FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Bachelor's Thesis

Automatic Creation of Summaries of Historical Documents

Václav Tran

PILSEN, CZECH REPUBLIC

2024

FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

DEPARTMENT OF
COMPUTER SCIENCE
AND ENGINEERING

Bachelor's Thesis

Automatic Creation of Summaries of Historical Documents

Václav Tran

Thesis advisor
Doc. Ing. Pavel Král, Ph.D.

PILSEN, CZECH REPUBLIC 2024

Citation in the bibliography/reference list:
TRAN, Václav. *Automatic Creation of Summaries of Historical Documents*. Pilsen, Czech Republic, 2024. Bachelor's Thesis. University of West Bohemia, Faculty of Applied Sciences, Department of Computer Science and Engineering. Thesis advisor Doc. Ing. Pavel Král, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Václav TRAN**
Osobní číslo:	**A21B0299P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Automatické vytváření souhrnů historických dokumentů**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte dostupné datové sady pro automatickou sumarizaci textu.
2. Seznamte se s relevantními metodami automatické sumarizace textu s důrazem na metody založené na neuronových sítích.
3. Vytvořte vlastní datovou sadu českých historických dokumentů pro sumarizaci textu.
4. Navrhněte a implementujte prototyp systému pro automatickou sumarizaci textu, který integruje alespoň dvě vybrané sumarizační metody.
5. Prototyp otestujte na vybrané a nově vytvořené datové množině.
6. Zhodnoťte dosažené výsledky a navrhněte další možná rozšíření.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Pavel Král, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **2. října 2023**
Termín odevzdání bakalářské práce: **2. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2023

Declaration

I hereby declare that this Bachelor's Thesis is completely my own work and that I used only the cited sources, literature, and other resources. This thesis has not been used to obtain another or the same academic degree.

I acknowledge that my thesis is subject to the rights and obligations arising from Act No. 121/2000 Coll., the Copyright Act as amended, in particular the fact that the University of West Bohemia has the right to conclude a licence agreement for the use of this thesis as a school work pursuant to Section 60(1) of the Copyright Act.

In Pilsen, on 14th March 2024

. . . . . .

Václav Tran

Abstract

In the realm of automatic text summarization, the application of neural networks has shown a promising performance. This thesis probes into the task of automatic summarization of Czech historical documents, a largely unexplored niche area with a scant amount of datasets available. To evaluate and improve the performance of our methods, we create our own dataset constructed from a corpus of historical documents. Then we fine-tune and utilize Transformer-based models Mistral 7B and mT5. We also implemented and evaluated a method, where we utilize state-of-the-art machine translation and English summarization methods to generate Czech summaries which we refer to as Translation-Summarization-Translation (TST). The performance of these methods set a new baseline for the task of summarizing Czech historical documents.

Abstrakt

Ve světě automatického vytváření shrnutí dokumentů a textů se neuronové sítě ukázaly jako slibná technologie. Tato bakalářská práce se zabývá automatickým vytvářením souhrnů českých historických dokumentů, což je téma, které není příliš prozkoumané. Pro vyhodnocení a případného zlepšení výkonu našich metod jsme vytvořili vlastní dataset ze sady historických dokumentů. Poté jsme natrénovali a využili modely Mistral 7B a mT5, které využívají Transformer architekturu. Navíc jsme také naimplementovali a vyohodnotili metodu, která využívá state-of-the-art metody pro strojový překlad a metody pro automatické vytváření shrnutí textu v angličtině. Tuto metodu označujeme jako Translation-Summarizaton-Translation (TST). Výsledky zmiňovaných metod představují nový standard pro úkol shrnutí českých historických dokumentů.

Keywords

Neural network • Artificial intelligence • Text summarization • Czech historical documents

Acknowledgement

Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Contents

Introduction 1

Recent breakthroughs in natural language processing and neural networks have achieved remarkable progress, significantly advancing the state of the art in the field. However, the task of summarizing historical documents in Czech poses a considerable challenge. These documents, written in archaic Czech, present significant difficulties for neural networks primarily trained on modern data. In the domain of natural language processing, a majority of methods and datasets predominantly cater to the English language, resulting in a notable scarcity of datasets tailored to the Czech language.

In this bachelor's thesis, the focus is on utilizing the capabilities of neural networks for the automatic summarization of Czech historical documents. Recognizing the immense potential of neural networks in text processing tasks such as text summarization, this work seeks to design and develop a system that manages to summarize historical documents in Czech while preserving the integrity and essence of the original documents. The work foundation is first built through an examination of existing datasets necessary for neural network training. Datasets are instrumental for teaching a neural network to understand the structural nuances and complexities of the task at hand. Insights derived from the dataset analysis impact the selection process, ensuring the most suitable dataset is chosen for this thesis' task. A custom dataset, comprised of Czech historical documents and their summaries, will be created for evaluating the selected summarization methods. This dataset is prepared to serve as a benchmark for assessing the summarization methods' performance. We will additionally find a metric that can measure the methods' performance on the custom dataset. The research and implementation of text summarization methods form a significant part of this work. We aim to identify methods that align with the thesis' goals and are most likely to deliver optimal performance. This involves a detailed comparison of text summarization methods, identifying those with the highest performance or the highest potential for multilingual text summarization, and subsequent training of the neural network on the most suitable dataset.

The final phase involves evaluation of the methods on the custom dataset to gain insights into each method's performance using an appropriate metric.

Neural Networks 2

Neural networks [1] are computational models inspired by the structure and function of the human brain. They have gained significant prominence in various fields, including machine learning and artificial intelligence. This chapter provides a basic understanding of neural networks.

2.1 Basic Components

A neural network consists of interconnected nodes organised into layers. The basic components include:

- **Neurons:** The elementary units of neural networks. They receive inputs, apply a transformation, and produce an output.
- **Layers:** Neurons are organised into layers - input, hidden, and output layers. Information flows from the input layer through the hidden layers to the output layer.
- **Weights and biases:** Each connection between neurons is associated with a weight, representing the strength of the connection. Biases are constants, which affect the output of a neuron.

2.2 Activation Function

Activation function is a function that is used to calculate the output of a neuron. The nonlinear activation function introduces non-linearity into the network. Such functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU). They enable the network to learn complex patterns and relationships in the data.

2.3 Training Process

Neural networks learn from data through a process called training. Training involves modifying the weights and biases of the neural network so that the output of the neural network is closer to the target output. The key steps involved are:

1. **Forward propagation:** The input data is passed through the network, layer by layer, to generate predictions.
2. **Loss function:** A measure of the difference between the predicted output and the actual target is calculated.
3. **Backpropagation:** The error is propagated backward through the network, and the weights and biases are adjusted to minimize the loss.
4. **Training loss:** This is the measure of error for the training dataset, representing the difference between the predicted outputs and the actual targets within the training data. It is calculated using the loss function during the training process. The training loss provides information on how well the model is learning from the training data.
5. **Validation loss:** This represents the error or loss on the validation dataset, which is a separate set of data not used during training. The validation loss is calculated using the same loss function as the training loss but applied to the validation data. It is a metric for evaluating how well the model generalizes to unseen data.

2.4 Types of Neural Networks

There are various types of neural networks, each designed for specific tasks. Some common types include:

- **Perceptron:** The simplest type of a neural network. It is a type of linear classifier, therefore it can only solve linearly separable problems.
- **Feedforward Neural Networks (FNN):** Information flows in one direction, from input to output without going backwards. Also known as multilayer perceptron.
- **Recurrent Neural Networks (RNN):** Neurons have connections that form cycles, allowing them to retain information about previous inputs.
- **Convolutional Neural Networks (CNN):** Specialized for processing grid-like data, such as images.

Datasets 3

The success of natural language processing tasks, such as text summarization, heavily relies on the availability of high-quality datasets for training and evaluation. In this chapter, we explore and analyze several datasets considered for fine-tuning summarization models. The primary objective of this thesis is the summarization of historical documents in the Czech language, a task that demands a specialized dataset to ensure the model's proficiency in handling of the target language.

3.1 SumeCzech

SumeCzech [2] is a dataset comprised of one million Czech news articles sourced from five Czech news sites: České Noviny[1], Deník[2], iDNES[3], Lidovky[4], Novinky.cz[5]. The dataset is structured in the JSON Lines format, with each document represented as a JSON object containing fields such as URL, headline, abstract, text, subdomain, section, and publication date. Data cleanup involved filtering out irrelevant entries and removing unnecessary information such as advertisements and links. Language recognition was performed to retain only Czech documents. Further cleaning steps involved dropping documents with empty headlines, short abstracts, or very short full texts. Duplicates were also removed based on headline, abstract, or text similarity. The dataset can be used for different summarization setups, including headline generation and multi-sentence abstract generation. The authors also propose a language-agnostic variant of the ROUGE [3] metric for automatic evaluation called $\text{ROUGE}_{\text{RAW}}$. The dataset was created at the Institute of Formal and Applied Linguistics from Charles University. It can be downloaded using scripts[6] provided by the authors of the paper. The analysis of text word counts within the SumeCzech

[1] https://ceskenoviny.cz
[2] https://denik.cz
[3] https://idnes.cz
[4] https://lidovky.cz
[5] https://novinky.cz
[6] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2615

dataset is detailed in Table 3.1. This analysis provides insights into the dataset's composition, highlighting the average, median, maximum, and minimum word counts across the different splits.

