BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DETECTION OF KEY INFORMATION IN EMERGENCY CALLS

DETEKCE KLÍČOVÝCH INFORMACÍ V HOVORECH NA TÍSŇOVÉ LINKY

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. MAREK SARVAŠ**
AUTOR PRÁCE

SUPERVISOR **Ing. PETR SCHWARZ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

156967

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Sarvaš Marek, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Machine Learning
Title:	**Detection of key information in emergency calls**
Category:	Speech and Natural Language Processing
Academic year:	2023/24

Assignment:

1. Study the type of information an emergency contact center agent collects from a caller and compile a list of entities to be detected automatically.
2. Study an emergency call dataset and records created by agents describing the calls. Prepare the data set in a form suitable for training and evaluating machine learning algorithms.
3. Learn about approaches for detecting named entities, especially an algorithm based on converting text to semantic vectors and classifying named entities from them, and an approach based on large language models. Choose the appropriate approach that is more suitable for the target task.
4. Train the algorithm on the prepared data set and evaluate its accuracy
5. Discuss the results achieved concerning practical use in an emergency contact center.

Literature:
According to the consultation with the supervisor.

Requirements for the semestral defence:
Points 1 to 3.

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

Supervisor:	**Schwarz Petr, Ing., Ph.D.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	17.5.2024
Approval date:	9.11.2023

Abstract

Emergency calls are usually made under extremely stressful conditions, where callers often provide crucial information rapidly, making it difficult for emergency line agents to capture all details accurately. This can result in repeated questions about information that was already provided and cause delays in response times from emergency services. This work aims to mitigate this problem and potentially speed up the response of emergency services by deploying a neural network models for information extraction, specifically targeting the Named Entity Recognition (NER) task. This work explores various Transformer-based approaches for NER task, such as pre-trained encoder-only, encoder-decoder (sequence-2-sequence) and Large Language Models. The best models achieved state-of-the-art results on publicly available Czech NER datasets. In addition, new NER datasets were created from available recordings of real emergency calls and the corresponding metadata. The models were trained and evaluated on the created datasets successfully achieving reasonable performance in name and location extraction.

Abstrakt

Tiesňové volania sa zvyčajne uskutočňujú v extrémne stresujúcich podmienkach, kde volajúci často poskytuje dôležité informácie rýchlo, čo sťažuje operátorom tiesňovej linky presne zachytiť všetky podrobnosti. To môže viesť k opakovaným otázkam o už poskytnutých informáciách a oneskoreniu reakcie pohotovostnej služby. Cieľom tejto práce je zmierniť tento problém a potenciálne urýchliť reakciu pohotovostných služieb nasadením neurónovej siete na extrakciu informácií, konkrétne so zameraním na úlohu Rozpoznávania pomenovaných entít (NER). Táto práca skúma rôzne prístupy založené na architektúre typu Transformers, ako sú predtrénované enkodér modely, enkodér-dekodér (sequence-2-sequence) a veľké jazykové modely. Vybrané modely dosiahli zatiaľ najlepšie výsledky na verejne dostupných českých NER datasetoch. Okrem toho boli vytvorené nové NER datasety z poskytnutých nahrávok skutočných tiesňových volaní a odpovedajúcich metadát. Predstavené modely boli natrénované a vyhodnotené na týchto novovytvorených datasetoch a úspešne dosiahli rozumné výsledky pre extrakciu mien a polohy.

Keywords

natural language processing, Named Entity Recognition, emergency line calls, Large Language Models

Klíčová slova

spracovanie prirodzeného jazyka, rozpoznávanie menných entít, hovory tiesňovej linky, veľké jazykové modely

Reference

SARVAŠ, Marek. *Detection of key information in emergency calls.* Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Petr Schwarz, Ph.D.

Rozšířený abstrakt

V dnešnom svete sa čoraz viac začína využívať umelá inteligencia v každodennom živote s cieľom pomôcť ľuďom alebo zlepšiť či automatizovať určité úlohy bez nutnosti ľudského zásahu. V posledných rokoch sú na vzostupe technológie v oblasti spracovania zvuku a prirodzeného jazyka, najmä od predstavenia veľký jazykových modelov (LLM), ktoré prekonávajú doterajšie modely v širokom spektre úloh.

Jednou z oblastí, v ktorej začína byť záujem o tieto technológie a kde by modely strojového učenia mohli mať zásadný prínos, je verejný sektor. Konkrétne sa zvyšuje záujem o využitie modelov strojového učenia v oblasti pohotovostných služieb ako napríklad záchranné služby či polícia. Takéto systémy by mohli byť nasadené do riešenia menších úloh ako extrakcia informácií, ktoré by slúžili ako pomocná ruka a využívali by sa v spolupráci s ľudskými operátormi alebo aj ako plne automatizovaný hlasový asistent.

Využívanie najmodernejších modelov strojového učenia v spojení s pohotovostnými službami má veľký potenciál, no vyvinúť takéto systémy nie je jednoduché z niekoľkých dôvodov. Dáta sú jednou z najdôležitejších vecí pre vybudovanie dobrého systému založeného na neurónových sieťach. Avšak, získať dáta práve z oblastí ako pohotovostné služby je veľmi obtiažne keďže sa jedná o citlivé údaje obsahujúce množstvo osobných informácií. Okrem toho sa ľudia, ktorí využívajú pohotovostné služby, často nachádzajú v nebezpečných alebo život ohrozujúcich situáciách, čo vytvára malý priestor na chyby.

Táto diplomová práca je súčasťou univerzitného projektu[1] sponzorovaného Ministerstvom vnútra Českej republiky pre operátorov tiesňových liniek v Českej republike. Motivácia tohto projektu spočíva v tom, že počas tiesňového volania je volajúci v strese a chce čo najrýchlejšie zdeliť všetky dôležité informácie, aby mu bola poskytnutá pomoc v čo najkratšom čase. Vo veľa prípadoch sa ale stáva, že volajúci poskytne veľa informácii v prvých sekundách hovoru a operátor tiesňovej linky nie je schopný ich všetky spracovať. Toto vedie k opätovnému pokladaniu otázok ohľadom mena volajúceho, jeho polohy či situácie v ktorej sa nachádza. Cieľom tejto práce je získať informácie, ktoré by pomohli agentovi a mohli skrátiť dĺžku hovoru a potenciálne urýchliť reakciu záchranných zložiek.

Navrhovaným riešením je vytvorenie neurónovej siete s architektúrou Transformer pre extrakciu dôležitých informácií z textových údajov. Prvotným cieľom je natrénovať tento model na získanie mien a polohy z tiesňových volaní. Typ extrahovaných informácií možno v budúcnosti rozšíriť na základe potrieb záchranných zložiek. Sekundárnym cieľom je vytvoriť štruktúrovaný dataset z poskytnutých nahrávok tiesňových volaní a ich metadát. Dataset bude použitý na dodatočné dotrénovanie modelov na dátach z tejto domény. Tento dataset bude tiež užitočný pre budúci vývoj nových modelov a je možné ho použiť na rozšírenie riešení prezentovaných v tejto práci, napríklad na extrahovanie kľúčových informácií priamo z reči.

Bolo experimentované s tromi prístupmi k tejto úlohe a pre každý prístup bol použitý iný typ modelu. Vo všetkých prípadoch ide o predtrénované modely s architektúrou Transformer, ktoré boli dotrénované pre úlohu extrakcie informácií. Prvým typom sú modely obsahujúce iba enkodér vrstvy a klasifikačnú vrstvu. Výstupom týchto modelov je typ entity pre každé slovo. Druhým typom sú tzv. sequence-to-sequence modely kde vstup je sekvencia slov a na výstupe je rovnaká sekvencia slov doplnená o označenia vybraných, pre nás dôležitých, informácií. Tretím typom sú veľké jazykové modely ktorých výstup je vo for-

[1] VK01020132 - Praktické ověření možnosti integrace umělé inteligence pro příjem tísňových volání pomocí hlasového chatbota, vyvinutého v rámci výzkumného projektu BV č. VI20192022169, s technologií pro příjem tísňové komunikace 112 a 150 v ČR (TCTV 112). Dostupné z: `https://starfos.tacr.cz/cs/projekty/VK01020132`

máte JSON. V tejto práci sa podarilo získať 0.89 F1 skóre na voľne dostupnej CoNLL verzii CNEC2.0 datasetu, čo predstavuje nové najlepšie F1 skóre na tomto datasete. Následne bol pomocou automatizovaných techník s manuálnou korekciou vytvorený nový dataset z poskytnutých nahrávok tiesňovej linky obsahujúci 56 000 slov a 3000 označených mien a polôh. Na tomto datasete bolo dosiahnuté 0.89 F1 skóre pre detekciu mien a 0.87 F1 skóre pre detekciu polohy danej udalosti.

Detection of key information in emergency calls

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Ing. Petr Schwarz, Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Marek Sarvaš
May 15, 2024

Acknowledgements

I would like to thank my supervisor Ing. Petr Schwarz, Ph.D for his valuable advice and guidance during this work. I would also like to thank all colleagues at BUT Speech@FIT group for providing insightful suggestions. This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

Contents

Chapter 1

Introduction

In today's world, Artificial Intelligence has started to be incorporated more and more into real-world applications to help people, improve some tasks, or automate things without human intervention in various scenarios in image, speech, or natural language processing fields. Especially with the rise of Large Language Models, new state-of-the-art results were achieved in a range of tasks and the models started to be utilized more in real-world scenarios.

One of the areas where machine learning models could bring crucial benefits is the public sector. Specifically, there is an increasing interest in the incorporation of machine learning models for the emergency services. From smaller tasks such as information extraction that would serve as a helping hand and would be used in collaboration with human operators to fully automated voice-bots. Using state-of-the-art machine learning models has great potential in collaboration with emergency services, however, it is not easy to develop such systems for several reasons. This area comes with a good portion of bureaucracy, as the data in this domain are difficult to obtain and very sensitive, containing a lot of personal information. In addition, people who use emergency services are often in dangerous or life-threatening situations, creating a small margin for error.

