**	**79.51**	58.18
Filtered	mT5-small	44.78	28.78	40.96	79.06	55.05
	mT5-base	44.40	28.39	40.66	78.90	57.92
	mT5-large	44.96	29.04	41.31	79.26	**60.88**

Table 6.12: Comparison of abstractive models trained on the filtered and complete Hepato-gastroenterology dataset. The models were evaluated on the **filtered** test dataset split. We report text similarity metrics Rouge and BERTScore and factuality metric AlignScore-CS.

Conclusion

On both datasets, we confirmed the performance decreased when using the complete dataset. By evaluating the models trained using the completed dataset also on the filtered versions, we found out it performs better compared to the models trained only on the filtered data. We came to the conclusion this is caused by more samples being present in the complete datasets, even though some of the samples might be more challenging.

6.3 Manual evaluation

Until now, we have relied on automatic text evaluation metrics. The metrics were shown to correlate with human judgment to some extent. However, our datasets are specific due to medical domain text, frequent use of abbreviations, and being in Czech. These factors might cause the automatic evaluations to be imprecise. To make a step toward a more thorough analysis of the model's performance, we evaluate some of the models by means of manual annotation of the predicted summaries. We are interested in two metrics we want to measure through the manual evaluation. First, we want to measure how factual and faithful the generated summaries are. And, second, we assess the general quality of the generated summaries related to how we defined the task.

Measuring *factuality* and *faithfulness* (we use these terms as defined in Sec. 2.1) is quite straightforward. We compare the generated summary with the source text and the true summary. We annotate the generated summary as follows:

- **Faithful**, if all the information contained in the generated summary can also be found in the source text. In other words, if the generated summary is factually consistent with the source text and there are no hallucinations.
- **Factual**, if all the information contained in the generated summary can be found in the source text or in the true summary. Using this label, we want to distinguish situations in which a model generates information that can not be found in the source text (it is unsourced) but is actually factual according to the true summary written by the doctor. The summarization models often tend to generate a sentence stating that a particular procedure or surgery was carried out without complication. While this information is not to be found in the source text as is, there is also the absence of any information about the procedure being problematic. In such cases, we believe the model is factual, even though we say it is hallucinating.
- **Unfactual**, if some information contained in the generated summary is not factually consistent with the source text or the true summary written by the doctor.

Measuring the overall *quality* of the generated summary is more complicated. A medical expert might be able to quickly decide whether the generated summary is sufficient or not. We have no means to gather medical experts' evaluations of the generated summaries at the moment, so we decided to manually evaluate the summaries ourselves. To make the manual evaluation more rigorous and as unbiased as possible, we define rules for how we evaluate the generated summaries. We need to incorporate the fact that our training data contains some unsourced sentences in the target summaries into our evaluation rules, making the evaluation more intricate. To evaluate a sample, we compare the generated summary, the true summary written by a doctor, and the source text. We choose one of the following labels:

- **Bad**, if the generated summary contains only the reason for admission or contains almost no information compared to the true summary. The reason for admission is either the first sentence of the Current problems or Hospitalization reasons paragraphs. The sentence is usually copied to the summary as it is, and it is therefore very easy for the model to predict this part of the hospitalization summary, and we see that as insufficient completion of the defined task.
- **Sufficient**, if the generated summary includes the correct hospitalization reason (according to the true summary). It also needs to include some of the other information contained in the true summary written by the doctor. It might not include all the information, either due to the model not performing well or some of the information being unsourced.
- **Good**, if the generated summary includes all the information contained in the true summary written by a doctor, which can be sourced in the source text. This means the generated summary might not contain the unsourced sentences from the true summary. We do this to evaluate how well the model can extract all the relevant information, irrespective of how sourced or not the sentences of the true summary actually are.
- **Perfect**, if the generated summary includes all the information in the true summary written by a doctor. The amount of such samples is influenced not only by the model performance but also by how often true summaries contain unsourced sentences.

We evaluated two models on the Acute Cardiology dataset. The first evaluated model is the best-performing model trained on the filtered version of the dataset, the mT5-base. We also evaluated the mT5-base model trained on the complete dataset. We chose to evaluate this model, as it performed better than the first model trained only on the filtered version (see Table 6.10) and to continue the comparison of the models from Sec. 6.2.2. Both models were evaluated on the test split of the filtered dataset.

To evaluate each model, we randomly chose 50 samples of the filtered Acute Cardiology dataset. We evaluated the generated summaries in terms of general quality and factuality, assigning the labels described above. The results are reported in Table 6.13 and Table 6.14. In both tables, we report how many of the 50 random samples were assigned various quality or factuality labels. We further evaluate samples of each label using the automatic text similarity and factuality metrics and report those in the tables as well.

We used the same process to manually evaluate the mT5-large model (best-performing) trained on the filtered Hepatogastroenterology dataset and we report the results in Table 6.15. We chose to evaluate only this model on the Hepatogastroenterology dataset, as the other models (even if trained on the complete version of the dataset) performed similarly.