Table 3.1: Analysis of the SumeCzech text word count

Metric	Train	Dev	Test
Average	409.27	411.74	415.71
Median	325	326	329
Maximum	14 745	13 283	12 007
Minimum	93	99	99

3.2 CNN/Daily Mail

The CNN/Daily Mail Dataset [4] is a dataset comprised of over 300k English news articles from CNN and Daily Mail, where each article also contains the highlight of the article written by the article author. The dataset undergoes three versions: 1.0.0 focuses on question answering, using CNN and Daily Mail articles; 2.0.0 and 3.0.0 shift to summarization of long articles into one or two sentences. Version 3.0.0 is non-anonymized, revealing names of the entities that were hidden from the highlight for the purpose of question answering. The initial data collection was carried out by authors of [4]. The summarization variant was produced by Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Bing Xiang of IBM Watson, and Caglar Gulcehre of Université de Montréal. The non-anonymized and publicly available version of the dataset, which was used in [5] is provided by Abigail See of Stanford University, Peter J. Liu of Google Brain, and Christopher D. Manning of Stanford University.

3.3 XSum

XSum [6] dataset contains over 226k BBC[7] articles from a wide variety of domains such as news, sports, science, politics, where each article comes with a one-sentence summary. The dataset was created using the same methodology authors of [4] used to create the first version of the dataset in Section 3.2. Extractive methods perform poorly on XSum, highlighting its lower bias towards extractive summarization. The dataset, although lacking diversity as it focuses on a single news outlet and follows a single-sentence summarization style, is large enough for neural network training.

[7] `https://bbc.com`

3.4 Arxiv Dataset

Arxiv Dataset [7] is a dataset that contains 215k article and abstract pairs, where both elements of the pairs were retrieved from scientific papers that are available on the arXiv[8] website. The dataset only includes papers that have an abstract, a discourse structure, and are not excessively long or short. The papers are converted from LATEX to plain text using Pandoc[9], and figures and tables are removed using regular expressions. Math formulas and citation markers are normalized with special tokens. Conclusion sections are analysed and detected and sections after conclusion are removed.

3.5 XLSum

XLSum [8] is a dataset comprised of over 1 million article-summary pairs extracted from various BBC[10] sites. It covers a range of 44 languages, from low-resource ones like Bengali and Swahili to high-resource languages such as English and Russian. However, the Czech language is not included in the dataset. Summaries, which were written by the authors, were typically presented as bold paragraphs in the first two paragraphs of each article. To ensure effective extraction, heuristics mentioned in [8] were used. This dataset lacks in the diversity of summarization styles, however, it covers a multitude of languages and offers a wide collection of articles and their summaries.

3.6 MLSUM

MLSUM [9] is a dataset containing over 1.5 million article-summary pairs in five different languages. The included five languages are: German, Russian, French, Spanish, and Turkish. However, similarly to the dataset described in Section 3.5, the Czech language is also not included in the dataset. The data collection process involved selecting newspapers in each language, ensuring the newspapers contained a broad representation of topics and a substantial number of articles in their online archives. The chosen newspapers were Le Monde[11], Suddeutsche Zeitung[12], El Pais[13], Moskovskij Komsomolets[14], and Internet Haber[15]. Articles from 2010 to

[8] https://arxiv.org
[9] https://pandoc.org
[10] See footnote 7
[11] https://lemonde.fr
[12] https://sueddeutsche.de
[13] https://elpais.com
[14] https://mk.ru
[15] https://internethaber.com

2019 were crawled and archived, with a filter applied to exclude very short articles or summaries.

3.7 BOOKSUM

BOOKSUM [10] is a dataset designed for the task of summarizing long texts like novels, plays, and stories. It includes summaries of these texts at different levels: paragraphs, chapters, and entire books. All the books were either written in English or translated to English. BOOKSUM is structured to support both extractive and abstractive summarization methods. The primary source of these documents was the Project Gutenberg repository[16], which offers a vast collection of free eBooks. The summaries were gathered from various independent sources via the Web Archive[17]. The authors trained and evaluated multiple extractive and abstractive summarization models for the establishment of baseline performance for future research.

Table 3.2: The SumeCzech dataset is the only dataset that aligns with the task of summarizing historical documents in the Czech language. Other datasets were explored for broader insights into summarization tasks and multilingual contexts. This table summarizes key information about the datasets explored for fine-tuning summarization models, such as dataset type, size (amount of rows), language, and train/dev/test split ratio.

Name	Type	Dataset Size	Train/Dev/Test	Language
SumeCzech	News	1 000 000	86.5/4.5/4.5	Czech
CNN/Daily Mail	News	311 672	92/4.3/3.7	English
XSum	BBC	226 711	90/5/5	English
MLSUM	News	1 500 000	Described in [9]	Multilingual
Arxiv Dataset	Scientific	215 000	94/3/3	English
BOOKSUM	Literary	12 500	80/10/10	English

[16] https://www.gutenberg.org/
[17] https://web.archive.org/

Methods 4

In this chapter, we will discuss various text summarization methods. Text summarization can be split into two types: abstractive and extractive. Each approach uses different methodologies for extracting essential information from source texts. Abstractive summarization involves the creation of a summary with newly generated sentences that may not exist in the source document. This approach requires a deeper understanding of the content and the ability to generate concise, coherent, and contextually appropriate sentences. Extractive summarization constructs a summary using existing sentences directly extracted from the source document. This method selects sentences that are the most representative of the document's content. We will also discuss the metric that will be used for method performance evaluation.

4.1 Extractive Summarization Methods

Extractive summarization involves the identification and extraction of representative sentences or phrases from a given document to form a coherent summary. This section examines notable extractive summarization methods that use a variety of techniques for identifying and selecting crucial information from the source text.

4.1.1 Leveraging BERT for Extractive Summarization On Lectures

This method introduced in the research paper titled *Leveraging BERT for Extractive Text Summarization on Lectures* [11] leverages the BERT [12] model for generating text embeddings and employs ***K-Means Clustering*** [13] to identify sentences closest to the centroid. Sentences closest to the centroids were then chosen for the summary. However, the author mentions using BERT [12] which was pre-trained on a large English corpus as described in [12], therefore results might be less than ideal with Czech documents.

4.1.2 TextRank

TextRank [14] is an algorithm for extractive summarization and keyword extraction. It was created by researchers Rada Mihalcea and Paul Tarau from the University of North Texas. The algorithm is based on the idea of representing a document as a graph of sentences, where vertices represent sentences and edges are based on the content overlap between sentences. PageRank [15] algorithm is then used to compute the importance of each sentence. A final summary is then created with the most important sentences.

4.1.3 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) method [16] can be used for language-agnostic extractive summarization using a simple algorithm. The method relies on the importance of words in a document with respect to a set of documents.

4.1.3.1 Term Frequency

The term frequency (TF) of a word in a document is calculated as the number of times the word appears in the document.

$$\mathrm{TF}(t, d) = \frac{\text{Number of times word } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

where $\boldsymbol{t}$ is the word and $\boldsymbol{d}$ is the document.

4.1.3.2 Inverse Document Frequency

The inverse document frequency (IDF) of a word is calculated as the logarithm of the total number of documents divided by the number of documents containing the word.

$$\mathrm{idf}(t, D) = \log \frac{1 + \text{Number of documents } D}{1 + \text{Number of documents where the word } t \text{ appears}}$$

where $\boldsymbol{D}$ is a set of documents and $\boldsymbol{t}$ is the word.

The addition of one to both the numerator and denominator serves the purpose of preventing division by zero, in cases where the term $\boldsymbol{t}$ is absent from any document.

4.1.3.3 Result

TF-IDF of a word $\boldsymbol{t}$ is then calculated as a simple multiplication of TF and IDF values. The importance of each sentence is calculated based on the sum of TF-IDF scores of words in that sentence. A certain percentage or a fixed number of sentences with the highest TF-IDF scores are then selected to form the final summary.

4.2 Abstractive Summarization Methods

Abstractive summarization involves the generation of summaries that capture the essential meaning of a given text. The coming of neural network architectures, particularly Transformers [17], has revolutionised the field, enabling the development of many state-of-the-art (SOTA) abstractive summarization models. This section explores some prominent models that have demonstrated significant advancements in abstractive summarization tasks.