This thesis is part of a project[1] for emergency service call centers in the Czech Republic sponsored by the Ministry of Interior of the Czech Republic. During emergency calls, the person calling is often stressed and wants to get all the information out as soon as possible to receive help. This often happens in the first seconds of the call and the agent is not capable of processing all the information which leads the agent to ask questions that could have already been answered. This work aims to help extract information that would help the agent, potentially shortening emergency calls and speeding up the response of the emergency services.

The proposed solution is to create a Transformer-based neural network for the extraction of important information from textual data. The first goal is to extract names and locations from emergency calls. The type of information extracted can be extended based on emergency services needs. The secondary goal is to create a good labeled dataset from provided emergency call recordings and metadata that will be used for supervised fine-tuning of the models on in-domain data. This dataset could be very helpful in the future

[1] VK01020132 - Praktické ověření možnosti integrace umělé inteligence pro příjem tísňových volání pomocí hlasového chatbota, vyvinutého v rámci výzkumného projektu BV č. VI20192022169, s technologií pro příjem tísňové komunikace 112 a 150 v ČR (TCTV 112). Available at: `https://starfos.tacr.cz/cs/projekty/VK01020132`

development of new models and could be used to extend the solutions presented in this work to extract key information directly from speech.

Chapter 2 introduces neural networks and currently popular model architectures and training approaches. It presents the, Transformers architecture, Large Language Models and approaches to their training. Chapter 3 describes the Named Entity Recognition (NER) task and popular approaches to tackle this task, such as token classification or the sequence-to-sequence approach.

The following Chapter 4 lists publicly available Czech Named Entity Recognition datasets that will be used for baseline systems and the creation of the Czech emergency calls datasets. Chapter 5 presents an overview of the technologies and different models used in this work. It also compares the different approaches to this task based on the previous works presented in Chapter 3. Chapter 6 is divided into three main blocks based on the content of the experiments. The first part explores the proposed approaches for NER on publicly available datasets. The second part describes the process of creating the labels for a clean and usable emergency calls dataset (TCTV112 dataset). The third part contains inference and fine-tuning experiments on the new TCTV112 dataset along with a description of common mistakes made by the models and drawbacks of the available data. Chapter 7 concludes this work.

Chapter 2

Machine Learning

Machine learning experienced one of the biggest growths in the area of technical fields in recent years and has started to be widely applied in our day-to-day lives [24]. This chapter serves as an introduction to machine learning and neural networks, concentrating on deep learning architectures and concepts used throughout this thesis.

Machine learning is a subfield of Artificial Intelligence focusing on algorithms that are able to "learn" from the available data. In the case of machine learning algorithms, learning from an experience E is an improvement measured by P on a set of tasks T [18]. P stands for a quantitative performance measure, and it is a means of evaluating a given algorithm for a specific task or a set of tasks T. This evaluation enables the comparison of different algorithms. The measure is task-specific, for example, **accuracy** can be used to evaluate the classification algorithm, which is a proportion of correctly classified inputs to all inputs.

Goodfellow et al. [18] describe a task as a process of how the examples $\mathbf{x} \in \mathbb{R}^n$ are processed by the algorithm, where $\mathbf{x}$ are features such as pixel values in an image. There is a diverse number of tasks based on input feature processing and output produced by the ML algorithm [18], a few common examples:

Classification

The goal of classification is to assign a category from a set of $1, \ldots, k$ to each input. The algorithm is designed to produce a function $f : \mathbb{R}^n \to 1, \ldots, k$, where for $y = f(\mathbf{x})$, $\mathbf{x}$ is input feature vector and y is the assigned category. Classification can also have variants such as, the output of the algorithm is a probability distribution over the classes, or assigning multiple classes to a single input, also called multi-label classification. Another variant is described by Goodfellow et al. [18] as a **Classification with missing inputs**. In this case, some information is missing in the input, and the algorithm needs to learn a set of functions instead.

The main task of this thesis, the Named Entity Recognition, can also be approached as a classification problem where each word is classified as some entity type. Chapter 3 gives more details.

Transcription

According to Goodfellow et al. [18] description, transcription is a machine learning task where an algorithm takes some unstructured data as input and produces textual form for the information at the output. An example of this task, used in this thesis, is Automatic

Speech Recognition (ASR), which takes speech recording, for example, waveform, as input and produces its text transcription.

Machine Translation

In Machine Translation, the input of an algorithm is a sequence of symbols in one language, and a desired output is another sequence of symbols in a different language. Although not very common, this approach is also explored for Named Entity Recognition in Sections 3.2 and 6.1.

2.1 Neural Networks

Machine learning has a prominent role in today's world, advancing faster than ever, especially with the introduction of artificial neural networks (NN), formed from a simple perceptron [48] inspired by biological neurons to state-of-the-art deep neural network models that solve complex tasks in the fields of image, speech, and natural language processing. The effectiveness of deep neural networks (DNNs) lies in the ability to automatically extract hierarchical features from raw data, allowing the representation of intricate patterns and relations. As the name suggests, DNNs are composed of multiple different layers indicating the „depth" of the network, each consisting of many neurons. The structure of a neural network and the types of layers used vary according to the target task and the nature of the data [1].

Deep neural network training is an optimization process in which the goal is to adjust the weights and biases in the network to minimize the error between the output of the network and the desired target data [18]. This error is referred to as the network loss and is computed between the network output and the targets by a loss function. The loss function is usually selected based on the solved problem and is a crucial part of the training process, leading to model performance. Such loss function is, for example, Cross-Entropy loss [3]

$$L(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w}) \ , \tag{2.1}$$

which measures the error between two probability vectors and will be used in this thesis. In this equation, $\mathbf{x_n}$ represents an input vector, where $n = 1, \ldots, N$, N is the number of vectors and $t_k \in \{0, 1\}$ represents the element of the corresponding target vector. The loss is computed over K classes, y_k is the output of a network, and $\mathbf{w}$ are weights to be learned. After a forward pass through the model and a loss computation, the weights are updated using a backpropagation algorithm based on the gradient that is acquired for each layer, applying the chain rule of differential calculus [1, 18].

As machine learning algorithms learn from the data, they can be categorized based on the type of data provided during training into 3 categories [18, 3]:

- In **Supervised learning**, the datasets consist of both input data for the algorithm to learn from and corresponding labels for each training data point. The goal is to learn a mapping between these two.
- **Unsupervised learning** is the opposite, where the dataset does not contain the corresponding labels. In this case, the goal is to discover data with similar features and group them together based on these features, learn the probability distributions within the input data, or project the input to lower dimensionality.

- The above categories have a fixed-sized dataset and examples of optimal output during the training. In **Reinforcement learning** the algorithm interacts with the environment and has to discover the optimal result by itself through trial and error to maximize the reward.

2.1.1 Learning challenges

The goal of machine learning is to generalize to real-world scenarios for which they were designed, and to perform well not only on selected data. In addition to the challenge of generalization, some tasks, such as speech or natural language processing, become exponentially difficult, since they usually have high-dimensional features that model complex relations. This motivated research and the design of more complex algorithms, such as deep neural networks [18].

Neural networks are highly dependent on the data on which they are trained, and sometimes there is not enough data for the specific task that is being solved. The lack of data often occurs in supervised learning tasks, where it is demanding to acquire the labels. Machine translation for some languages can be an example in a scenario where the model is trained to translate from one language to another and we do not have enough `language1` - `language2` pairs. There are techniques to tackle these problems and/or boost the performance by leveraging unlabeled data or data available for different but similar tasks, such as:

- **Self-supervised learning** (SSL) is a machine learning paradigm where the model is trained on unlabeled data and the supervision signal is generated by the model itself. In Natural Language Processing, an objective for SSL can be masking some inputs and forcing the model to learn these masked inputs [14], or generating the input from the model's hidden intermediate representation [10]. Utilizing this approach, models can be first trained on a large amount of unlabeled data followed by training on a specific task with limited labeled data [14, 10, 35]. All models presented in Chapter 5 use SSL during pre-training.

- **Multi-task learning** (MTL) led to improved performance in various machine learning tasks by leveraging the information available in similar tasks. MTL can be defined as improving the performance of a model on task T_i using information from m tasks $\{T_i\}_{i=1}^{m}$ [60]. Multiple learning approaches take advantage of similarities between tasks and can be considered as some variations of MTL [49]. In MTL the goal is to obtain predictions for multiple tasks at once. However, in many works, a model benefits from learning **auxiliary tasks** during training. **Auxiliary task**, or auxiliary loss, is an objective that is used during training in order to improve the performance or generalization of the main tasks [49]. The use of multi-task learning in this work is described in Section 5.3.

2.2 Deep Neural Networks in Natural Language Processing

Natural Language Processing is a subfield of artificial intelligence that contains tasks such as natural language generation, machine translation, question answering, text generation, etc. The main goal is to train models that are able to understand (to a certain extent) or generate text in human language that is meaningful or relevant to the task performed. Algorithms used for these tasks are usually trained by using large text corpora from various sources

such as proprietary materials, public datasets, web crawls, etc. Several breakthroughs have been made in the NLP field in recent years, with the introduction of new deep learning architectures and approaches that allowed better representations of natural language on a word, sentence, or higher level. For example, the works [38] and [15] present approaches for learning word or sentence representation in vector space. Recurrent neural networks (RNNs) were the go-to models not only for NLP tasks, where the goal was to process and usually also to generate sequences with variable lengths. RNNs achieved state-of-the-art results in tasks such as machine translation, language modeling, or named entity recognition (further described in Chapter 3). However, RNNs have some drawbacks, and in 2017 Vaswani et al. [57] introduced a new architecture called **Transformers** that achieved new state-of-the-art results. Since then, multiple variations of the Transformer architecture have been introduced and used across the NLP tasks together with new training approaches.