We can infer various findings from the results. We first want to comment on the performance of the models according to the manually assigned labels. We claim that the performance of both models evaluated on the Acute Cardiology dataset is similar (see Table 6.13 and Ta-

Category	Count	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore
Perfect	22%(11)	72.28	63.16	72.28	88.42	61.56
Good	22%(11)	56.68	43.37	53.76	83.12	66.19
Sufficient	40%(20)	49.41	37.82	46.38	79.69	68.49
Bad	16%(8)	37.02	21.98	32.70	75.98	61.90
Faithful	64%(32)	55.70	43.95	52.78	82.36	69.03
Factual	20%(10)	53.61	38.62	50.37	81.20	53.50
Unfactual	16%(8)	49.83	38.89	47.85	80.15	65.78

Table 6.13: Manual evaluation of mT5-base abstractive summarization model trained on the **filtered** Acute Cardiology dataset. Randomly selected 50 samples from the **filtered** test dataset split were manually assigned labels based on the quality and factuality of the generated summary.

Category	Count	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore
Perfect	6%(3)	68.66	56.59	66.79	86.89	66.02
Good	38%(19)	58.43	46.19	56.10	82.80	63.03
Sufficient	40%(20)	58.43	45.16	54.73	83.33	70.99
Bad	16%(8)	46.17	35.11	42.88	77.01	77.79
Faithful	68%(34)	56.07	44.00	52.98	81.79	73.73
Factual	26%(13)	61.15	47.54	58.08	83.96	61.72
Unfactual	6%(3)	50.95	39.14	49.27	81.46	42.88

Table 6.14: Manual evaluation of mT5-base abstractive summarization model trained on the **complete** Acute Cardiology dataset. Randomly selected 50 samples from the **filtered** test dataset split were manually assigned labels based on the quality and factuality of the generated summary.

ble 6.14), according to the distribution of the quality labels. The only notable difference is the distribution of `Perfect` and `Good` samples. Both perfect and good generated summaries contained all the important information that could be inferable from the source text. This means that both labels say the model solved the defined task correctly. The perfect generated summaries are better, as they contain all the information the true summary does. Such cases are, however, limited to the samples where the source text contains all the needed information, which means the distribution of good and perfect generated summaries is to some extent more affected by the dataset properties and by only sampling 50 samples to evaluate, and not by the model performance. We note that only 16% of the evaluated samples were assigned the quality label `Bad`. Those were summaries that we evaluated to not contain any useful part of the hospitalization summary. The rest (84%) generated summaries did solve the task at least partially, according to the task definition and our evaluation scheme described above. According to the factuality labels, only 16% (and only 6% for the second model evaluated on the Acute Cardiology dataset) of the generated summaries contained unfactual information. Over 60% samples were hallucination-free (assigned the `Faithful` label), utilizing strictly only information contained in the source text.

The model trained and evaluated on the Hepatogastroenterology dataset performed worse according to the manually assigned labels (see Table 6.15). While the total amount of

Category	Count	Rouge-1	Rouge-2	Rouge-L	BERTScore	AlignScore
Perfect	8%(4)	70.44	52.64	67.51	88.84	52.92
Good	36%(18)	53.06	36.76	50.42	82.32	60.64
Sufficient	30%(15)	47.07	30.78	43.74	79.69	59.21
Bad	23%(13)	46.36	32.67	43.62	79.64	59.70
Faithful	46%(23)	54.77	30.83	52.86	82.51	59.94
Factual	36%(18)	52.19	33.46	48.75	82.50	60.67
Unfactual	18%(9)	38.47	24.14	34.15	76.10	55.20

Table 6.15: Manual evaluation of mT5-base abstractive summarization model trained on the **filtered** Hepatogastroenterology dataset. Randomly selected 50 samples from the **filtered** test dataset split were manually assigned labels based on the quality and factuality of the generated summary.

perfect and good generated summaries (44%) is the same as above when evaluating the Acute Cardiology models, the model generates more summaries that we evaluate as bad (not solving the defined task at all). Factuality-wise, the model hallucinates more, with only 46% of the generated summaries being `Faithful`. During the manual evaluation, we noticed the true summaries contained some unsourced information more often than in the Acute Cardiology dataset, which could explain why the model trained on the Hepatogastroenterology dataset hallucinates more. This observation is supported by the fact that while manually evaluating the generated summaries on the Hepatogastroenterology dataset, more samples were labeled as `Factual` (containing factual, although hallucinated information) or `Unfactual` than in the case of Acute Cardiology dataset. The worse performance of the model on the Hepatogastroenterology dataset is consistent with evaluation using the automatic text similarity and factuality metrics (see Sec. 6.2).

Using the manually evaluated samples, we also want to discuss how the assigned quality and factuality labels correlate with the corresponding automatic metrics. Considering both Rouge and BERTScore, the values are generally higher for samples assigned better quality labels (with `Perfect` being the best) across both datasets. There are some irregularities between `Good` and `Sufficient` labels in Table 6.14 and `Sufficient` and `Bad` labels in Table 6.15, which show the metrics are not perfect for evaluating the task. While not perfect, the automatic metrics prove to be a useful proxy for evaluating the models through this analysis. Our conclusion is that the automatic metrics can be used to evaluate our task, even though it deals with domain-specific Czech text data, but should be used together with manual evaluation to assess the performance more thoroughly, also considering the training data imperfections and limitations.