4.2.1 T5

T5, or Text-To-Text Transfer Transformer, is a pre-trained model that approaches natural language processing tasks in a text-to-text framework. Introduced in the research paper titled *T5: Text-To-Text Transfer Transformer* [18], T5 treats all text processing problems as text-to-text problems, where text is the input and the output is also a text. This enables the application of the same model, loss function, and set of hyperparameters across a wide range of tasks. Pre-training was done using the ***span-corruption*** objective. The term corruption refers to the removal or modification of parts of the input text, which the model must then predict based on the remaining unaltered text. T5 demonstrates competitive performance across multiple benchmarks such as SuperGlue [19], GLUE [20] or SQuAD [21] showcasing its effectiveness in a wide range of natural language processing tasks. The model was pre-trained using the “Colossal Clean Crawled Corpus" (C4) dataset that was introduced in the research paper [18] along with T5. The dataset is comprised of a large amount of cleaned English text extracted from Common Crawl[1]. The model is available in five model sizes: t5-small (60M parameters), t5-base (228M), t5-large (770M), t5-3B (3B), and t5-11B (11B).

4.2.2 PEGASUS

PEGASUS is an abstractive summarization model that belongs to the family of Transformer-based [17] models. The model was introduced in a research paper titled *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization* [22].

The author used a self-supervised objective tailored for text summarization called ***Gap Sentences Generation*** (GSG) to pre-train the model on large corpora of news and articles. The core idea of GSG is to remove (mask) important sentences from an input document and then generate these masked sentences as a single output sequence, using the remaining content. The results show that the pre-training has

[1] `https://commoncrawl.org/`

considerable positive effects on downstream summarization tasks in comparison to PEGASUS without pre-training.

According to Zhang et al., "PEGASUS achieves SOTA performance on all 12 downstream datasets measured by ROUGE scores" [22]. The model was pre-trained using the C4 dataset described in Section 4.2.1 and the HugeNews dataset, which was introduced in the research paper [22] along with PEGASUS.

4.2.3 BART

BART is a pre-trained transformer [17] encoder-decoder model suited for fine-tuning on text generation tasks such as summarization or translation. The model was first introduced in a research paper titled *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension* [23]. Pre-training is done using various methods such as token masking, token deletion, and text infilling. BART matches the performance of RoBERTa [24] on GLUE [20] and SQuAD [21] and achieves SOTA performance in tasks like summarization and question answering.

4.2.4 mT5

mT5 is a multilingual variant of T5 (described in Section 4.2.1) introduced in a research paper titled *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer* [25]. Similarly to T5, it also uses a text-to-text framework. The authors tried to deviate as little as possible from the steps done to create the original T5 model. The research paper introduces a multilingual variant of the C4 dataset called mC4, which is used as the pre-training dataset. Czech data was included in the mC4 dataset, therefore the tokenizer can more efficiently process Czech words. The model is available in five model sizes: mt5-small (300M parameters), mt5-base (580M), mt5-large (1.2B), mt5-xl (3.7B), and mt5-xxl (13B).

4.2.5 mBART

mBART is a multilingual variant of BART (described in Section 4.2.3). It uses the same pre-training objective as BART and it uses a multilingual Common Crawl[2] corpus that contains data from 25 languages, including Czech, as the dataset. Many mBART models were pre-trained using only a subset of the multilingual corpus such as mBART02, which is a bilingual model where one language of the bilingual pair is always English. The mBART model that was pre-trained using the whole multilingual corpus is called mBART25.

[2] `https://commoncrawl.org/`

4.2.6 Mistral 7B

Mistral 7B, introduced by Jiang et al. [26], is a 7-billion-parameter language model designed for high performance and efficiency across a range of NLP tasks. This model outperforms the existing 13B and 34B models in various tasks such as reasoning, mathematics, and code generation capabilities on a wide range of evaluation metrics. Mistral 7B incorporates attention mechanisms such as ***Grouped-Query Attention*** (GQA) [27] and ***Sliding Window Attention*** (SWA) [28] to enhance inference speed and manage long sequences efficiently, without significant trade-offs in computational cost.

SWA has each token focus only on nearby tokens, unlike standard full attention, where each token has to focus on every other token. This creates subquadratic computational complexity with respect to sequence length, which allows Mistral 7B to handle longer sequences without a substantial increase in computational resources.

GQA is a technique that improves the efficiency of language models by optimizing how they focus on different parts of input data. It divides attention heads into groups, allowing each group to focus on the same parts of input data, which in turn reduces computational load.

4.2.7 LongT5

LongT5, introduced by Guo et al. [29], is an extension of the T5 model that is designed to efficiently handle long sequences. By integrating attention mechanisms from a transformer architecture ***Extended Transformer Construction*** (ETC) [30] and adopting pre-training strategies from summarization pre-training PEGASUS (described in Section 4.2.2) within the T5 architecture, LongT5 achieves significant improvements in processing long documents in comparison to T5.

LongT5 has demonstrated SOTA performance on various summarization and question-answering tasks, demonstrating its ability to handle significantly longer sequences than the original T5 models without a substantial increase in computational costs.

4.3 Leveraging Translation for Czech Text Summarization

The amount of models that can summarize text in Czech is limited and the only large publicly available Czech text summarization dataset available is SumeCzech (3.1). However, there are many capable English text summarization models and additionally, there exists a range of SOTA translation models capable of converting text between Czech and English. Therefore another approach towards abstractive

summarization could be translating the given Czech text to English, summarizing it using the preferred English summarization model and translating it back to Czech. For the purpose of shortness, this method will henceforth be referred to as ***Translation-Summarization-Translation*** (TST).

4.3.1 ALMA-R

One such SOTA translation model is ALMA-R, introduced by Xu et al. in *Contrastive Preference Optimization: Pushing the Boundaries of LLM Performance in Machine Translation* [31] which matches or exceeds GPT-4 [32] or WMT[3] winners on various translation benchmarks.

The model was trained using ***Contrastive Preference Optimization*** (CPO), a novel training method that was proposed in the paper above. Traditional methods such as ***Supervised fine-tuning*** (SFT) [1] methods train models to mimic reference translations, where the resulting model's performance relies on dataset quality. Xu et al. demonstrate that translations by advanced models can be superior to reference translations. Its training objective is designed to minimize the error not between the model output and a single reference translation but rather increase the likelihood of generating a translation that is preferred and decrease the likelihood of generating a dis-preferred one. This involves generating a triplet of translations for a given source sentence: one from a reference (human-generated), one from GPT-4, and one from an ALMA [33] model (prior to CPO application). Each translation in the triplet is then scored using reference-free translation quality evaluation models and the translations are ranked based on their quality. The highest-scoring translation is labeled as the preferred translation, and the lowest-scoring as the dis-preferred.

The code and models are released to the public at `https://github.com/fe1ixxu/ALMA`. The model is also available through Hugging Face (described in Chapter 5) in 7B version or 13B version.

4.4 ROUGE Metric

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [3] metric is an automated tool designed to evaluate the quality of summaries by comparing them with a set of reference summaries crafted by humans. Its core objective is to measure the overlap between the generated summary under evaluation and reference summaries. Overlap between two summaries is the amount of certain units that are contained in both summaries. These units can be n-gram word sequences or longest common subsequences (LCS). ROUGE encompasses several measures, each offering a different perspective on the summary's alignment with the reference texts. These

[3] `https://machinetranslate.org/wmt`

include measures such as ROUGE-N or ROUGE-L. Each measure offers a different view of the generated summary's quality, from precise word and phrase replication (ROUGE-N) to the preservation of overall structure (ROUGE-L). While ROUGE offers more measures than the ones mentioned in this section, we will only focus on ROUGE-1, ROUGE-2, and ROUGE-L.

4.4.1 ROUGE-N

ROUGE-N focuses on the overlap of n-grams between the generated summary and the reference summaries. Its precision can be calculated using:

$$\text{Precision} = \frac{\text{Number of overlapping n-grams}}{\text{Number of n-grams in the generated summary}}$$

The recall is calculated by considering the overlap of n-grams between the generated summary and the reference summaries with respect to the reference summaries. The formula for recall is:

$$\text{Recall} = \frac{\text{Number of overlapping n-grams}}{\text{Number of n-grams in the reference summary}}$$

The F-score, specifically the F1-score, is a measure which represents the precision and recall in one metric can be calculated as:

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

4.4.2 ROUGE-L

ROUGE-L focuses on the LCS to evaluate the similarity between the generated summary and the reference summaries. The LCS is the longest sequence of words that appears in both the generated and reference summaries in the same order, but not necessarily continuously. Precision in the context of ROUGE-L is calculated by considering the length of the LCS between the generated summary and the reference summaries relative to the length of the generated summary. The formula for precision is:

$$\text{Precision}_L = \frac{\text{Length of LCS}}{\text{Length of the generated summary}}$$

Recall is calculated by considering the length of the LCS relative to the length of the reference summary. This shows how much of the content from the reference summaries is captured in the generated summary. The formula for recall is:

$$\text{Recall}_L = \frac{\text{Length of LCS}}{\text{Length of the reference summary}}$$

The formula for the F1-score is the same as for ROUGE-N but applies to the precision and recall calculations specific to ROUGE-L:

$$\text{F1-score}_L = 2 \times \frac{\text{Precision}_L \times \text{Recall}_L}{\text{Precision}_L + \text{Recall}_L}$$

4.4.3 $\text{ROUGE}_{\text{RAW}}$

$\text{ROUGE}_{\text{RAW}}$ [2] is a language-agnostic variant of ROGUE proposed by authors of SumeCzech. This variant maintains the core objectives of ROUGE, focusing on the comparison of n-grams and least common subsequences between the generated summaries and reference summaries. However, by discarding language-specific processing methods like stemmers, stop words, and synonym lists, $\text{ROUGE}_{\text{RAW}}$ becomes more adaptable to a variety of languages.