2.2.1 Language representations

Even though deep neural networks showed great ability to perform tasks in the NLP field, they cannot work with text directly and need pre-processing to transform the text into input data that can be further processed by the neural models. This process is called tokenization, where the text is broken down into smaller units (tokens) and transformed into a numerical representation. One token can represent various parts of text based on the tokenization technique. The simplest tokenizer can work on the word level, where an integer number is assigned to each word. The drawback of this method is the closed vocabulary. This may cause problems in some NLP tasks such as machine translation, which is an open vocabulary problem. Sennrich et al. [51] introduced the byte pair encoding (BPE) technique for textual data, based on the BPE compression technique, where frequent characters or sequences are iteratively merged. BPE can therefore represent the out-of-vocabulary words by sequence of the sub-words that are present in the vocabulary.

Next, tokenized text data goes through a trainable embedding layer that maps the tokens into a vector of chosen size followed by the rest of the network. A model can learn different representations depending on its architecture and the objectives for which it is trained, as introduced in this chapter. Some models are explicitly trained to learn the intermediate complex language representation in a high-dimensional subspace and produce these representations that can later be used in a chosen downstream task.

Neural network pre-training for language representations is an effective way of improving performance on various NLP tasks [14]. Devlin et al. [14] introduced a method to learn language representation while taking into account context from both directions. Nowadays, this technique is very popular for transformer-based models (Section 2.4) that are pre-trained in an unsupervised fashion on large amounts of textual data, achieving SOTA results when applied on down-stream tasks.

2.2.2 Recurrent Neural Networks

RNNs are not used in any of the experiments in this thesis; however, this architecture was used in a previous work that will be compared with the proposed approaches. Recurrent neural network [18, 21, 42] is a type of NN that can process sequential data with variable length where the length of the output does not have to match the length of the input. This processing of sequential data and learning long-term dependencies is done by preserving the hidden state at each timestamp and using the hidden state from the previous step in the computation at the current step. Although RNNs achieved state-of-the-art results in

various NLP tasks, they have drawbacks that sometimes make them difficult to train. Some of the widely addressed problems [42] are the vanishing gradient [42], exploding gradient, or slow training. The former two problems can be mitigated by various approaches, such as introducing variations to original RNNs, for example, Gated Recurrent Unit [7] and Long-Short Term Memory [21] networks. Other approaches proposed by Pascanu et al. [42] are gradient norm clipping addressing the exploding gradient problem and a soft constraint to mitigate the vanishing gradient problem.

2.3 Transformers

Transformer is a deep neural network architecture based entirely on the attention mechanism proposed by Vaswani et al. [57]. This section describes how the model works, all the information used is gathered from the "Attention Is All You Need" [57] paper. The transformer model was originally implemented as an encoder-decoder and demonstrated on a machine translation that can be characterized as a sequence-to-sequence modeling task. The main idea behind this architecture was to overcome the limitations of previously mentioned recurrent neural networks, which were state-of-the-art approaches used in sequence modeling tasks such as machine translation. The attention mechanism reduces the sequential computation between the input and output symbols present in RNNs. This effectively creates room for a higher degree of parallelization. The original Transformer model is shown in Figure 2.1.

The input of the Transformer is tokenized text that is further passed through the trainable embedding layer to represent each token with a vector. Such vectors go through a stack of N encoder layers, producing an intermediate representation of the input sentence, further used in cross-attention in the decoder part of the model. In the case of machine translation, the target sentence is the input sentence translated to the target language. During training, the target sentence is tokenized, passes through the embedding layer, and is used as input to the decoder. However, unlike the input sentence in the encoder, the target is masked and shifted in a way that the model cannot attend to tokens in the future.

2.3.1 Multi-head Attention

Attention is described as a function mapping query and a key-value set to an output. In Transformers, scaled dot-product attention is used, where queries, keys, and values are the input of the attention, with dimensions d_k, d_k, and d_v, respectively. However, in a real implementation, these computations are performed simultaneously for a set of queries, keys, and values in the form of matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. The size of these matrices are $d_k \times d_{model}$, $d_k \times d_{model}$, $d_v \times d_{model}$, where d_{model} is size of an embedding vector;

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \ . \tag{2.2}$$

The paper proposing transformer architecture [57] further improves self-attention by introducing a multi-head attention mechanism, depicted in Figure 2.2. The multi-head attention allows the model to attend to information from different sub-space representations in different positions and is computed as

$$MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Concat(head_1, ..., head_h)\mathbf{W}^O \tag{2.3}$$

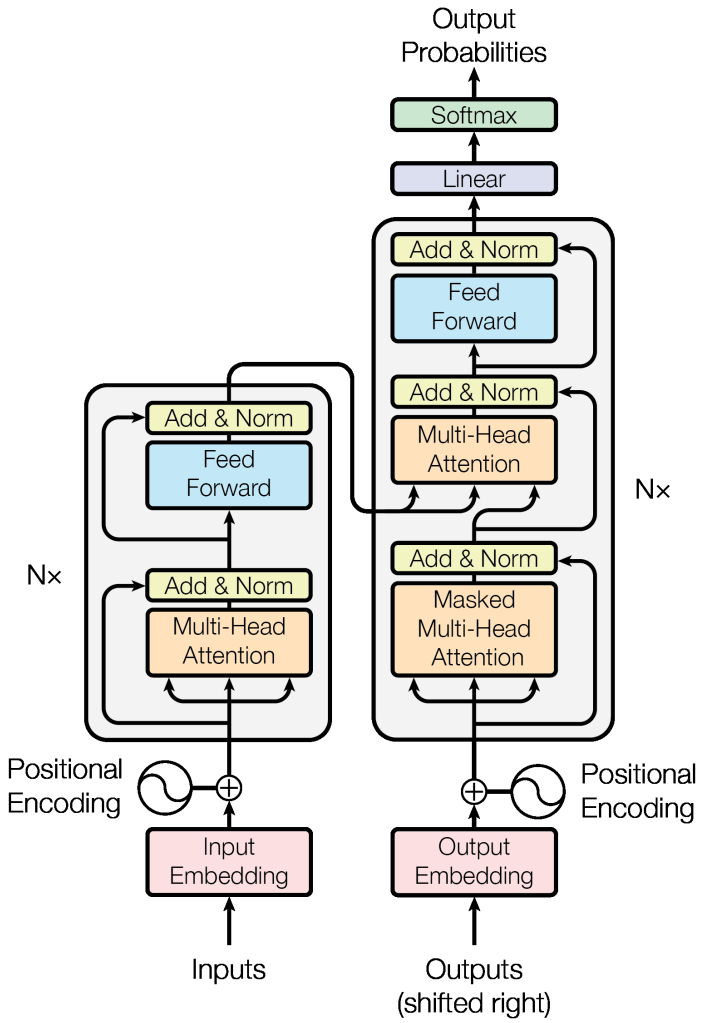

Figure 2.1: Encoder-Decoder architecture of the original Transformer. Adopted from [57].

where

$$head_i = Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \tag{2.4}$$

and the $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_k}$ and $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$ are projection matrices. In the transformer model, we can see that the multi-head attention is placed in three different places.

1. Inside the encoder, where all three keys, values, and queries in self-attention are computed from the encoder inputs.

2. Masked multi-head attention inside the decoder where keys, values, and queries are computed from the decoder input and masked to not attend to tokens in the future.

3. Encoder-decoder attention (also called cross-attention), where keys and values are computed from encoder output, while queries are obtained from the previous decoder layer.

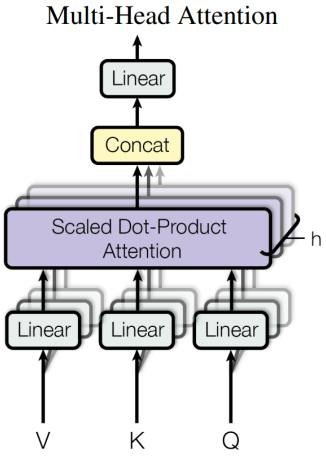

Figure 2.2: Multi-head attention mechanism. Adopted from [57].

2.3.2 Positional Encoding

The attention mechanism in the original Transformer architecture computes the representation of a sequence by relating to different positions at the same time with a constant number of operations. It is beneficial to encode the position of each token into the input embeddings. In this case, the authors used an absolute positional encoding. Positional encoding inserts crucial information about the token order in the sequence [57] into the model. In the original paper [57], fixed positional embeddings were used, computed with the sine and cosine functions as:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/1000^{2i/d_{model}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/1000^{2i/d_{model}}\right), \end{aligned} \tag{2.5}$$

where pos is a position and i represents a dimension. Shaw et al. [53] introduce another approach on how to encode the positions of words in the sentence called relative positional encoding.

2.4 Transformer architecture variations

In the original paper presenting the Transformer architecture [57], the model could be divided into two parts, encoder and decoder. The encoder part takes a source text as input, and, through self-attention and additional layers, it produces an intermediate representations of the input tokens. The decoder part combines the output of the encoder and the masked target sequences [57]. This masking ensures that the model has information only from previous tokens when predicting a token in position i.

Encoder-only models

In this case, as the name suggests, the models consist only of the encoder transformer. These models are usually pre-trained on large amounts of text data in a self-supervised fashion, where the input text is altered and the model is trained to reconstruct the original text. An example of such a training objective is Masked Language Modeling (MLM) shown in Figure 2.3. This approach was used to train the BERT [14] and subsequent BERT-like

models, where during training, a percentage of input tokens were masked and the model was tasked to predict the original tokens. Encoder-only models learn to produce word or sentence representations during the pre-training and are usually fine-tuned with additional layers to perform specific tasks such as sentence classification, entity recognition (also called token classification), part of speech tagging, etc. Other models in this encoder-only family are RoBERTa [36], LABSE [15], other variations of BERT [14] such as Albert [30], or multilingual approaches such as XLM [29].