For the factuality label, we are mainly interested in the relationship with the values of AlignScore-CS. On the Acute Cardiology dataset, the `Faithful` samples achieve a higher AlignScore-CS score than `Factual` and `Unfactual` samples, which is expected, as those contain information not contained in the source text. However, focusing on Table 6.13, we see the difference between AlignScore-CS of `Faithful` and `Unfactual` samples is not large, but we would expect it to be for a useful factuality metric. Further, on the Hepatogastroenterology dataset, the `Faithful` samples score worse according to the AlignScore-CS than the `Factual` samples, even though those contain hallucinated content. These observations lead us to believe that the AlignScore-CS metric is not very useful for evaluating the facticity of generated summaries on our medical domain datasets. Our conclusion is that while the AlignScore-

CS achieves favorable results on Czech language general-domain benchmarks, using it on medical domain text would probably need some further domain-specific finetuning, as assessing factuality requires more involved text understanding than assessing text similarity. We also note that the `Unfactual` samples are generally evaluated worse in terms of automatic text similarity, showing that, expectedly, the factually correct generated summaries are evaluated as better in terms of quality by the automatic metrics.

We provide examples of the generated summaries in Appendix B. The example summaries were generated by the best-performing models trained on the filtered datasets. For each model, we show one model-generated summary labeled as `Good` and one model-generated summary labeled as `Sufficient`.

6.3.1 Further observations

During the manual evaluation, we noticed a few other properties of the datasets and the generated summaries we want to mention. While we understand the need for data anonymization, it can cause some problems. More specifically, we noticed that the hospitalization summaries often mention a medical examination that followed a specific number of days after some procedure. While the source text contains information about both the procedure and the examination, the dates are obfuscated, making it impossible for the model to predict the summary correctly.

The abstractive models tend to only copy various parts of the source text to create the summary, without paraphrasing it or leaving out anything in between. While this is expected, as these types of summaries often occur in the training data, there are samples where it is desirable for the model to paraphrase an examination conclusion or leave out some subsequence of a sentence. Samples where this is needed are not represented often enough in the training dataset, making it hard for the model to learn the right behavior, as copying parts of the input is the easiest way for it to optimize the loss function.

This is related to our next observation. The model is often too wordy, including an unnecessary amount of information when reporting admission reason, examination conclusion, or procedure.

6.4 Discussion

We experimented with solving the defined task of automatic hospitalization summary generation with both extractive and abstractive methods. The extractive methods, while predicting non-trivial summaries, did not achieve favorable results, especially compared to the abstractive models. We compared abstractive models trained on both filtered and complete versions of the datasets, confirming that the datasets with all samples are more complicated to solve but also that using more training samples (although some are more complicated) tends to increase the performance of the model when evaluated on the less complicated samples. We concluded by manually evaluating two models trained on the Acute Cardiology dataset and one model trained on the Hepatogastroenterology dataset. This allowed us not only to evaluate the individual models' performance but also to assess how well the automatic text similarity and factuality metrics correlate with the manual evaluation labels. We found out that the text similarity metrics Rouge and BERTScore can be used effectively as a proxy for the models' general performance. Comparing manual evaluation with the factuality metric AlignScore-CS showed that the metric is not able to assess the factuality of the generated summaries effectively. We hypothesize this is caused by the medical nature of the data we use, and an effective automatic factuality metric would have to be further fine-tuned for this domain.

Chapter 7

Conclusion

In this thesis, we dealt with the task of automatically generating the hospitalization summary paragraph contained in a discharge report, an important medical document summarizing a patient's hospital stay. Currently, the manual generation of discharge reports is a time-consuming task. Automating clerical work in the medical domain has the potential to save physicians valuable time that can be devoted to patient care, thus increasing the well-being of doctors and patients alike.

In Chapter 2, we defined the general NLP task of text summarization and described the abstractive and extractive approaches for solving it. We described the Transformer architecture, highlighting its role in the recent significant increase in language model capabilities. We reviewed various SOTA language models, ranging in different model sizes, architectures (encoder-only or sequence-to-sequence) and pretraining schemes. While high-quality pretrained language models are abundant today, we were restricted by the model's ability to understand and generate text in the Czech language. Also, not all models (their implementation and pretrained weights) are openly available, further limiting our choice. To facilitate the training of a large language model, we described a parameter-efficient approach to finetuning we use: QLoRA. We also discussed automatic text similarity and factuality metrics for the evaluation of the models on our task.

In Chapter 3, we briefly reviewed relevant works to further motivate the use of language models in the clinical domain. These related works address similar tasks to the goal of this thesis, but using English clinical datasets. Since our dataset is in Czech, the findings are not directly transferable but can serve as a valuable reference.

In Chapter 4, we introduced two Czech medical text datasets we obtained from IKEM. The datasets are from the Department of Acute Cardiology and the Department of Hepatogastroenterology. We precisely defined the task of generating the hospitalization summary paragraph using samples of the datasets and explained various preprocessing steps we undertook to prepare the data for the task. We followed by thoroughly analyzing the datasets and hypothesizing how the findings might affect the models' performance. We also identified imperfections of the dataset, that might hinder the possibility of solving the defined task.