Implementation 5

In this chapter, we will introduce the Hugging Face ecosystem and its libraries, which were heavily used during the implementation. Then we will discuss the creation of an evaluation dataset on which we will evaluate our chosen methods. Lastly, we will provide an in-depth discussion on the implementation of these methods. All implementations were conducted in the Python programming language. We also extensively utilized Jupyter notebooks [34] for development and testing.

5.1 Hugging Face

Hugging Face[1] is a prominent platform in the machine learning community, providing an ecosystem for working with SOTA models and datasets. The platform hosts a vast collection of pre-trained models for various natural language processing (NLP) tasks. One of the key contributions towards the platform's popularity is the Hugging Face Transformers [35] library, which provides a high-level abstraction for the usage of many large language models (LLM). It allows the users to perform inference, pre-training, and fine-tuning without a deep knowledge of the used architecture.

Inference is the process of using a trained model to make predictions or decisions. Pre-training is the initial training of a model on a large dataset, where the model gains general knowledge of the task or tasks it is trained on. Fine-tuning is the process of additional training of a pre-trained model for the purpose of improving the model's performance on specific tasks. For example, T5 (described in Chapter 4) is a model that was pre-trained on a variety of NLP tasks. However, it can be fine-tuned on a narrower or more specific dataset to enhance its performance for particular tasks such as text summarization, question answering, or translation. This fine-tuning process adapts T5 to the specific characteristics of the task at hand, leveraging the general knowledge it acquired during pre-training and applying it to solve the targeted problems more effectively.

[1] `https://huggingface.co/`

5.1.1 Hugging Face Transformers

The Hugging Face Transformers (HFT) [35] library is a framework that supports the usage of thousands of pre-trained models for various machine learning tasks text, vision, and audio. It supports integration with popular machine learning libraries: PyTorch[2], TensorFlow[3], and JAX[4].

The library offers not only high-level APIs for training and inference but also low-level access for more detailed customization.

The library also simplifies the deployment and usage of many models. For example, deploying a fine-tuned LongT5 for text generation can be done with just a few lines of code (see 5.1).

5.1.1.1 Trainer

For model training, we used HFT Trainer class[5]. HFT Trainer allows users to train their models for specific tasks such as, but not limited to NLP, computer vision (object detection, image classification) or speech recognition. The class provides an abstraction for the training loop, automating processes such as the forward and backward passes.

For dealing with computationally heavy tasks, it also supports distributed training across multiple GPUs, CPU offloading and disk offloading. The Trainer also supports ***Parameter-Efficient Fine-Tuning*** (PEFT) [36] methods which significantly reduces hardware requirements for model training by fine-tuning only a fraction of the model's parameters rather than all of the model's parameters. It also includes logging and monitoring functionalities, enabling users to track training metrics, such as training loss, validation loss, and evaluation metric scores on chosen metrics, throughout the training process.

5.1.1.2 Inference

The inference process in text generation models, including those like mT5 and Mistral 7B, involves generating new text based on an input sequence of tokens. These models produce a probability distribution for each subsequent token, from which the next token is selected. The method of selection relies heavily on this probability distribution and the chosen text generation strategy for selecting subsequent tokens.

Settings. In HFT, several parameters can be adjusted to influence the text generation strategy, therefore affecting the model's output. These parameters include:

[2] `https://pytorch.org/`
[3] `https://www.tensorflow.org/`
[4] `https://jax.readthedocs.io/en/latest/`
[5] `https://huggingface.co/docs/transformers/main_classes/trainer`

- `max_new_tokens:` Max amount of tokens that can be generated.
- `do_sample:` If set to `false`, uses default decoding strategy greedy search, where the token with the highest probability is chosen. If set to `true`, allows the usage of different decoding strategies as described in Hugging Face documentation[6].
- `temperature:` Modifies the probability distribution of tokens. A higher temperature usually leads to a more uniform probability distribution.
- `top_p:` Only tokens that cumulatively comprise up to `top_p` probability can be chosen.
- `repetition_penalty:` Penalizes previously generated tokens to reduce repetition in the output. For more details see [37].

Adjusting these parameters allows for modification of the generation process, allowing the creation of outputs that range from highly deterministic to varied and non-deterministic, depending on the desired outcome.

Pipeline. HFT also includes a feature known as the pipeline[7]. The pipeline simplifies the usage of inference without requiring extensive coding. A pipeline bundles together a model and its associated tokenizer, streamlining the workflow for common tasks such as text classification, named entity recognition, text generation, summarization, and more. Users can create a pipeline for the desired task using the `text` parameter and apply it to input text without the need for any extensive coding. For a pipeline usage example in code, see 5.1.

Source code 5.1: Text summarization inference example using pipeline

```
from transformers import pipeline

text = "A long string for summarization." #text to be
    summarized
summarizer = pipeline(
    text = "summarization", #the performed task
    model = "pszemraj/long-t5-tglobal-xl-16384-book-summary",
    #the model used
)
result = summarizer(text)
print(result[0]['summary_text']) #prints out summary to the
    console
```

[6] https://huggingface.co/docs/transformers/generation_strategies
[7] https://huggingface.co/docs/transformers/main_classes/pipelines

5.1.2 Hugging Face Datasets library

Hugging Face Datasets[8] is a library for managing and creating datasets. Users can use the datasets library to load a variety of formatted datasets from the Hugging Face platform or they can load their own custom datasets through methods available in the library. By leveraging the Apache Arrow[9] format, the library is optimized to process large datasets with high speeds and efficiency.

The SumeCzech dataset is only available through scripts provided by the author. However, the Hugging Face Datasets library supports many formats, including the JSON Lines format, which is the format that the SumeCzech dataset uses.

5.2 Creating Dataset for Model Evaluation

We obtained data from Porta fontium [38], more specifically, OCR-processed versions of historical journals, *Posel od Čerchova* and *Domažlické listy*. The received journals span a publication period from the 19th century up to the beginning of the 20th century. While constructing the dataset, we only used the texts from the journal *Posel od Čerchova*.

Porta fontium [38] is a joint Czech-Bavarian project aimed at reconnecting historically significant archival materials related to Czech and German histories, which were physically separated in the past. Through digitization, this project facilitates the creation of a virtual collection, making these archives accessible to the public, researchers, and regional historians via a shared online platform.

5.2.1 Dataset Format

Before we started with the creation of the dataset, we had to decide on its format. The dataset is therefore systematically organized, initially sorted by journal titles and their respective publication years. These journals are further divided into monthly issues, which are then subdivided into individual pages. A total summary of the issue resides alongside the list of individual pages. The structure of an individual page is as follows:

- **text:** OCR-processed text extracted from the given page, a digital rendition of the original printed content.
- **summary:** Summary of the page, which is no more than 5 sentences long.
- **year:** Publication year of the journal.

[8] `https://huggingface.co/docs/datasets/index`
[9] `https://arrow.apache.org`

- **journal:** This identifier specifies the issue source: the day, month, and the number of the issue is contained within this identifier.
- **page_src:** The name of the file, where the contents of *text* identifier come from.
- **page_num:** The number of the page.

Initially, our approach was focused solely on summarizing individual pages. However, after consultations with the data providers at Porta fontium, we expanded our format to include a total summary for each issue in addition to the page summaries. For a visual representation of the dataset's format, see 5.2.

Source code 5.2: Showcase of the dataset format

```
{
  "posel-od-cerchova-1882": {
    "03-30-n1": {
      "pages": [
        {
          "text": "Text from file described by page_src",
          "summary": "Summary of contents of text",
          "year": "1882",
          "issue": "03-30-n1",
          "page_src": "posel-od-cerchova-1872-03-30-n1_0010.jp2.txt",
          "page_num": 1
        }
      ],
      "summary_total": "Summary of issue 03-30-n1"
    }
  }
}
```

5.2.2 Building the Dataset

The construction of the dataset involved addressing the challenge of creating summaries for the provided texts, which were composed in archaic Czech and in some rare cases even German. The texts also covered a variety of different topics, from local news surrounding Domažlice, opinion pieces, and various local advertisements to internal and worldwide politics and feuilletons. Furthermore, it was important to construct a dataset of sufficient size to ensure the accuracy and reliability of our evaluations. These aspects added complexity to the summarization task.