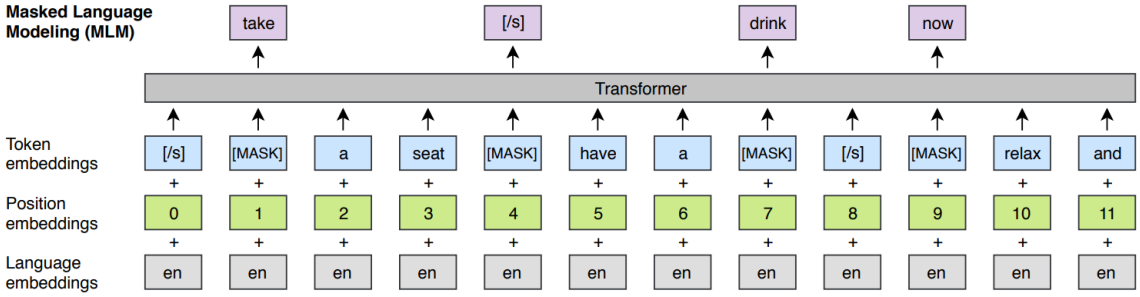

Figure 2.3: Example of Masked Language Modeling (MLM). Adopted from [29].

Decoder-only models

Decoder-only models are based on the decoder part of the original Transformers paper [57]. During training and inference, these models can access only tokens before the current one for every token, unlike encoder-only models. They are again pre-trained on large amounts of text in a self-supervised way. By learning the probability distribution of words, they can predict the next word based on the previous words in a sequence. These models, often called `auto-regressive` models, are good for text generation tasks. Examples of these models are nowadays popular Large Language Models such as GPT [59], GPT-2 [44], GPT-3 [4], LLaMa [55], LLaMa 2 [56], or Tranformer XL [11].

Encoder-Decoder models

Encoder-Decoder models are also called sequence-to-sequence models as both the input and the output are sequences. These models are used for tasks that involve generating a text output that depends on the input. Usually, there is a complex relation between the input and target sequences. The application tasks can be machine translation, summarization, automatic speech recognition, or speech-to-text translation that also involves audio input, not only text. This architecture can utilize training techniques used in encoder-only or decoder-only models; however, it usually involves more complex approaches. An example of encoder-decoder models in NLP is T5 [45], its multilingual variant mT5 [58], or BART [31].

2.5 Large Language Models

Large language models have become a target of research in the Artificial Intelligence field in recent years, mainly in its Natural Language Processing sub-field, reaching state-of-the-art results in various tasks. Nowadays, the term “Large Language Model” refers to the type of model based on the Transformer architecture that is trained on vast amounts of

textual data in an unsupervised or semi-supervised way. The "large" in the name refers to the amount of training data used and the size of these models, which, in the case of the state-of-the-art models, ranges in billions of parameters. Because of this, the models are able to learn complex patterns and structures in the natural language. This allows them to achieve state-of-the-art results in various natural language processing tasks. As mentioned in Section 2.4, examples of such models are GPT [59], GPT-2 [44], GPT-3 [4], PaLM [6], LLaMa [55], LLaMa2 [56], etc.

These models are trained in an auto-regressive way to generate text or answer either with or without some additional information (prompt) by predicting the most probable next tokens. The core of this approach [44] is to estimate a distribution of examples $(x_1, x_2, \dots x_n)$ where each example is composed of sequences $(s_1, s_2, \dots, s_n)$ with variable length which can be factorized into

$$p(x) = \prod_{i=1}^{n} p(s_n | s_1, \dots, s_{n-1}). \tag{2.6}$$

The NLP field shifted from designing task-specific model architectures into task-agnostic models pre-trained on large quantities of data combined with task-specific fine-tuning achieving state-of-the-art results in several NLP tasks. However, authors of GPT-2 and GPT-3 suggested that the Large Language Models can be pushed even further. By scaling the amount of training data and size of the models, they are able to perform reasonably well in zero-shot or few-shot settings. Zero-shot refers to the evaluation of a model on a task it was not trained on. When the model is trained on small amount of data for specific task or several examples are included in the prompt it is referred to as few-shot. These models are usually developed by research teams in big tech companies such as Open AI or Google, and trained with proprietary data, Meta released LLaMa and later LLaMa 2, LLama 3 models that achieved better results than GPT-3 while being trained on publicly available data.

2.5.1 Low-Rank Adaptation of LLMs

With the rising popularity of Large Language models and other big pre-trained models, solutions to problems in natural language processing follow a very similar pattern. It starts with an extensive model pre-training on broad domain data, followed by fine-tuning to specific tasks. Fine-tuning a model for a specific task is usually done by updating all weights of model. This has become quite a challenging task considering the size of the current LLM models, for example, GPT-3 with 175 billion parameters. In addition, managing multiple models, each fine-tuned for different tasks, creates additional challenges in terms of training, storage, and operation.

When the model is fine-tuned, all weights are updated to $\Phi_0 + \Delta\Phi$, where Φ_0 are weights from the pre-trained model. This weight update is performed multiple times in a loop during fine-tuning while trying to maximize the language modeling objective (Equation 2.7) [22].

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log(P_\Phi(y_t | x, y_{t<1})) \tag{2.7}$$

The problem arises from the fact that the newly learned parameters $\Delta\Phi$ are the same size as the weights of the pre-trained model, that is $|\Delta\Phi| = |\Phi_0|$ and we would obtain these new parameters for every new task where the model is fine-tuned to.

Hu et al. [22] proposed a technique called **Low-Rank Adaptation** to mitigate these problems and overcome some problems of current solutions, such as inference latency or reduction of the model's usable sequence length. This approach involves a substitution of the update matrix $\Delta\mathbf{W}$ with the lower rank matrix decomposition $\mathbf{BA}$ in a fine-tuning process $\mathbf{W_0} + \Delta\mathbf{W} = \mathbf{W_0} + \mathbf{BA}$, where $\mathbf{W_0} \in \mathbb{R}^{d\times k}$ is a weight matrix of a pre-trained model, and $\mathbf{A}$, $\mathbf{B}$ are low-rank matrices of size $\mathbf{A} \in \mathbb{R}^{r\times k}$, $\mathbf{B} \in \mathbb{R}^{d\times r}$ with rank $r \ll \min(d, k)$. The pre-trained weights W_0 are frozen during training and only $\mathbf{BA}$ is updated. The forward pass with low-rank matrix decomposition modification, where $h = \mathbf{W_0 x}$:

$$h = \mathbf{W_0 x} + \Delta\mathbf{W x} = \mathbf{W_0 x} + \mathbf{BAx}, \tag{2.8}$$

is illustrated in Figure 2.4.

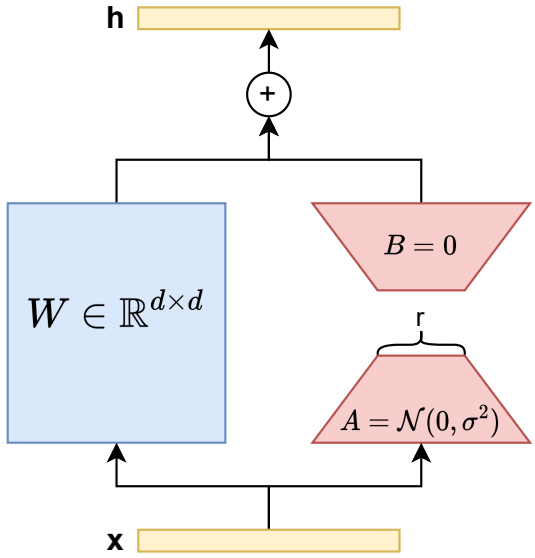

Figure 2.4: Illustration of forward pass with LoRA approach, where $\mathbf{W}$ are weights of pre-trained model, $\mathbf{A}$ and $\mathbf{B}$ are low rank matrices initialized to $\mathcal{N}(0, \sigma^2)$ and 0 with sizes $r \times k$ and $d \times r$ respectively. This illustration is a recreation from the original LoRA paper [22].

2.5.2 Quantization

Although the LoRA approach made the training of the LLMs more affordable, with the size of some current LLMs even the inference is hardware demanding requirement. With billions of parameters, one still needs a large enough GPU. The quantization technique mitigates this problem by reducing the precision of the model. Weights, usually stored in 32 or 16 bits, are scaled to 8 or 4 bit integers, effectively reducing the size of the model, while maintaining comparable performance [33]. Quantization can be defined as [34]:

$$Q(\mathbf{w}) = \Delta \cdot Round(\frac{\mathbf{w}}{\Delta}), \tag{2.9}$$

where Q is qunatization function, $\mathbf{w}$ are weights and Δ is a scaler defined as:

$$\Delta = \frac{max(|\mathbf{w}|)}{2^{N-1}}, \tag{2.10}$$

where N represents number of quantization bits. There are several quantization methods for LLMs such as GPTQ [16] or AWQ [34]. For example, AWQ (Activation-aware Weight Quantization) performs per-channel scaling to reduce the quantization loss of important weights, as in LLMs, not all weights have the same importance.