We described in detail the methods we implemented and used in our experiments in Chapter 5. We explained our extractive summarization method together with various pseudo-labeling approaches needed for the supervised training. For the abstractive summarization approach, we described various pretrained multilingual sequence-to-sequence language models we fine-tuned on our datasets during experiments.

We conducted several experiments to test how well the proposed methods can solve the defined task in Chapter 6. We started by evaluating the pseudo-labeling approach needed for the supervised training of the extractive method. Using the best-performing pseudo-labeling approach based on the BERTScore metric, we fine-tuned the extractive method on both datasets and evaluated the models with automatic metrics. While the methods generated non-trivial summaries, the performance was not promising, compared to the abstractive summarization methods explored afterward. Training of the extractive method was further complicated due to difficulties with generating high-quality pseudo-labels and imperfections of the dataset. We decided not to explore the extractive approach further due to the complications and subpar performance, in favor of the abstractive summarization models.

To experiment with the abstractive approach, we fine-tuned several models. Initially, we fine-tuned the models on the filtered version of the datasets (containing only short samples), to be able to compare all our methods, as some have limited length of input they can process. The models showed higher performance on the Acute Cardiology dataset, as we hypothesized during the data analysis. On both datasets, mT5 models showed the best performance. We also fine-tuned the models on the complete versions of the datasets, where they performed worse. This confirmed the longer samples are more complicated to generate the hospitalization summary with, as we expected. Interestingly, when evaluated on the filtered datasets, the models fine-tuned on the complete versions of the datasets outperformed those fine-tuned on the filtered datasets, showing the benefit of using more samples to train language models.

We concluded the experiments by manually evaluating the best-performing models. The best-performing model fine-tuned on the Acute Cardiology dataset predicted the hospitalization summary correctly on 44% tested samples, with only 16% samples being evaluated as not solving the task properly at all. The model also showed a low tendency to generate unfactual information (6-16%). We also used the manually assigned quality and factuality labels to infer whether the automatic metrics are a good proxy for the real model performance on our task. We concluded that while the Rouge and BERTScore metrics are consistent with the manual evaluation of the generated summary quality, the AlignScore-CS factuality metric showed inconsistent results.

We believe this thesis serves as a good starting point for further research on applying language models for processing Czech clinical data. We showed that upon fine-tuning, the model can solve the defined task to some extent. We also identified problems with the dataset and common mistakes in the model-generated summaries. A natural next step for future research would be to incorporate more medical notes (for example all the daily notes) as a source to generate the hospitalization summary. Another approach to consider is to generate the whole discharge report using multiple medical documents concerning the hospitalization as a source. As shown in the related work, the models benefit from pretraining on medical text corpus before being fine-tuned for the specified task, this should also be explored but will be problematic due to the inaccessibility of freely available Czech clinical text corpus.

Chapter 8

References

[1] S. Singh, F. Vargus, D. Dsouza, *et al.*, "Aya Dataset: An Open-Access Collection for Multilingual Instruction Tuning," no. arXiv:2402.06619, Feb. 2024. DOI: 10.48550/arXiv.2402.06619. arXiv: 2402.06619 [cs].

[2] A. Üstün, V. Aryabumi, Z.-X. Yong, *et al.*, "Aya Model: An Instruction Finetuned Open-Access Multilingual Language Model," no. arXiv:2402.07827, Feb. 2024. DOI: 10.48550/arXiv.2402.07827. arXiv: 2402.07827 [cs].

[3] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," *Advances in Neural Information Processing Systems*, vol. 36, pp. 10 088–10 115, Dec. 2023.

[4] T. Dettmers and L. Zettlemoyer, "The case for 4-bit precision: K-bit Inference Scaling Laws," in *Proceedings of the 40th International Conference on Machine Learning*, PMLR, Jul. 2023, pp. 7750–7774.

[5] V. Halama, "Zkvalitňování fakticity generovaných sumarizací," Jun. 2023.

[6] Z. Ji, N. Lee, R. Frieske, *et al.*, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, 2023, ISSN: 0360-0300. DOI: 10.1145/3571730. [Online]. Available: https://doi.org/10.1145/3571730.

[7] P. Landes, A. Chaise, K. Patel, S. Huang, and B. Di Eugenio, "Hospital Discharge Summarization Data Provenance," in *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, D. Demner-fushman, S. Ananiadou, and K. Cohen, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 439–448. DOI: 10.18653/v1/2023.bionlp-1.41.

[8] S. Ramprasad, E. Ferracane, and S. P. Selvaraj, "Generating more faithful and consistent SOAP notes using attribute-specific parameters," in *Proceedings of the 8th Machine Learning for Healthcare Conference*, PMLR, Dec. 2023, pp. 631–649.

[9] T. Searle, Z. Ibrahim, J. Teo, and R. J. B. Dobson, "Discharge summary hospital course summarisation of in patient Electronic Health Record text with clinical concept guided deep pre-trained Transformer models," *Journal of Biomedical Informatics*, vol. 141, p. 104 358, May 2023, ISSN: 1532-0464. DOI: 10.1016/j.jbi.2023.104358.