To overcome this, we employed SOTA LLMs, including GPT-4 [32](specifically the `gpt-4-1106-preview` version) and Claude 3 Opus [39] (Opus) (specifically the

`claude-3-opus-20240229` version), for text summary creation. These models were selected based on their SOTA performance in many NLP tasks and they proved to be capable of producing satisfactory summaries. In regards to complete issue summaries, even though both models were capable of processing the entire issue all at once, we concatenated the summaries of the individual pages in their respective order and then summarized the concatenation. This approach was adopted because initial experiments revealed that summarizing the entire issue all at once led to overly broad summaries, due to the diverse range of topics covered within each issue. In contrast, summarizing concatenated individual page summaries preserved a higher level of detail.

While generating the summaries, it was crucial for them to be concise and due to the fact, that most implemented methods used the SumeCzech dataset for fine-tuning, we wanted the summaries to be created in the style of a news reporter. Therefore, all inputs to the models generating the summary were prepended with:

```
Vytvoř shrnutí následujícího textu ve stylu novináře. Počet
    vět <= 5:
```

Through this methodology, we summarized 432 pages, effectively resulting in the creation of 100 issue summaries. To ensure accuracy and prevention of any major mistakes, the summaries underwent a human review.

In our experience, we observed that while both models produced summaries of acceptable quality, Opus tended to create more succinct and stylistically appropriate summaries, closely aligning with the news reporter format. However, there were instances where Opus' summaries exhibited an excessive focus on a single topic. On the other hand, GPT-4 aimed to incorporate a greater level of detail within the five-sentence constraint but occasionally deviated from the specified stylistic prompt. When the model-generated summary exhibited significant stylistic deviations or excessive focus on a single topic, we either modified or regenerated it until an acceptable version was achieved.

5.3 mT5

The decision to train the mT5 model was motivated by its multilingual capabilities and its proven track record in achieving success[10,11] across various text summarization tasks in different languages. Despite the model being available in several sizes, constraints related to computational resources limited our efforts to the base variant, the only model size we managed to train successfully. Table 5.1 presents a detailed analysis of the SumeCzech dataset after tokenization using the mT5 tokenizer. In

[10] `https://huggingface.co/csebuetnlp/mT5_multilingual_XLSum`
[11] `https://huggingface.co/tsmatz/mt5_summarize_japanese`

comparison with Table 3.1, we can see that on average one word is approximately equivalent to two tokens. The table includes tokenized text and abstract metrics for the training, development (validation), and test splits of the dataset, displaying average, median, maximum, and minimum values.

Table 5.1: SumeCzech text and abstract analysis using mT5 tokenizer

Metric	Train	Dev	Test
Tokenized Text			
Average	901.94	906.91	917.01
Median	719	720	731
Maximum	32 857	28 744	26 473
Minimum	185	196	193
Tokenized Abstract			
Average	85.40	85.41	86.25
Median	85	85	86
Maximum	885	540	459
Minimum	13	15	15

5.3.1 Training

The base variant of the mT5, which is a 580 million parameter model, was used for fine-tuning on SumeCzech dataset. Training was done using HFT Trainer. No optimization techniques such as PEFT (as discussed in Section 5.1.1.1) were used. Each article text and abstract were first tokenized using the mT5 tokenizer and then truncated to 512 tokens.

The optimizer used was AdamW, the learning rate was set to 0.001, weight decay was set to 0.01, and batch size was set to 8. Initial attempts at bigger batch sizes or longer sequences led to out of memory (OOM) errors. The training was conducted over 8 epochs, totalling 867500 steps, on a single NVIDIA A40 GPU. The training duration was 168 hours. Additionally, a linear scheduler was employed to adjust the learning rate over time.

Further Training. Unlike Mistral 7B in Section 5.4, we deemed further training on the dataset created for evaluation as fruitless due to the 512 token context length available on mT5 being far exceeded by the amount of tokens in the dataset's pages.

5.4 Mistral 7B

The task of fine-tuning an LLM requires significant computational resources. The fine-tuning of the base variant of mT5 with only 580M parameters led to near

full capacity usage of the 45 GB VRAM available in the NVIDIA A40 GPU. In the case of Mistral 7B, a normal fine-tuning would be impossible on the given GPU. To train such a large model, we had to either reserve more GPUs or use various optimizations. Due to time constraints that came with reserving more GPUs from MetaCentrum's[12] facilities, we opted to use the latter option.

5.4.1 Optimization

The optimization methods or libraries that were used during fine-tuning of Mistral 7B are briefly described in this section.

5.4.1.1 Unsloth

Unsloth [40] is a library that is designed to make fine-tuning LLms faster and more memory efficient. This is achieved through various optimizations like manual derivation of backpropagation steps and the use of OpenAI's Triton[13] language for kernel rewriting. Developed by Daniel Han, Michael Han, and the open-source community, it is fully compatible with the Hugging Face ecosystem which includes the Hugging Face Transformer library, therefore usage with HFT Trainer is possible. As of this writing, the amount of supported architectures is limited. From the models described in Chapter 4, only Mistral architecture models are supported. Fine-tuning with multiple GPUs is currently not supported.

5.4.1.2 Quantization

Quantization [41] is a technique used to reduce the computational cost and model size of neural networks by approximating the weights and activations with lower bit representations. Common quantizations are `float32` to `float16` and `float32` to `int8` (8-bit integer). There are various types of quantization methods [42] such as ***Post-Training Quantization*** (PTQ) or ***Quantization-Aware Training*** (QAT).

PTQ calibrates the pre-trained model first and then it applies quantization to the pre-trained model. The model is calibrated using a subset of the training data to determine the optimal way to map floating-point numbers to a lower-bit representation. However, PTQ might lead to a larger drop in accuracy when compared to QAT.

With QAT the pre-trained model is quantized and then fine-tuned using a subset of the training data. The weights are then updated in a way that the model learns how to perform well even with quantized values. QAT has usually higher accuracy than PTQ but requires more time due to fine-tuning.

[12] `https://metavo.metacentrum.cz/`
[13] `https://github.com/openai/triton`

5.4.1.3 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) [43] is a method to adapt large LLMs to specific tasks (such as text summarization) in a more parameter-efficient way. The key idea behind LoRA is to freeze the pre-trained model weights and introduce trainable low-rank matrices that modify the behavior of the model. LoRA works by injecting these low-rank matrices into specific layers of the Transformer architecture. The layers are selected based on several factors, such as the downstream task or model's architecture. The matrices then alter the output of the layers which they are injected into. This results in a much lower memory requirement and a reduction in the computational resources needed for adaptation in comparison to full fine-tuning, where all parameters are adjusted while achieving similar performance.

5.4.1.4 QLoRA

QLoRA [44] is a fine-tuning method that significantly reduces the memory needed to fine-tune large language models by combining quantization and LoRA together. It first quantizes the pre-trained language model to 4-bit using a novel method. Then it introduces LoRA into the quantized model. During fine-tuning, the backpropagation only tunes the low-rank matrices. QLoRA uses bitsandbytes[14] for quantization and it is fully supported by the Hugging Face Transformers library as a PEFT method.

5.4.2 Training

We encountered some problems during the initial installation and usage of the Unsloth library. The library uses `ldconfig`[15] command to create the necessary links for the CUDA library. However, this command accesses and writes into the `/etc/` directory which requires root permissions. The environment in which the training was done does not grant root permissions by default. To avoid using root permissions, we set the environment variable `TRITON_LIBCUDA_PATH` to `/usr/local/cuda/compat`, which is a directory where `libcuda.so` file resided. We also set the environment variable `LIBRARY_PATH` to `/usr/local/cuda/lib64`, which in our case is a path where libraries necessary for the development and execution of CUDA applications are located.

To fine-tune Mistral 7B, we utilized resources from publicly available fine-tuning notebooks[16,17] and a 4-bit pre-quantized Mistral 7B model[18] from Hugging Face platform, both provided by the developers of Unsloth library. In contrast to the

[14] `https://github.com/TimDettmers/bitsandbytes`
[15] `https://man7.org/linux/man-pages/man8/ldconfig.8.html`
[16] `https://jupyter.org/`
[17] `https://colab.google/`
[18] `https://huggingface.co/unsloth/mistral-7b-bnb-4bit`

utilization of HFT Trainer for the mT5 model fine-tuning, we use SFTTrainer[19] from TRL [45] library, which serves as a wrapper for HFT Trainer.