2.5.3 QLoRA

As mentioned before, fine-tuning large pre-trained models is a widely used approach to improve model performance on some specific downstream task. However, this has become a problem for LLMs because of their size and the large number of trainable parameters. Methods for mitigating this problem during training (LoRA) and inference (quantization) are presented in previous subsections. QLoRA, suggested by Dettmers et al. [13], is an approach that combines quantization and low-rank adaptation techniques to further reduce the memory requirements of the model. This allows models with billions of parameters to be trained on consumer GPUs with memory as small as 24GB [13]. QLoRA reduces memory consumption with 3 main techniques:

- **4-bit NormalFloat** data type - is a new data type for quantization that achieved better results than 4-bit integers or floats
- **double quantization** - quantization of quantization constants
- **paged optimizers** - to avoid gradient checkpointing spikes for mini-batches containing long sequences

Chapter 3

Named Entity Recognition

A vast supply of textual data in various shapes and forms comes with the internet access. Their content covers different styles, from scientific papers to casual conversations on social media. However, this creates a problem, as it is impossible for humans to read and process this amount of data or even navigate it [9]. Information extraction is one of the techniques developed to tackle this problem. In Chang et al. [5], this task is defined as a set of inputs and corresponding extraction targets.

Information extraction (IE) [9, 20, 5] is the process of extracting relevant information from unstructured data. Relevant information includes entities, their relations, etc., while ignoring irrelevant information in a specific domain or domain sets. Hobbs et al. [20] describe two categories of data for IE, unstructured (natural sentences) and semi-structured (the physical layout of the document is important). The datasets in this thesis consist of only unstructured data, that is, natural sentences without any information about the document layout, consisting solely of spoken natural language and its transcription. The extraction targets are predefined structures or objects that can include several slots [43]. The extracted structured data can be used for numerous applications, for example, data analysis, updating or building databases, creating datasets for other tasks, machine learning models, etc. According to Piskorski et al. [43], IE consists of several sub-tasks, including Named Entity recognition. In this work, IE is used in form of NER task to speed up reaction time of emergency services and build new datasets for future models. The goal of named Named Entity Recognition (NER) is to identify and classify pre-defined objects or entities from a given text. Some common entities through the datasets are

- **person** - names and/or surnames of people, characters, etc.
- **location** - names of places, geographic locations, cities, etc.
- **organization** - for example, World Health Organization, football clubs [46]
- **numerical values**.

The number and types of entities can vary depending on the NER system and dataset chosen for training. For example, the original version of the Czech Named Entity Corpus (CNEC2.0)[52] defines 46 entities. Furthermore, some entities can be defined as nested, which means that an entity is composed of several different entities within it, as presented in the CNEC2.0 or NNE corpus [47] (Figure 3.1). In the CoNLL version of the CNEC2.0 (Section 4.2) similar entity types were grouped, creating only 7 entity types. On the other hand, the WikiANN [46] dataset (all languages, including Czech) contains only 3 entity types.

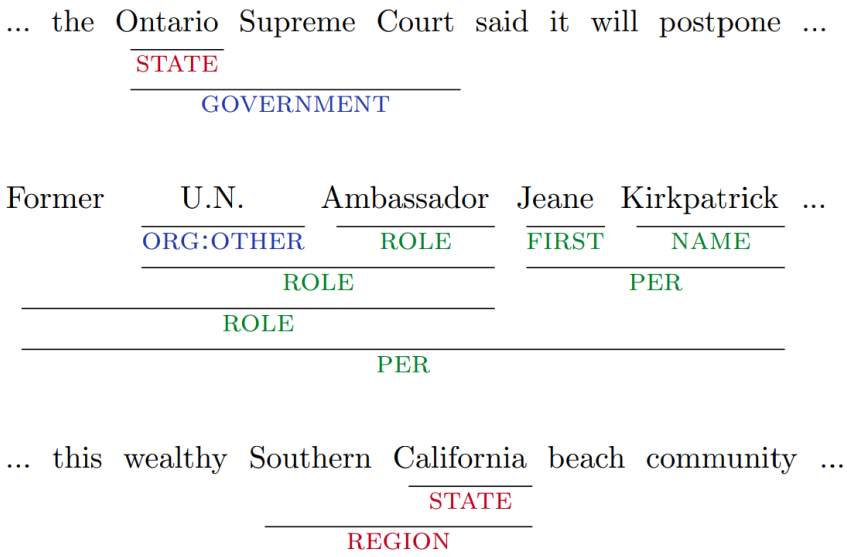

Figure 3.1: Example of nested entities for NER task. Adopted from Ringland et al. [47].

3.1 NER approaches

Early approaches to Named Entity recognition were based on hand-crafted rules and features, and gazetteers. Such rules could be based on grammar rules combined with created contextual rules as presented by Mikheev et al. [37], rules created based on synonym dictionaries or parts of speech tagging [32]. An example of such rules for **person** identification is

$$\text{Xxxx+ is? a? JJ* PROF,}$$

where Xxxx+ represents a sequence of capitalized words, JJ* is a sequence of adjectives, and PROF stands for a profession [37]. Gazetteers are lists of known organizations, locations, people, or other entities, and have been widely used to tackle the problem of Entity Recognition. Ringland et al. [32] briefly summarized other approaches in addition to used methods based on hand-crafted rules. Unsupervised methods based on clustering, where words with the same or similar semantic meaning would belong to the same group. Feature-Based Supervised Approaches that used machine learning algorithms such as Hidden Markov Models (HMM), Decision Trees, Maximum Entropy Models, or Conditional Random Fields (CRF) are used to this day. Features for these models were based on numerical representations of words, word-level features (e.g. parts of speech), or gazetteers. Currently the most used approach is based on deep neural networks.

Conditional Random Field

Conditional Random Field (CRF) [25] is a discriminative probabilistic graphical model that is used to predict labels from data sequences. To this day, CRF models are used for Named Entity Recognition with varying combinations of input features or in combination with pre-trained neural network models. The model is trained to compute the probability $p(\mathbf{y}|\mathbf{x})$ with Equation 3.1, where $\mathbf{x} = x_1, x_2, \dots x_n$ represents the input sequence and $\mathbf{y} = y_1, y_2, \dots y_n$ is a sequence of the corresponding labels, F_k is the feature function that maps $\mathbf{x}$ and $\mathbf{y}$ to

a feature vector and K is the number of features.

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\Big(\sum_{k=1}^{K} w_k F_k(\mathbf{x}, \mathbf{y})\Big) \tag{3.1}$$

Final labels are obtained by selecting the most probable sequence of labels from all possible sequences as:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x}). \tag{3.2}$$

The CRF implementation used in this work is defined with Equation 3.3 and is described in more detail in [50]. It uses only two feature functions denoted as $\theta_1 \cdot f(y_t, \mathbf{x}, t))$ and $\theta_2 \cdot f(y_t, y_{t+1})$. The first feature function represents a neural network model, providing a score of how likely the t-th word belongs to each NER tag. This is also called **emission score**. The second feature function, called **transition score**, is a $C \times C$ matrix, where C is the number of classification labels. This matrix represents how likely it is that the label y_{t+1} belongs to the class c, given the previous label y_t. The term $Z(\mathbf{x})$ is a normalizer to ensure that the equation is a probability distribution and is defined in Equation 3.4.

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{n} \exp(\boldsymbol{\theta_1} \cdot f(y_t, \mathbf{x}, t)) \prod_{t=1}^{n-1} \exp(\boldsymbol{\theta_2} \cdot f(y_t, y_{t+1})) \\ &= \frac{1}{Z(\mathbf{x})} \exp\Big(\sum_{t=1}^{n} \boldsymbol{\theta_1} \cdot f(y_t, \mathbf{x}, t) + \sum_{t=1}^{n-1} \boldsymbol{\theta_2} \cdot f(y_t, y_{t+1})\Big) \end{aligned} \tag{3.3}$$

$$Z(\mathbf{x}) = \sum_{y'} \exp\Big(\sum_{t=1}^{m} \boldsymbol{\theta_1} \cdot f(y'_t, \mathbf{x}, t') + \sum_{t=1}^{m-1} \boldsymbol{\theta_2} \cdot f(y'_t, y'_{t+1})\Big) \tag{3.4}$$

Approaches Based on Deep Neural Networks

Deep neural network models in natural language, speech, or image processing usually outperform previously used methods. A lot of tasks in NLP, including NER, benefit from models' ability to learn on huge amounts of unsupervised corpora, potentially learning more complex language representations, having bigger vocabulary, and modeling natural language better. This approach was accelerated with the introduction of transformer architecture and methods of unsupervised pre-training, as described in Chapter 2. Today, the majority of Named Entity Recognition approaches that report state-of-the-art results on standard NER datasets are based on some pre-trained encoder transformer model followed with the classification layer and optionally CRF on top of it (Figure 3.2). Such architectures are able to efficiently extract language features to perform the token classification.

3.1.1 Label formats in NER

NER systems based on neural networks are trained in a supervised fashion, where the input is a sentence in a particular language and the target is a sequence of entity labels, each for a corresponding word in the input sentence. Labeling comes in different formats that mainly differ when it comes to multiple word entities, e.g., World Health Organization, or distinguishing subsequent entities. Konkol et al. [28] provide a summary of these formats with experiments and references that show that the format in which the entities are labeled can have an impact on the performance of the model. Different NER tags are characterized

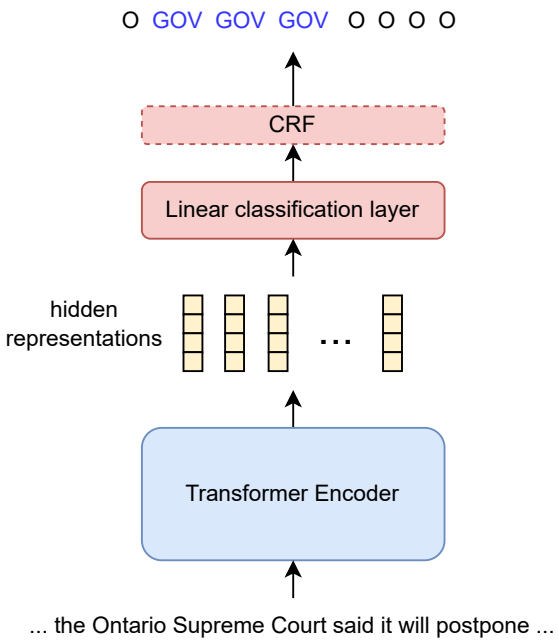

Figure 3.2: Example of transformer encoder-only based neural architecture for Named Entity Recognition with optional CRF layer. The input is tokenized text and the output of the model consists of entity labels corresponding to each word, where **GOV** is "government" entity label and **O** is a widely used label for no entity.

by the entity type and the prefix. Common prefixes are **B** - the first word of an entity, **I** - a word is part of an entity, **L** (or **E**) - last word of an entity, **U** - single word entity, and **O** - words that are not entities (usually comes without entity type sufix). Some widely used formats composed of different combinations of tags are described below [28] and Figure 3.1 shows these formats on a real example taken from the CoNLL-based CNEC2.0 corpus [27].