[10] D. Van Veen, C. Van Uden, M. Attias, *et al.*, "RadAdapt: Radiology Report Summarization via Lightweight Domain Adaptation of Large Language Models," no. arXiv:2305.01146, Jul. 2023. DOI: 10.48550/arXiv.2305.01146. arXiv: 2305.01146 [cs].

[11] D. Van Veen, C. Van Uden, L. Blankemeier, *et al.*, "Clinical Text Summarization: Adapting Large Language Models Can Outperform Human Experts," no. arXiv:2309.07430, Oct. 2023. arXiv: 2309.07430 [cs].

[12] Y. Zha, Y. Yang, R. Li, and Z. Hu, “AlignScore: Evaluating Factual Consistency with a Unified Alignment Function,” no. arXiv:2305.16739, May 2023. DOI: 10.48550/arXiv.2305.16739. arXiv: 2305.16739 [cs].

[13] K. Ando, T. Okumura, M. Komachi, H. Horiguchi, and Y. Matsumoto, “Is artificial intelligence capable of generating hospital discharge summaries from inpatient records?” *PLOS Digital Health*, vol. 1, no. 12, e0000158, Dec. 2022, ISSN: 2767-3170. DOI: 10.1371/journal.pdig.0000158.

[14] G. Adams, E. Alsentzer, M. Ketenci, J. Zucker, and N. Elhadad, “What’s in a Summary? Laying the Groundwork for Advances in Hospital-Course Summarization,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, *et al.*, Eds., Online: Association for Computational Linguistics, Jun. 2021, pp. 4794–4811. DOI: 10.18653/v1/2021.naacl-main.382.

[15] V. Gupta, P. Bharti, P. Nokhiz, and H. Karnick, “SumPubMed: Summarization dataset of PubMed scientific articles,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, J. Kabbara, H. Lin, A. Paullada, and J. Vamvas, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 292–303. DOI: 10.18653/v1/2021.acl-srw.30. [Online]. Available: https://aclanthology.org/2021.acl-srw.30.

[16] E. J. Hu, Y. Shen, P. Wallis, *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models,” no. arXiv:2106.09685, Oct. 2021. DOI: 10.48550/arXiv.2106.09685. arXiv: 2106.09685 [cs].

[17] K. Krishna, S. Khosla, J. Bigham, and Z. C. Lipton, “Generating SOAP Notes from Doctor-Patient Conversations Using Modular Summarization Techniques,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 4958–4972. DOI: 10.18653/v1/2021.acl-long.384.

[18] L. Xue, N. Constant, A. Roberts, *et al.*, “mT5: A massively multilingual pre-trained text-to-text transformer,” no. arXiv:2010.11934, Mar. 2021. DOI: 10.48550/arXiv.2010.11934. arXiv: 2010.11934 [cs].

[19] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, “Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning,” no. arXiv:2012.13255, Dec. 2020. DOI: 10.48550/arXiv.2012.13255. arXiv: 2012.13255 [cs].

[20] T. Brown, B. Mann, N. Ryder, *et al.*, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901.

[21] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, “Unsupervised Cross-lingual Representation Learning at Scale,” no. arXiv:1911.02116, Apr. 2020. DOI: 10.48550/arXiv.1911.02116. arXiv: 1911.02116 [cs].

[22] Y. Liu, J. Gu, N. Goyal, *et al.*, “Multilingual Denoising Pre-training for Neural Machine Translation,” *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 726–742, Nov. 2020, ISSN: 2307-387X. DOI: 10.1162/tacl_a_00343.

[23] C. Raffel, N. Shazeer, A. Roberts, *et al.*, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020, ISSN: 1533-7928.

[24] N. Reimers and I. Gurevych, “Making monolingual sentence embeddings multilingual using knowledge distillation,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2020. [Online]. Available: https://arxiv.org/abs/2004.09813.

[25] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT,” no. arXiv:1904.09675, Feb. 2020. DOI: 10.48550/arXiv.1904.09675. arXiv: 1904.09675 [cs].

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” no. arXiv:1810.04805, May 2019. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805 [cs].

[27] M. Lewis, Y. Liu, N. Goyal, *et al.*, “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” no. arXiv:1910.13461, Oct. 2019. DOI: 10.48550/arXiv.1910.13461. arXiv: 1910.13461 [cs, stat].

[28] Y. Liu, “Fine-tune BERT for Extractive Summarization,” no. arXiv:1903.10318, Sep. 2019. DOI: 10.48550/arXiv.1903.10318. arXiv: 1903.10318 [cs].

[29] Y. Liu, M. Ott, N. Goyal, *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” no. arXiv:1907.11692, Jul. 2019. DOI: 10.48550/arXiv.1907.11692. arXiv: 1907.11692 [cs].

[30] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” no. arXiv:1908.10084, Aug. 2019. DOI: 10.48550/arXiv.1908.10084. arXiv: 1908.10084 [cs].