The model was fine-tuned on the SumeCzech dataset for one epoch with a total batch size of 32, therefore the training took 27112 steps in total. The whole process took 400 hours and the model was fine-tuned using 1x NVIDIA A40 45GB. Initially, a broader training scope beyond one epoch was considered. However, such training would have required more time than was available for this thesis.

5.4.2.1 Dataset Processing

To pass the SumeCzech dataset into the SFTTrainer, we needed to format the dataset first. We formatted the dataset using the format used in the Stanford Alpaca project [46][47][48]. Each dataset entry had these columns after formatting:

- **input:** The full text of the article.
- **output:** The summary of the article.
- **instruction:** The instruction for the model. The instruction was set to `Summarize the following text:` for all entries.
- **text:** The combination of input, output, and instruction in a structured format. See 5.3 for an example. Each text had to be appended with an end-of-sequence (EOS) token <s>.

Source code 5.3: Example of `text` column in formatted SumeCzech dataset entry. The input and response were truncated and appended with `...` due to their long length

```
Below is an instruction that describes a task, paired with an
    input that provides further context. Write a response
   that appropriately completes the request.

### Instruction:
Summarize the following text:

### Input:
Kdy jste slyšela jako cizinka slovo Ostrava poprvé? O Ostravě
    jsem poprvé slyšela v Českých Budějovicích, byla jsem tam
    asi rok v angažmá...

### Response:
Pračiková žije v Ostravě už téměř dvacet let... <s>
```

[19] https://huggingface.co/docs/trl/sft_trainer

5.4.2.2 Parameters

The training utilizes ***Mixed Precision Training*** [49] method, where some variables are stored at half-precision (`float16` or `Bfloat16` [50] instead of the standard full-precision `float32`). This does make computations faster, although it does not reduce the memory requirements. We specifically use `Bfloat16`. The max sequence length was set to 8192. The batch size was set to 8 with gradient accumulation steps set to 4, which makes an effective batch size of 32. The optimizer used was 8-bit AdamW with 0.01 weight decay along with a linear learning rate scheduler. For ***QLoRA*** parameters, we used rank at 16 and alpha at 16. The targeted layers were `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj` and `down_proj`. Warmup steps were set to 100.

5.4.3 Further Training

After the initial training on the SumeCzech dataset was complete, we employed further training on the model using the dataset described in Section 5.2 due to unsatisfactory performance. We further elaborate on the performance in Chapter 6.

The evaluation dataset was divided into a non-shuffled 75-25 ratio for the train and test splits, yielding 324 page summaries for the training set and 108 for the testing set. Subsequently, the model underwent an additional 16 epochs of training on the training split.

A validation set was not constructed for several reasons. Initially, an issue was encountered where the validation loss consistently returned a Not a Number (NaN) value. This issue had also appeared during the initial training phase; however, it was deemed less critical then due to the one epoch training duration, which minimized the risk of overfitting [51]. Furthermore, given the constrained size of our dataset, it was important to maintain a test set of adequate size to ensure the quality of the evaluation.

The absence of validation loss meant we lacked a method to identify overfitting. To overcome this, we implemented a strategy of saving checkpoints after each training epoch. We then assessed the performance of these checkpoints on the test set using $\text{ROUGE}_{\text{RAW}}$ metrics, selecting the epoch that demonstrated the best performance. Details of this process are further discussed in Chapter 6.

Total summaries were not utilized in this phase of training due to the complete issue text length exceeding the model's maximum context length capacity.

5.4.3.1 Parameters

The training parameters were largely maintained as they were during the initial phase, with a few modifications: the batch size was adjusted to 8, and gradient

accumulation steps were set at 1, effectively setting the batch size at 8. Warmup steps were calculated as 10% of the total training steps; with 16 epochs leading to a total of 640 steps, this resulted in 64 warmup steps.

5.5 TST

In the method study described in Chapter 4, Section 4.3, we implemented the Translation-Summarization-Translation (TST) method. For high-quality translation, the ALMA-R 13B variant was utilized, while English text summarization was performed using the 4-bit pre-quantized, instruct fine-tuned variant of Mistral 7B[20], made available by the Unsloth library developers on the Hugging Face platform. The ALMA-R 13B model, in its non-quantized form, was downloaded and subsequently quantized using the bitsandbytes[21] library.

During the translation, we encountered challenges associated with the ALMA-R 13B model. Specifically, the model's maximum context length of 4096 tokens and a noticeable drop in translation quality for long texts were the biggest concerns. This decline in performance can be attributed to the model being fine-tuned on datasets with a maximum of 512 tokens, as detailed in [31]. To prevent this, texts were divided into segments of 10 sentences each for translation. These segments were then translated independently and reassembled to produce the complete translation.

The translation quality however varied with the number of sentences per segment, affecting the quality of subsequent summarization. Specifically, larger segments sometimes resulted in the model's outputs repeating its generated sequences, while smaller ones, though less repetitive, provided insufficient context which negatively affected the translation quality. We achieved best results with 10-sentence segments, using inference settings of `temperature` and `top_p` set to 1, and `repetition_penalty` set to 1.3.

For summarization, a specific template supported by the model's tokenizer was used, as shown in an example 5.4.

Source code 5.4: Summarization template for TST

```
[{
        "role": "user",
        "content": "Summarize my texts using only 5 sentences
    "
    },
    {
        "role": "assistant",
        "content": "Sure. I will write summaries in the style
    of a news reporter and use only 5 sentences."
```

[20] https://huggingface.co/unsloth/mistral-7b-instruct-v0.2-bnb-4bit
[21] https://github.com/TimDettmers/bitsandbytes

```
    },
    {
        "role": "user",
        "content": "The text to summarize..."
}]
```

This template was processed using the tokenizer's `apply_chat_template` function, which converts the list into a formatted string prompt. This prompt was immediately tokenized because the parameter `tokenize` is to `true` by default. The specific template as shown in 5.4 was used because the model had problems with following instructions, usually generating more than 5 sentences or summarizing text in different styles. The inference settings were also adjusted to a `temperature` of 0.3 and `top_p` to 1 for higher adherence to instructions. The model also had a tendency to enumerate sentences, therefore regular expressions were used to remove them. The process for translating the English summaries back to Czech language followed the same procedure as the initial translation process with the target and source language swapped.

Evaluation 6

This chapter focuses on evaluating the methods implemented, as detailed in Chapter 5. We will assess the performance of both mT5 and Mistral 7B, which were trained on the SumeCzech dataset. Moreover, these models will be evaluated on the SumeCzech test set to gain a comprehensive understanding of their capabilities. Evaluation will be conducted on a dataset created in Chapter 5, hereafter referred to as the POC (Posel od Čerchova) dataset for conciseness. The POC dataset comprises two types of summaries: issue summaries (POC-I) and page summaries (POC-P). Additionally, we will outline the evaluation settings employed. If specific inference settings were not mentioned, it implies that the default settings were utilized. These defaults are detailed in the Hugging Face documentation[1]. All evaluations in this Chapter will be done using the $\text{ROUGE}_{\text{RAW}}$ metric.

For ease of reference, we will be using various abbreviations in this chapter. A list of these abbreviations is provided in Table 6.1.

In the upcoming sections, we will compare the performance of these models. In the comparison tables, a value highlighted in **bold** will denote the highest metric value obtained.

Table 6.1: List of abbreviations

Abbreviation	Description
POC	Posel od Čerchova dataset
POC-I	Issue summaries from the POC dataset
POC-P	Page summaries from the POC dataset
M7B-SC	Mistral 7B model trained on the SumeCzech dataset
M7B-POC	M7B-SC further trained on the POC dataset
mT5-SC	mT5 model trained on the SumeCzech dataset
TST	Translate-Summarize-Translate method

[1] `https://huggingface.co/docs/transformers/main_classes/text_generation#transformers.GenerationConfig`

6.1 Performance of SumeCzech Trained Models on SumeCzech Test Set

This section examines the performance metrics of the mT5-SC and M7B-SC models on the SumeCzech dataset. The results can be seen in Table 6.2. Additionally, this evaluation includes comparative analyses with methodologies utilized by the creators of SumeCzech and a fine-tuned mBART model, referred to as HT2A-S [52]. The HT2A-S model, developed by Marian Krotil in their Bachelor's thesis, was trained using a slightly different methodology; the article headline was included with the text during abstract generation. However, the author also included an evaluation of HT2A-S on text-to-abstract generation excluding the headline incorporation, which is the evaluation presented in Table 6.2.

Extractive methods, evaluated by SumeCzech authors, like Textrank (as described in 4.1.2), and simplistic strategies such as selecting the first few sentences or random sentences for summaries, are also compared. An abstractive text summarization model developed using the tensor2tensor [53] framework by the SumeCzech authors is evaluated as well. Among all the various methods, M7B-SC demonstrates superior performance across all listed evaluation metrics.