- **IO** is the simplest representation where every entity is labeled with one tag. The problem in this case is that this format cannot differentiate between entities of the same type that follow each other.
- **BIO** format can represent each entity with two tags. **B**eginning of the entity and **I**nside of the entity. There are two versions of this format, BIO-1 where **B** is used only at the beginning of the entity if it follows an entity of the same type. BIO-2 format uses the **B** tag for every entity. This format will be used in this thesis.
- **BILOU** is the most complex representation adding the **L**ast tag for the last word of an entity and the **U**nit tag representing single word entity.

3.1.2 Evaluation metric

Evaluation plays a crucial role in comparing different NER systems and assessing their performance. NER systems, as many other systems in the field of NLP, are evaluated using the F1 score [12]. F1 score is calculated as a harmonic mean of two other summary measures: **precision** and **recall**. As mentioned by Derczynksi et al. [12], precision and recall are well-suited for scenarios where the objective is to identify a specific subset of items from a larger

Table 3.1: Example of different NER label formats. The prefix is changing based on the format, where **B-** stands for beggining of the entity, **I-** is inside of an entity, **L-** stands for last (end of the entity) and **U-** is used for single-word entities in BILOU format.

Text	Vylet	Bledule	v	Pekle	vlakem	z	Liberce	v	8	12
IO	O	I-Ins	I-Ins	I-Ins	O	O	I-Geo	O	I-Time	I-Time
BIO	O	B-Ins	I-Ins	I-Ins	O	O	B-Geo	O	B-Time	I-Time
BILOU	O	B-Ins	I-Ins	L-Ins	O	O	U-Geo	O	B-Time	L-Time

collection of items. This is applicable for NER, where generally the majority of words in the data do not belong to any particular entity type. Because of this imbalance between entities and non-entities, the accuracy is not well fitting evaluation metric, as labeling all words as non-entities would give high accuracy, but such NER system would be useless. Precision is computed as

$$P = \frac{|true\ positives|}{|true\ positives| + |false\ positives|}, \tag{3.5}$$

and represents a ratio of correctly recognized entities and all entities recognized by the model. Recall, on the other hand, represents how many entities were found by the model from all the entities labeled in the test dataset, and it is calculated as

$$R = \frac{|true\ positives|}{|true\ positives| + |false\ negatives|}. \tag{3.6}$$

F1 score is a special case of the F_β score, where $\beta = 1$, and it is calculated as

$$F_\beta = (1 + \beta^2)\frac{PR}{\beta^2 P + R}. \tag{3.7}$$

However, Derczynksi et al. [12] also present some drawbacks of evaluation using F1. When it comes to the comparison of two models, if one model scores a higher F1 it does not necessarily mean that the model is better. This comes from F1's lack of detail when blending together precision and recall, as it is hard to determine what mistakes the models make or what the differences in their behavior are.

3.2 Previous works

There is plenty of work for NER task that chose various approaches to this problem directly or had NER as a subtask. This section introduces selected previous works done on Czech data serving as a baseline, as the focus of this work is on Czech emergency line calls. In addition, the translation approach is described below as an example of an unconventional approach to this task.

3.2.1 NER on Czech Named Entity Corpus

The Czech Named Entity Corpus (CNEC) has a leaderboard of NER results for this dataset. However, most of the works present in the leaderboard are done and evaluated on the original dataset that includes nested entities and also many entity categories (described in more

detail in Section 4.2). As this work is focused on detecting a smaller number of entity categories, CoNLL version was selected along with two works as baselines.

First one is done by Konkol et al. [27]. In this work, the NER task is performed purely with a CRF model that was trained on a combination of multiple features such as: **lemmas**, **affixes**, **bag of words**, **bi-grams**, various **orthographic features**, and **gazeteers**. Features used for training the model have a context of 2 previous and 2 following words, and the distribution modeled by the CRF is:

$$p(\mathbf{y}|\mathbf{x}) \propto \prod_{j=1}^{N} exp \sum_{i=0}^{M} \lambda_i f_i(y_{j-1}, y_j, x_j), \tag{3.8}$$

where α_i is learnable weight, f_i is a feature function, $\mathbf{y}$ and $\mathbf{x}$ are NER labels and input tokens, respectively. M is number of features and N is number of tokens, achieving overall **F1** score of **74.08** %.

The second work on this dataset is done by Sido et al. [54], where BERT [14] and ALBERT [30]-like models are pre-trained on the Czech corpus and then fine-tuned for the six different tasks separately. For NER by adding a linear classification layer on top of the pre-trained embeddings obtained from the Czert model. The newly added layer was initialized randomly and trained with cross-entropy loss. The BERT-like Czert model achieved a of **86.27**% F1 score, while the SlavicBERT [2] achieved **86.57**% F1.

3.2.2 Translation between augmented natural languages

Paolini et al. [41] proposed a sequence-to-sequence approach to tackle different structured language prediction tasks. Examples of such tasks are joint entity and relation extraction, (nested) **named entity recognition**, relation classification, or coreference resolution. Typically, these problems are solved (as described above) by training task-specific models or discriminator parts on top of the pre-trained models [41]. In contrast, Paolini et al. used a pre-trained encoder-decoder T5 model in single-task and multi-task scenarios, where the model produces the so-called augmented natural language. Augmented natural language is a replicated input utterance in the same language with added special tokens that indicate spans of entities and additional information (such as type of entity or relation) depending on the task. Structured information that is extracted based on the task is present also in natural language form, for example, the person is tagged as **person** instead of the usual **PER** tag. Paolini et al. argue that with this approach, the model can better leverage semantic information. An example of the proposed framework is shown in Figure 3.3 and a more detailed description can be found in the original work of Paolini et al. [41].

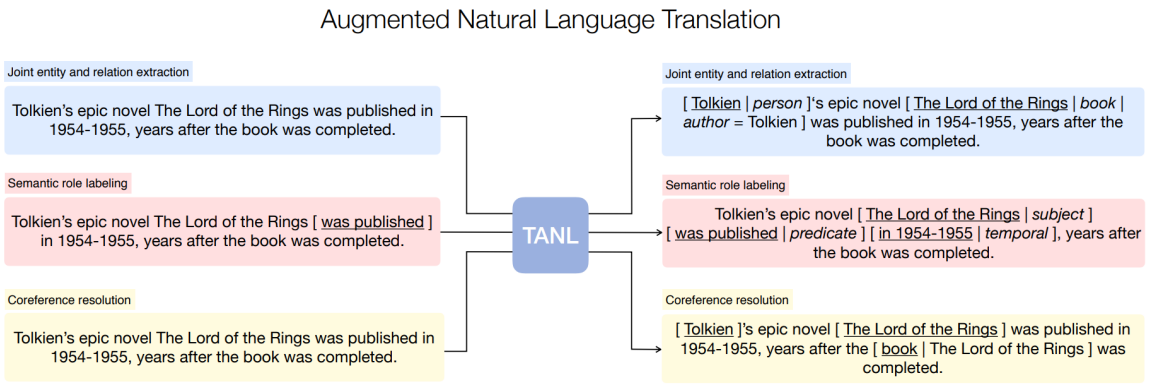

Figure 3.3: Translation between augmented natural languages with T5 model. Adopted from [41].

Chapter 4

Data

Named Entity Recognition (NER) is a well-known task with a significant amount of previous work done, as described in Chapter 3. In all instances where deep neural networks are utilized, the efficiency and quality of models in real-world scenarios are strongly influenced by the quality and amount of training data. Most of the state-of-the-art models for NER are trained or evaluated on either English only datasets or on a combination of English + other languages. The data for Czech NER are sparse. Examples of widely used datasets for NER are CoNLL-2003[1] or OntoNotes Release 5.0[2].

As described in Chapter 1, this thesis aims to provide help to emergency call center workers in the Czech Republic. Within this project, I had access to real-world data from emergency calls. However, since the data is classified and access is restricted, the performance of the proposed models (Chapter 5) had to be tested first with preliminary experiments done on other available datasets, preferably in the Czech language.

4.1 WikiANN

The first dataset used in the preliminary experiments was the Czech subset of the WikiANN dataset (sometimes called PAN-X[3]). This dataset was originally introduced by Pan et al. [40] as part of the cross-lingual framework for entity recognition and linking. However, the data used in the experiments in Chapter 6 were introduced and described by Rahimi et al. [46] for the multilingual transfer task for NER.

The WikiANN data were obtained from Wikipedia articles using its annotated markups. The Czech subset of this dataset is divided into training, validation, and test sets with 20k, 10k, and 10k utterances, respectively. In each utterance, the selected named entities are labeled with either **PER**, **LOC**, **ORG**, or **O**, meaning **person**, **location**, **organization**, or **no entity** respectively. The distribution of the entity types for the training and test set is shown in Figure 4.1.