[31] R. Jackson, I. Kartoglu, C. Stringer, *et al.*, “Cogstack-experiences of deploying integrated information retrieval and extraction services in a large national health service foundation trust hospital,” *BMC medical informatics and decision making*, vol. 18, pp. 1–13, 2018.

[32] T. Kudo, “Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates,” no. arXiv:1804.10959, Apr. 2018. DOI: 10.48550/arXiv.1804.10959. arXiv: 1804.10959 [cs].

[33] T. Kudo and J. Richardson, “SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing,” no. arXiv:1808.06226, Aug. 2018. DOI: 10.48550/arXiv.1808.06226. arXiv: 1808.06226 [cs].

[34] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, “Measuring the Intrinsic Dimension of Objective Landscapes,” no. arXiv:1804.08838, Apr. 2018. DOI: 10.48550/arXiv.1804.08838. arXiv: 1804.08838 [cs, stat].

[35] M. Straka, N. Mediankin, T. Kocmi, Z. Žabokrtský, V. Hudeček, and J. Hajič, “SumeCzech: Large Czech news-based summarization dataset,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://www.aclweb.org/anthology/L18-1551.

[36] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” no. arXiv:1412.6980, Jan. 2017. DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs].

[37] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[38] A. E. Johnson, T. J. Pollard, L. Shen, *et al.*, “Mimic-iii, a freely accessible critical care database,” *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[39] R. Sennrich, B. Haddow, and A. Birch, “Neural Machine Translation of Rare Words with Subword Units,” no. arXiv:1508.07909, Jun. 2016. DOI: 10.48550/arXiv.1508.07909. arXiv: 1508.07909 [cs].

[40] J. Straková, M. Straka, and J. Hajič, “Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition,” in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 13–18. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

[41] M. Schuster and K. Nakajima, “Japanese and Korean voice search,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2012, pp. 5149–5152. DOI: 10.1109/ICASSP.2012.6289079.

[42] P. Koehn, “Europarl: A Parallel Corpus for Statistical Machine Translation,” in *Proceedings of Machine Translation Summit X: Papers*, Phuket, Thailand, Sep. 2005, pp. 79–86.

[43] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries,” in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

Appendix A

Illustrative datasets samples

Source

Nynější onemocnění:

Pacientka se známou achaláziií jícnu, po 2. etapech aplikaci butulinotoxinu, 2x po impakci sousta, dnes přichází pro 2 dní trvající dysfagii. Od včerejška cca 30 minut po jakémukoliv perorálnímu přijmu zvrací, netoleruje ani tekutiny, nebrála ani léky. Jinak bez obtíží, bez teplot, průjem neměla. Dýchá se dobře.

Objektivní nález:

Při vědomí, orient. plně, klidná, bez alter. celk. stavu, spoluprac., eupnoe, acyan, anikter, hydratace snížena, výživa v normě. Izokorie, skléry bílé, spoj. lehce bledší, jazyk plazí středem, suchší, bez patol. povlaku. Krk bez hrubé patologie, patol. uzliny ani šž nehmatám, náplň krčních žil nezvýšena. Jizva po sternotomii klidná, dýchání bilat. čisté, difúzně oslabené, sklípkové, akce srdeční pravidelná, ozvy 2, snad slabý systol. šel. na hrotě. Břicho lehce nad niveau, dýchá volně v celém rozsahu, palp. bolestivé v pravém dolním kvadrantu, kde hmatná i bolestivá oblá rezistence v průměru cca 6cm, v okolí défense, Blumberg +, Rovsing -, Plenies negat, břicho jinak ve zbylých kvadrantech měkké, volně prohm, palp. nebol, aperit, bez patol. rezistence, poklep dif. bub. Perist přítomna. DKK bez otoků, bez zn. TEN

Závěr při přijetí:

Recediva dysfagie.

Důvod hospitalizace:

Impakce sousta

Operace:

Vyšetření:

Gastroskopie DD

Po lokální přípravě a premedikaci i.v. zaveden přístroj GIF 190 Zaseknuté tuhé sousto nad kardií - provedena jeho postuopná extrakce ROTHE kličkou, pak kardií lze celkem volně do žaludku a norm. nál. do duodena. Diostální jícen iritovaný, zdeformovaný

Závěr: Extrakce uvízlého sousta nad kardií, jinak obraz achalazie s lehkou dilatací a atyp. peristaltikou, jinak norm. nál. do duodena Defromace a iritace term. jícnu

Hospitalization summary

XX-letá pacientka s achalázií jícnu, po 2. etapách aplikace botulotoxinu a opakovaně po impakci sousta, byla přijata na KH pro dysfagii. DD provedena gastroskopie s extrakcí uvízlého sousta nad kardií, jinak byl popsán obraz achalazie s lehkou dilatací.

Figure A.1: Sample from Hepatogastroenterology dataset with pseudo-labeled sentences. Sentences highligted in green are pseudo-labeled by the BERTScore greedy approach. Sentences with bold orange font are pseudo-labeeld by the BERTScore pairwise approach.