Table 6.2: Results of various methods on SumeCzech test set

Method	$ROUGE_{raw}$-1			$ROUGE_{raw}$-2			$ROUGE_{raw}$-L		
	P	R	F	P	R	F	P	R	F
M7B-SC	**24.4**	**19.7**	**21.2**	**6.5**	**5.3**	**5.7**	**17.8**	**14.5**	**15.5**
mT5-SC	22.0	17.9	19.2	5.3	4.3	4.6	16.1	13.2	14.1
HT2A-S	22.9	16.0	18.2	5.7	4.0	4.6	16.9	11.9	13.5
first	13.1	17.9	14.4	0.1	9.8	0.2	1.1	8.8	0.9
random	11.7	15.5	12.7	0.1	2.0	0.1	0.7	10.3	0.8
textrank	11.1	20.8	13.8	0.1	6.0	0.3	0.7	13.4	0.8
tensor2tensor	13.2	10.5	11.3	0.1	2.0	0.1	0.2	8.1	0.8

6.2 Evaluating M7B-SC and M7B-POC on the POC-P Test Set

As discussed in Chapter 5, specifically in Section 5.4.3, the performance of M7B-SC on the POC-P dataset was deemed unsatisfactory. To address this, we conducted further training and monitored the model's progress by saving a checkpoint at the end of each epoch, yielding a total of 16 checkpoints. These checkpoints allowed us

to evaluate the model's performance on the POC-P test set at each epoch, with the results presented in Table 6.3.

Our analysis revealed that M7B-SC's performance on the POC-P test set was significantly lower than M7B-POC's performance from the very first epoch of further training. Notably, the model demonstrated its highest overall performance at the third epoch. Based on this finding, we selected the third epoch checkpoint to represent the M7B-POC model for subsequent evaluations and comparisons.

Table 6.3: Comparison of M7B-POC's performance at various epochs on POC-P test set. Epoch 0 represents the performance of the M7B-SC model.

Epoch	$ROUGE_{raw}$-1			$ROUGE_{raw}$-2			$ROUGE_{raw}$-L		
	P	R	F	P	R	F	P	R	F
0	19.9	5.1	7.1	2.9	0.8	1.1	15.2	3.9	5.4
1	23.6	15.7	18.1	3.6	2.4	2.8	**16.8**	11.2	12.8
2	**24.1**	16.2	18.9	4.2	2.8	3.3	16.4	11.1	12.9
3	23.3	17.4	19.5	**4.7**	**3.5**	**4.0**	16.4	12.3	**13.8**
4	22.8	17.5	19.4	4.1	3.1	3.5	15.3	11.8	13.1
5	22.1	20.1	**20.6**	3.7	3.5	3.5	14.5	13.0	13.4
6	22.2	19.3	20.4	3.9	3.4	3.6	14.6	12.7	13.4
7	22.1	18.8	19.9	3.7	3.1	3.3	14.4	12.3	13.0
8	21.2	**20.2**	20.4	3.6	3.3	3.4	13.7	**13.1**	13.2
9	21.0	19.5	19.9	3.6	3.3	3.4	13.3	12.3	12.6
10	21.4	19.5	20.0	3.3	3.0	3.1	13.8	12.5	12.9
11	20.6	18.6	19.1	3.4	3.1	3.1	13.7	12.3	12.6
12	20.1	19.7	19.5	3.4	3.3	3.2	13.2	13.0	12.8
13	20.0	19.1	19.3	3.0	2.9	2.9	13.0	12.5	12.6
14	20.7	19.9	19.9	3.3	3.3	3.2	13.2	12.8	12.7
15	20.8	19.5	19.8	3.4	3.2	3.2	13.5	12.6	12.8
16	21.1	19.4	19.7	3.3	3.0	3.1	13.6	12.4	12.6

6.3 Performance of Implemented Methods on POC-P and POC-I

The performance of M7B-POC, mT5, and TST can be seen in Table 6.4 for POC-P and Table 6.5 for POC-I. The asterisk (*) indicates that the model was evaluated on a subset of the POC-P test set, respectively the POC-I test set. Specifically, it was evaluated on 106 page summaries, respectively 25 issue summaries. Out of the original test set's 108 page summaries and 26 issue summaries, two page summaries and one issue were discarded because the two page summaries did not contain

enough information to form a complete issue summary. TST and mT5-SC were evaluated on all 432 page summaries inside POC-P.

The analysis indicates that M7B-POC exhibits better overall performance on both the POC-P and POC-I datasets. In contrast, the TST method achieved higher recall scores, likely due to generating longer summaries than M7B-POC, albeit with reduced precision. It's worth noting that M7B-POC's performance slightly varies from the results observed at the third epoch in 6.3, attributed to the exclusion of the two page summaries from the evaluation.

During the generation of issue summaries, M7B-POC encountered a specific problem where it produced summaries that appeared to be mere concatenations of page summaries. To address this, we adjusted the inference settings by setting the `temperature` to 0.6, `top_p` to 0.8, enabling `do_sample` as `true`, and adjusting the `repetition_penalty` to 1.1. These modifications resulted in the generation of more appropriate summaries, according to our observations.

Table 6.4: Results of implemented methods on POC-P. See Section 6.3 for more details.

Method	$ROUGE_{raw}$-1			$ROUGE_{raw}$-2			$ROUGE_{raw}$-L		
	P	R	F	P	R	F	P	R	F
M7B-POC*	**23.5**	17.4	19.6	**4.8**	3.5	**4.0**	**16.6**	12.2	**13.8**
TST	17.2	**25.1**	**19.9**	2.5	**3.8**	2.9	11.3	**16.4**	13.0
mT5-SC	20.2	8.2	11.1	1.4	0.5	0.7	14.9	6.1	8.2

Table 6.5: Results of implemented methods on POC-I. See Section 6.3 for more details.

Method	$ROUGE_{raw}$-1			$ROUGE_{raw}$-2			$ROUGE_{raw}$-L		
	P	R	F	P	R	F	P	R	F
M7B-POC*	17.5	16.6	16.6	**2.6**	2.4	**2.4**	12.4	11.8	**11.8**
TST	13.8	**24.7**	**17.2**	1.7	**3.1**	2.1	8.9	**16.2**	11.2
mT5	**17.9**	5.8	8.4	1.0	0.3	0.4	**13.8**	4.4	6.4

Conclusion 7

In this Bachelor's thesis, we introduce three methods which can summarize historical documents: mT5-SC, M7B-POC and TST with mT5-SC, M7B-SC and M7B-POC available on the Hugging Face platform[1,2,3]. We created our own dataset of historical documents for text summarization abbreviated as POC. We then evaluated these three methods on POC using the $\text{ROUGE}_{\text{RAW}}$ metric, where M7B-POC achieved the highest overall performance.

For future improvements, we suggest the development of a more extensive and higher quality dataset of historical documents for evaluation and training. Our findings have shown that while the task of generating abstracts from modern news texts does not directly align with the task of abstract generation from historical documents, models trained on modern text sources can still acquire a substantial understanding of the language. Furthermore, additional training on even a small dataset of historical documents demonstrated significant improvements from the initial epoch.

The TST method presented a possibility for improvement through the incorporation of more advanced models as they become available. This approach could yield better performance over time as new and better translation or text summarization methods get released.

Additionally, the quality of text summarization could have been enhanced by avoiding quantization. However, the limitation of computational resources, especially the significant time constraints, made this approach impractical within our capabilities.

[1] `https://huggingface.co/tranv/mt5-base-finetuned-sumeczech`
[2] `https://huggingface.co/tranv/mistral7b-sumeczech-qlora`
[3] `https://huggingface.co/tranv/mistral7b-poc-qlora`

Acronyms 8

SOTA State-of-the-art

ETC Extended Transformer Construction

GQA Grouped-Query Attention

SWA Sliding Window Attention

NLP Natural language processing

LLM Large language model

OOM Out of memory

PEFT Parameter-Efficient Fine-Tuning

CPO Contrastive Preference Optimization

SFT Supervised fine-tuning

GSG Gap Sentences Generation

TF-IDF Term Frequency-Inverse Document Frequency

TF Term frequency

IDF Inverse document frequency

ReLU Rectified linear unit

HFT Hugging Face Transformers

NaN Not a Number

Claude 3 Opus Opus

TST Translation-Summarization-Translation

LCS Least common subsequence

Bibliography

1. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep learning*. MIT press, 2016.
2. STRAKA, Milan et al. 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), 2018. Available also from: https://www.aclweb.org/anthology/L18-1551.
3. LIN, Chin-Yew. ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 74–81. Available also from: https://aclanthology.org/W04-1013.
4. HERMANN, Karl Moritz et al. Teaching Machines to Read and Comprehend. In: *NIPS*. 2015, pp. 1693–1701. Available also from: http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.
5. SEE, Abigail; LIU, Peter J.; MANNING, Christopher D. Get To The Point: Summarization with Pointer-Generator Networks. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 1073–1083. Available from DOI: 10.18653/v1/P17-1099.
6. NARAYAN, Shashi; COHEN, Shay B.; LAPATA, Mirella. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, 2018.
7. COHAN, Arman et al. *A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents*. 2018. Available from arXiv: 1804.05685 [cs.CL].
8. HASAN, Tahmid et al. XL-Sum: Large-Scale Multilingual Abstractive Summarization for 44 Languages. In: ZONG, Chengqing; XIA, Fei; LI, Wenjie; NAVIGLI, Roberto (eds.). *Findings of the Association for Computational Linguistics:*

ACL-IJCNLP 2021. Online: Association for Computational Linguistics, 2021, pp. 4693–4703. Available from DOI: 10.18653/v1/2021.findings-acl.413.