4.2 Czech Named Entity Corpus

Another dataset used in experiments was the Czech Named Entity Corpus [52] (CNEC), specifically, the second version of this dataset (CNEC2.0) introduced by Konkol et al. [27]

[1] https://huggingface.co/datasets/conll2003
[2] https://huggingface.co/datasets/tner/ontonotes5
[3] https://huggingface.co/datasets/wikiann/viewer/cs

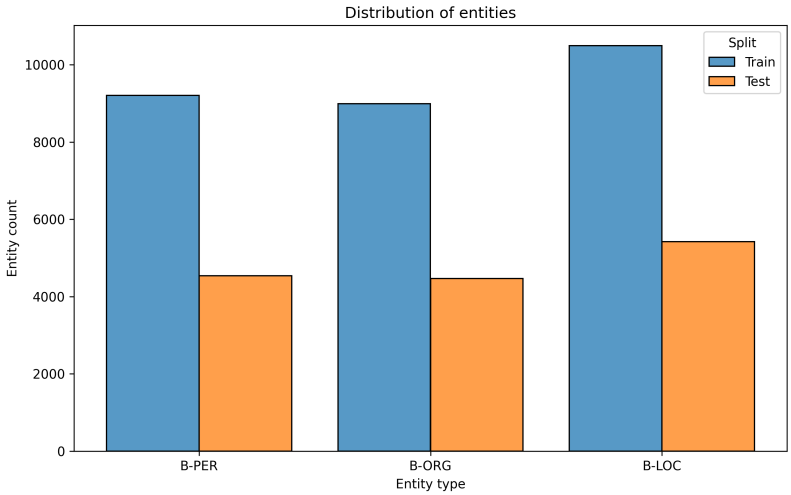

Figure 4.1: Statistics about Czech part of WikiANN dataset containing: B-PER (person), B-ORG (Organization), B-LOC (location) entities.

and modified to the CoNLL format. This version will be further referred to as CoNLL-based CNEC 2.0. The original dataset [52] consists of 46 unique entity types and 4 entity containers, where multiple entity types can be part of one entity container, for example, the forename and surname are part of the "person" container, etc. On the other hand, CoNLL-based CNEC 2.0 [27] includes only seven entities plus the "no entity" label. Another key difference between these two versions is the existence of nested entities (as mentioned in the example above) in the original versions of the CNEC dataset, unlike in the CoNLL-based version. The reason for choosing the CoNLL-based version over the original dataset are that we did not initially expect to predict nested entities. Moreover, the Czech emergency dataset described in the following section will also be prepared in the non-nested entity fashion.

As usual, this dataset is divided into training, validation, and test splits consisting of 7142, 885, and 890 utterances, respectively. The seven entities present in this version are **Time**, **Geography**, **Person**, **Address**, **Media**, **Other**, **Institution**, and "**no entity**", and the corresponding distribution of entities in the training and test split is shown in Figure 4.2.

4.3 TCTV112 - Czech emergency line calls

This section describes emergency call center data and the preparation of the dataset. It also serves as a documentation of the dataset for future work on the TCTV112 project. The data consists of a collection of mp3 recordings with corresponding meta-data from real-world emergency calls in the Czech Republic. This collection of raw data with metadata files and datasets will be referred to as the TCTV112 dataset.

TCTV112 data come from emergency centers in three locations: Praha (Prague), Olomouc, and Plzeň (Pilsen). One part of the data is from 2020 and the second part is from 2022. Various data are stored in multiple files across the directories (depicted in Figure 4.3). The `index.xml` contains mainly metadata that are not useful for this task, such as time

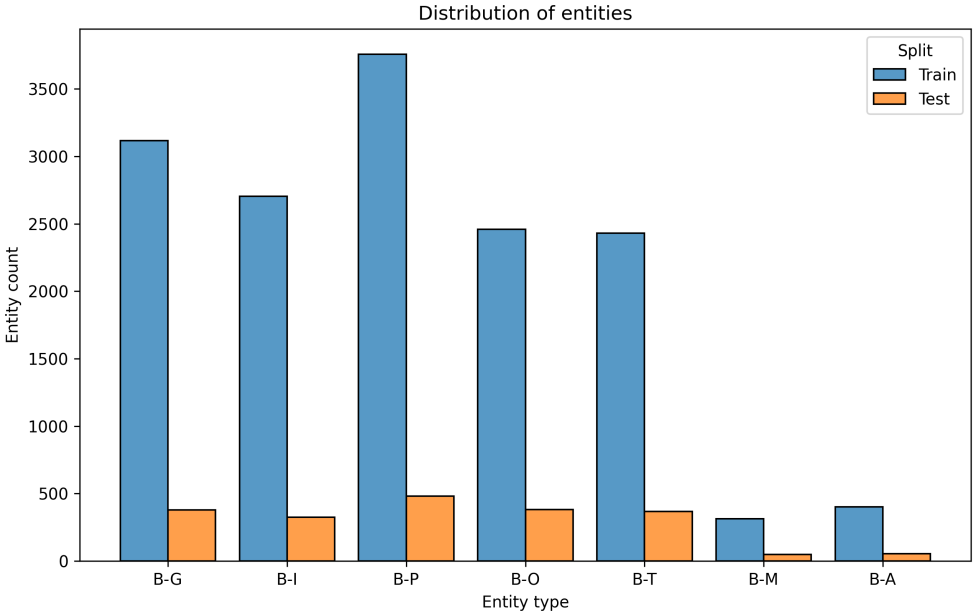

Figure 4.2: Number of entities in coNLL based CNEC 2.0 dataset.

and different IDs that are only used to pair individual mp3 files with other data needed to create the NER dataset.

The `<year_month>.csv` files in the `Udalosti` (events) directory contain information about the location and some description of the emergency situation, the number of injured people, cars, etc. Finally, the `NIS_DV_2020` directory contains the data sentences in XML format. It also contains additional information, such as additional description of the emergency or, more importantly, the name of the caller. An example of a data sentence is shown in the Appendix A.

However, at the time of writing this thesis, the CSV files are missing crucial information such as the caller's name. Furthermore, even though both data sources have a structured format, it is a mixture of information provided by the caller and information obtained by the agent from emergency line systems and stored manually. For example, the caller can provide the location vaguely or not at all, and the agent can get the location using GPS or other available methods. The "what happened" information is in the form of an informal description in natural language ranging from a couple of words to a longer sentence. Chapter 6 presents approaches to mitigate these drawbacks (multiplied with propagated ASR errors) to improve the performance of the proposed models that can be further used for automatic labeling of the dataset for future work.

4.3.1 Data linking

The mp3 recordings were paired with meta-data to create the TCTV112 dataset with the number of utterances shown in Table 4.1. The information in the table shows the number of calls (one call can be stored as multiple mp3 recordings) with the specified raw metadata available from linked XML and CSV files. These numbers will change throughout the data

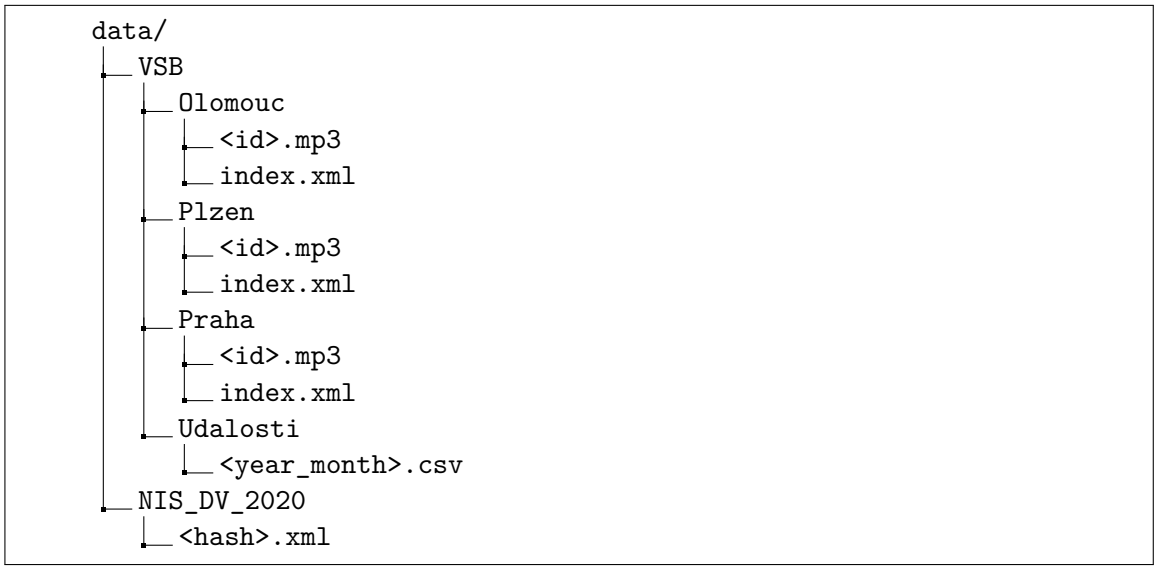

Figure 4.3: Content of emergency calls data directory. CSV files contain various information about the calls such as description of what happened, location, number of people injured, number of cars in traffic accidents, etc. XML files in `NIS_DV_2020` contain similar information to CSV with the addition of name of the caller. The XML files are available for the year 2020 only.

Table 4.1: Number of emergency calls available for each year. All numbers in the table correspond to the number of calls, not the number of mp3 recordings since some calls are stored as multiple mp3 recordings. **Name** is for calls with the caller name available in the metadata (suitable for NER). **Desc** contains a text description of the emergency. **Loc** utterances contain the location of the emergency. **Info** utterances contain some additional notes, which could be useful for the creation of NER dataset.