Source

Nynější onemocnění:

XX-letý polymorbidní pacient se susp. levostrannou ARVD/C, s recidivujícími KT, s implantovaným BiV ICD, stp. opak. RF ablacích endo i epikardiálních jak pro KT tak FiS, poslední pro perzistentní FiS DD, nyní hospitalizován k RFA pro další epizody KT.

Subj. námahová dušnost NYHA III - rychlejší chůze po rovině, chůze do kopce, do schodu po 1. patře, v klidu se nezadýchává, v nocí se produšnost nebudí; bolesti anginozního charakteru neguje; palpitace mejsou časté, spíše má častějí motání hlavy s pocity slabosti a nejistoty; nohy neotékají. **Bez dyspeptických a dysurických potíží.**

Objektivní nález:

Orientovaný, spolupracující, bez celk. alterace, eupnoe, dobře hydratován, bez cyanosy, bez ikteru. Hlava a krk s přiměř. nálezem, št. žl. nezvětš., uzliny 0, karotidy bez šelestu, šíje volná, jazyk vlhký, bez povlaku, plazí středem. **Normální náplň krčních žil.** Hrudník symetrický, klenutý, nadklíčky a axily volné. Dýchání čisté, sklípkové, bez vedlejších fenomenů. AS pravidelná, 2 ohran. ozvy, šelest nedif., břicho klidné, palp. nebol, bez rezist, H+L nezvětš., perist. +, DK bez otoků a zn. TEN, varixy klidné, pulzace nad AF bilat. +/+, bilat. bez šelestů, periferie teplá, fyziol. barvy.

Závěr při přijetí:

XX-letý polymorbidní pacient se susp. levostrannou ARVD/C, recidivující KT. s implantovaným BiV ICD, stp. opak. RF ablacích endo i epikardiálních jak pro KT tak FiS, poslední pro perzistentní FiS DD, hospitalizován k RFA pro další epizody KT.

Důvod hospitalizace:

Hospitalizace k reRFA KT

Operace:

Elektrofyziologie DD

Reablace pro KT u pacienta s DKMP - provedena biplární ablace mezikomorového basálního septa, dále ablace v kapsičkách ao chlopně

Vyšetření:

Echokardiografie KK DD

levá komora lehce dilatovaná (EDD 71 mm), s těžce omezenou ejekční frakcí (20-25%), porucha kinetiky: difúzní hypokiknéza, dyskinéza hrotu, relativně nejlepší kinetika lat. stěny, normální geometrie, normální tloušťka septa, bez trombu v dutině LK - ověřeno podáním SonoVue 1ml; levá síň středně dilatovaná (LAVi 46.8 cm3/m2); malá až střední mitrální regurgitace (2/4); aortální chlopeň trojcípá, bez vady, normální velikost aorty v sinusech valsalva, normální velikost ascendentní aorty; pravá komora nedilatovaná, s normální systolickou funkcí; pravá síň dilatovaná; malá trikuspidální regurgitace (1/4, PG 20 mmHg), nejsou přítomné známky klidové plicní hypertenze, elektroda(y) v pravostranných srdečních oddílech; malá pulmonální regurgitace (1/4); perikard bez separace

Závěr: LK dilatovaná s EDD 71mm a těžce omezenou EF 20-25%. Bez významné valvulopate. Normální funkce PK. Bez trombu v dutině LK (ověřeno podáním SonoVue).

Hospitalization summary

XX-letý polymorbidní pacient s DKMP (susp. levostranná ARVD/C), s recidivující KT, s implantovaným BiV ICD, stp. opak. RF ablacích endo i epikardiálních jak pro KT tak i pro FiS, nyní pro další epizody KT byl hospitalizován k RFA KT. Echokardiograficky LK dilatovaná s těžce omezenou EF 20-25%, normální funkce PK, bez trombu v dutině LK (ověřeno podáním SonoVue), bez významné valvulopate. DD provedena bipolární ablace mezikomorového basálního septa, dále ablace v kapsičkách ao chlopně, výkon bez komplikací. Pobyt na oddělení nekomplikován, telemonitoricky bez záchytu setrvalé arytmie, dle kontrolní echokg. po výkonu perikard bez výpotku.

Figure A.2: Sample from Cardiology dataset with pseudo-labeled sentences. Sentences highligted in green are pseudo-labeled by the BERTScore greedy approach. Sentences with bold orange font are pseudo-labeeld by the BERTScore pairwise approach.

Appendix B

Abstractive model predictions

Summary written by doctor

XX-letý muž s asymptomatickou parox. FIS nyní přijat ke kryoablaci. Laboratoř vstupně bez pozoruhodností, echokg porucha relaxace LKS, lehce dilatov. LS, lehká stacionární dilatace asc. aorty, jinak věku přiměřený nález. DD provedena kryoablace - IPŽ, výkon nekomplikovaný, třísla bez postpunkční patologie, echokg bez PV. Za další monitorace klidný SR.

Model-generated summary

XX-letý muž s asymptomatickou parox. FIS nyní přijat ke kryoablaci. Laboratoř vstupně bez pozoruhodností. Echokg normální velikost a normální systolická funkce obou komor, porucha relaxace LK, lehce dilatovaná levá síň, lehká stacionární dilatace asc. aorty, nevýznamné AV regurgitace, bez známek PH. DD kryoablace - IPŽ, výkon nekomplikovaný, třísla bez postpunkční patologie, echokg s odstupem bez výpotku. Za další monitorace klidný SR.