9. SCIALOM, Thomas; DRAY, Paul-Alexis; LAMPRIER, Sylvain; PIWOWARSKI, Benjamin; STAIANO, Jacopo. *MLSUM: The Multilingual Summarization Corpus*. 2020. Available from arXiv: 2004.14900 [cs.CL].

10. KRYŚCIŃSKI, Wojciech; RAJANI, Nazneen; AGARWAL, Divyansh; XIONG, Caiming; RADEV, Dragomir. BookSum: A Collection of Datasets for Long-form Narrative Summarization. 2021. Available from arXiv: 2105.08209 [cs.CL].

11. MILLER, Derek. *Leveraging BERT for Extractive Text Summarization on Lectures*. 2019. Available from arXiv: 1906.04165 [cs.CL].

12. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. Available from arXiv: 1810.04805 [cs.CL].

13. JIN, Xin; HAN, Jiawei. K-Means Clustering. In: *Encyclopedia of Machine Learning*. Ed. by SAMMUT, Claude; WEBB, Geoffrey I. Boston, MA: Springer US, 2010, pp. 563–564. ISBN 978-0-387-30164-8. Available from DOI: 10.1007/978-0-387-30164-8_425.

14. MIHALCEA, Rada; TARAU, Paul. TextRank: Bringing Order into Text. In: LIN, Dekang; WU, Dekai (eds.). *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 404–411. Available also from: https://aclanthology.org/W04-3252.

15. PAGE, Lawrence; BRIN, Sergey; MOTWANI, Rajeev; WINOGRAD, Terry. The PageRank Citation Ranking : Bringing Order to the Web. In: *The Web Conference*. 1999. Available also from: https://api.semanticscholar.org/CorpusID:1508503.

16. CHRISTIAN, Hans; AGUS, Mikhael; SUHARTONO, Derwin. Single Document Automatic Text Summarization using Term Frequency-Inverse Document Frequency (TF-IDF). *ComTech: Computer, Mathematics and Engineering Applications*. 2016, vol. 7, p. 285. Available from DOI: 10.21512/comtech.v7i4.3746.

17. VASWANI, Ashish et al. *Attention Is All You Need*. 2023. Available from arXiv: 1706.03762 [cs.CL].

18. RAFFEL, Colin et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*. 2020, vol. 21, no. 140, pp. 1–67. Available also from: http://jmlr.org/papers/v21/20-074.html.

19. WANG, Alex et al. *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. 2020. Available from arXiv: 1905.00537 [cs.CL].

20. WANG, Alex et al. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In: LINZEN, Tal; CHRUPAŁA, Grzegorz; ALISHAHI, Afra (eds.). *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 353–355. Available from DOI: 10.18653/v1/W18-5446.

21. RAJPURKAR, Pranav; ZHANG, Jian; LOPYREV, Konstantin; LIANG, Percy. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. Available from arXiv: 1606.05250 [cs.CL].

22. ZHANG, Jingqing; ZHAO, Yao; SALEH, Mohammad; LIU, Peter J. *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*. 2019. Available from arXiv: 1912.08777 [cs.CL].

23. LEWIS, Mike et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. Available from arXiv: 1910.13461 [cs.CL].

24. LIU, Yinhan et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. Available from arXiv: 1907.11692 [cs.CL].

25. XUE, Linting et al. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In: TOUTANOVA, Kristina et al. (eds.). *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, 2021, pp. 483–498. Available from DOI: 10.18653/v1/2021.naacl-main.41.

26. JIANG, Albert Q. et al. *Mistral 7B*. 2023. Available from arXiv: 2310.06825 [cs.CL].

27. AINSLIE, Joshua et al. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. Available from arXiv: 2305.13245 [cs.CL].

28. BELTAGY, Iz; PETERS, Matthew E.; COHAN, Arman. *Longformer: The Long-Document Transformer*. 2020. Available from arXiv: 2004.05150 [cs.CL].

29. GUO, Mandy et al. *LongT5: Efficient Text-To-Text Transformer for Long Sequences*. 2022. Available from arXiv: 2112.07916 [cs.CL].

30. AINSLIE, Joshua et al. *ETC: Encoding Long and Structured Inputs in Transformers*. 2020. Available from arXiv: 2004.08483 [cs.LG].

31. XU, Haoran et al. *Contrastive Preference Optimization: Pushing the Boundaries of LLM Performance in Machine Translation*. 2024. Available from arXiv: 2401.08417 [cs.CL].

32. OPENAI et al. *GPT-4 Technical Report*. 2024. Available from arXiv: 2303.08774 [cs.CL].

33. XU, Haoran; KIM, Young Jin; SHARAF, Amr; AWADALLA, Hany Hassan. *A Paradigm Shift in Machine Translation: Boosting Translation Performance of Large Language Models*. 2024. Available from arXiv: 2309.11674 [cs.CL].

34. KLUYVER, Thomas et al. Jupyter Notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (eds.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, 2016, pp. 87–90.

35. WOLF, Thomas et al. Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 38–45. Available also from: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

36. MANGRULKAR, Sourab et al. *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods* [https://github.com/huggingface/peft]. 2022.

37. KESKAR, Nitish Shirish; MCCANN, Bryan; VARSHNEY, Lav R.; XIONG, Caiming; SOCHER, Richard. *CTRL: A Conditional Transformer Language Model for Controllable Generation*. 2019. Available from arXiv: 1909.05858 [cs.CL].

38. *Bavorsko-česká síť digitálních historických pramenů* [online]. [visited on 2024-03-25]. Available from: https://www.portafontium.eu/.

39. ANTHROPIC. *The Claude 3 Model Family: Opus, Sonnet, Haiku*. 2024. Available also from: https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

40. HAN, Daniel; HAN, Michael. *2-5X faster 70% less memory QLoRA & LoRA finetuning* [https://github.com/unslothai/unsloth]. GitHub, 2023.

41. JACOB, Benoit et al. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. 2017. Available from arXiv: 1712.05877 [cs.LG].

42. GHOLAMI, Amir et al. *A Survey of Quantization Methods for Efficient Neural Network Inference*. 2021. Available from arXiv: 2103.13630 [cs.CV].

43. HU, Edward J. et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. Available from arXiv: 2106.09685 [cs.CL].

44. DETTMERS, Tim; PAGNONI, Artidoro; HOLTZMAN, Ari; ZETTLEMOYER, Luke. *QLoRA: Efficient Finetuning of Quantized LLMs*. 2023. Available from arXiv: 2305.14314 [cs.LG].

45. WERRA, Leandro von et al. *TRL: Transformer Reinforcement Learning* [https://github.com/huggingface/trl]. GitHub, 2020.

46. TAORI, Rohan et al. *Stanford Alpaca: An Instruction-following LLaMA model* [https://github.com/tatsu-lab/stanford_alpaca]. GitHub, 2023.

47. TOUVRON, Hugo et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. Available from arXiv: 2302.13971 [cs.CL].

48. WANG, Yizhong et al. *Self-Instruct: Aligning Language Models with Self-Generated Instructions*. 2023. Available from arXiv: 2212.10560 [cs.CL].

49. MICIKEVICIUS, Paulius et al. *Mixed Precision Training*. 2018. Available from arXiv: 1710.03740 [cs.AI].

50. KALAMKAR, Dhiraj et al. *A Study of BFLOAT16 for Deep Learning Training*. 2019. Available from arXiv: 1905.12322 [cs.LG].

51. KOMATSUZAKI, Aran. *One Epoch Is All You Need*. 2019. Available from arXiv: 1906.06669 [cs.LG].

52. KROTIL, Marian. *Text Summarization Methods in Czech*. 2022. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics. Supervised by Ph.D. ING. JAN DRCHAL.

53. VASWANI, Ashish et al. Tensor2Tensor for Neural Machine Translation. *CoRR*. 2018, vol. abs/1803.07416. Available also from: http://arxiv.org/abs/1803.07416.

List of Figures

List of Tables

List of Listings