Figure B.1: An example of summary generated by the mT5-base model fine-tuned on the filtered **Acute Cardiology dataset**, which was labeled as '**Good**' during manual evaluation. The true summary written by the doctor is displayed above for comparison.

Summary written by doctor

Pacientka po kombinované transplantaci jater a ledviny s bilat. nefrektomií (DD) pro polycystozu, byla přijata k provedení protokolární biopsie štěpu. Výkon i následná observace proběhly bez komplikací. Histologicky patrny metabolické změny štěpu (převážně velkokapénková steatóza s ballooning degenerací v 85 % hepatocytů), s mírnou imunitní reakcí proti štěpu.

Model generated summary

Pacientka po kombinované transplantaci jater a ledviny s bilat. nefrektomií pro polycystosu byla plánovaně přijata k provedení protokolární biopsie jater. Výkon i následná observace byly bez komplikací. Histologicky popsány metabolické změny jaterní tkáně (převážně velkokapénková steatóza s ballooning degenerací v 85 % hepatocytů), s morfologií mírné imunitní reakce proti štěpu (RAI 2-3/9 – indeterminate).

Figure B.2: An example of summary generated by the mT5-large model fine-tuned on the filtered **Hepatogastroenterology dataset**, which was labeled as '**Good**' during manual evaluation. The true summary written by the doctor is displayed above for comparison.

Summary written by doctor

XX-letý obézní hypertonik, s dlouhodobě perzistující Fis (CHA2DS2VASc:1, antikoagulovaný Warfarinem, DD po thorakoskopické IPŽ), byl elektivně přijatý k endovazální RFA. Dle jícnoné echokardiografie bez intrakardiální trombózy, LKS nedilat., konc. remodelovaná s hraniční EF, bikuspidální ao chlopeň se stopovou regurgitací, asc. aorta s dilatací na 38mm, dále malé AV regurgitace s dilat. LS (LAVi 38.8 cm3/m2), PKS dilat. s hraniční systolickou funkcí, PFO s L-P zkratem bez průkazu P-L, bez zn. PV. Provedena re-IPŽ všech plícních žil, dále extenzivní ablace stropní i zadní dolní linie, dosažen blok na dolní zadní linii, bloku na stropě LS nelze dosáhnout. Dle kontrolního TTE bez PV po výkonu.

Model-generated summary

XX-letý obézní hypertonik, DD po thorakoskopické IPŽ, byl elektivně přijatý k endovazální RFA. Echokardiograficky EF LK 55% IAS s triviálním LP zkratem, PL zkrat nezobrazuji, ouško s normálními rychlostmi bez trombů, malá MiR, malá TriR, triviální echogenní separace kolem PK jinde perikard naléhá. Provedena RFA po COBRA ablaci - reizolace všech žil samostaně ostiálně, nedokončen box, blok na zadní dolní linii v LS, výkon bez komplikací. Třísla po výkonu klidná.

Figure B.3: An example of summary generated by the mT5-base model fine-tuned on the filtered **Acute Cardiology dataset**, which was labeled as '**Sufficient**' during manual evaluation. The true summary written by the doctor is displayed above for comparison.

Summary written by doctor

Polymorbidní XX-letý pacient planovaně přijat k provedeni septotomie v celkové anestézii. Pacient běhen hospitalizace bez známek komlikace výkonu a celkové anstézie. Pro lehkou kardiální dekompenzaci posílena chronická diuretická terapie za hospitalizace. Po výkonu tekutiny toleruje.

Model generated summary

Pacient s Zenkerovým divertiklem byl přijat plánovaně k Z-POEM/septotomii. Výkon i následná observace proběhly bez komplikací.

Figure B.4: An example of summary generated by the mT5-large model fine-tuned on the filtered **Hepatogastroenterology dataset**, which was labeled as '**Sufficient**' during manual evaluation. The true summary written by the doctor is displayed above for comparison.

Appendix C

Attached files

While not useful without the data we cannot publish, we provide the implementation of the used models and experiments for reference. The attached files have the following structure:

```
├── requirements.txt
├── scripts
│   ├── create_abstractive_dataset.py
│   ├── create_dataset.py
│   ├── create_extractive_dataset_xlmr.py
│   ├── generate_naive_and_random_extractive_summary.py
│   ├── pseudo_label_for_evaluation.py
│   ├── train_abstractive_aya.py
│   ├── train_abstractive_mbart.py
│   ├── train_abstractive_mt5.py
│   └── train_extractive_xlmr.py
├── setup.py
└── src
    ├── data
    │   ├── abstractive_dataset.py
    │   ├── extractive_dataset.py
    │   ├── lemmatize.py
    │   ├── pseudo_labeling.py
    │   ├── rouge_raw.py
    │   ├── text_dataset.py
    │   └── text_splitting.py
    ├── evaluate.py
    ├── model
    │   ├── abstractive.py
    │   └── extractive.py
    └── vizualization
        └── print.py
```