Year	**Number of calls**				
	Total	**Name**	**Desc**	**Loc**	**Info**
2020	1 945 849	85 410	454 531	1 415 484	18 549
2022	1 409 898	-	384 482	1 409 898	-

preparation process as more structured information will be extracted and some calls will be filtered out.

4.3.2 Producing call transcriptions

Another part of the emergency calls data preparation pipeline is to obtain transcriptions of audio data. This is a crucial part of the process, as entity recognition is performed on textual data, with all experiments done with a cascade of ASR + NER. Prior to this work, no true labels were available for Named Entity Recognition besides the metadata created by the call center agents during an emergency and barely any transcriptions with the exception of some automatic transcription of calls recorded during a tornado disaster that happened in the Czech Republic in 2021.

The transcribed audio recordings were selected based on the audio format and paired metadata available for a particular call. For year 2020, most calls were stored in mono

audio, where both speakers, the caller and the agent, are mixed in one channel. This is not ideal, so all mono recordings were removed, shrinking the data used for the experiments. This 2020 subset included all types of emergency events, resulting in a dataset containing 121k recordings. Figure 4.4 shows the distribution of different emergency event types in December 2020. All of these were automatically transcribed, but the final number of utterances obtained from transcriptions was further reduced for the NER experiments based on the available metadata.

For 2022, everything was recorded and saved as stereo; therefore, there is no data format restriction, and the focus can be shifted to choosing data that match the intention of this work. That is, to help emergency line agents during larger disasters that cause the number of incoming calls to skyrocket. Figure 4.5, shows the distribution of calls to the emergency line in 2022 with highlighted timestamps that were chosen for the 2022 TCTV112 dataset. The large spikes during these timestamps were caused by strong winds and fires. The incoming calls for those events contain mainly property damage and fallen trees. After selecting the calls for transcriptions, they were further filtered for the NER experiments based on the following assumptions:

- There is no reason to use the calls with missing metadata information as there is no way to label the data without any existing solution. A pre-trained NER model or rule-based approach could be used, at the cost of potentially miss-labeled entities. However, such transcriptions will be removed prior to the creation of the dataset.
- Calls without the metadata can also indicate short calls without conversation, which provide no value for the experiments, and also no data for any model training.
- Only stereo recordings will be used because in future experiments, there will be need to distinguish between the caller and the emergency call center agent. Data in this format can be used to simulate fully automated conversations with an audio channel that contains caller audio only.

Automatic Speech Recognition model

A neural network speech-to-text model (ASR) provided by Phonexia[4] was used for audio transcriptions. The ASR architecture is a Factorized Time Delay NN (TDNNf) composed of 6 convolutional and 9 TDNNf layers and was trained with the neural network x-vector extractor. The model and training process are described in more detail in the original work [26]. The ASR model was trained on 110 hours of BABEL data, and the x-vector extractor was trained on multilingual data containing 204k recordings from 40k speakers and 27 languages [26].

Due to the poor quality of emergency call recordings, two additional adaptations were made in the preceding emergency line project[5] (a project on the integration of neural networks / machine learning in an emergency line call center). The first modification was made to the acoustic model, which was adapted to the TCTV112 data using 507 hours of calls. The second was a 3-gram language model (LM) added to the ASR. This LM was trained on the text extracted from the data sentences (Appendix A) containing almost

[4] https://www.phonexia.com/

[5] VK01020132 - Praktické ověření možnosti integrace umělé inteligence pro příjem tísňových volání pomocí hlasového chatbota, vyvinutého v rámci výzkumného projektu BV č. VI20192022169, s technologií pro příjem tísňové komunikace 112 a 150 v ČR (TCTV 112). Available at: `https://starfos.tacr.cz/cs/projekty/VK01020132`

4 million words. The ASR was evaluated on a small test set containing 102 recordings that was also created in the preceding TCTV112 project. These ASR modifications improved the performance from 59% to 49% WER. High word error rates, despite the adaptations of the ASR made specifically for this data, are caused by the poor quality of the recordings. The recordings are stored in compressed mp3 format with a low bit rate of $24kbit/s$. After the change of recording storage from mono to stereo, the bit rate stayed the same, but 2 channels were stored instead of one, effectively worsening the audio quality.

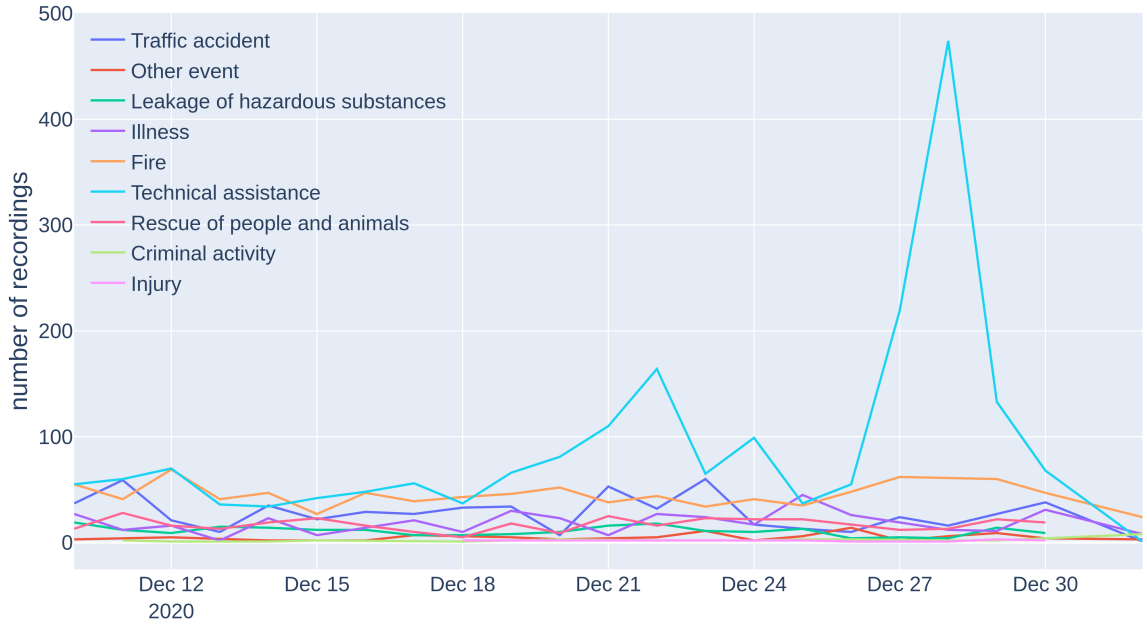

Figure 4.4: Distribution of emergency line calls based on the emergency event type in December 2020 and beginning of January 2021.

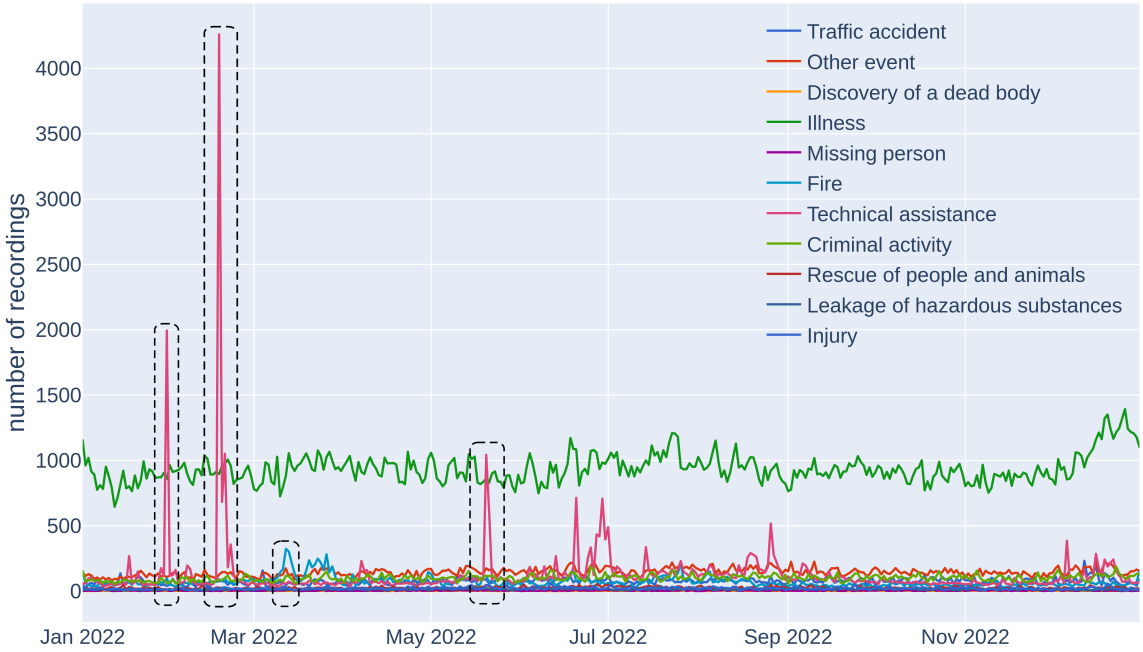

Figure 4.5: Distribution of emergency line calls based on the emergency event type in 2022. Calls marked by the black boxes were selected for the 2022 TCTV112 dataset. These big spikes are caused mainly by the strong winds where people call about the property damage or fallen trees. In addition, one spike in calls caused by the fire was selected for the dataset diversity.

Chapter 5

Proposed solutions

This chapter describes different approaches and model architectures used in experiments (Chapter 6) along with their input/output data format that was pre-/post-processed as needed for the task. Three approaches were explored for the NER task; Pre-trained encoder-only models, Seq2Seq models, and Large Language Models.

The "traditional" token classification approach was chosen first, as this is the most common way of doing NER, followed by seq2seq and LLMs. This decision was based on multiple factors present in this field. Many works have shown state-of-the-art results for NER by extracting word embeddings or other information from text and performing classification on top of these embeddings, e.g., an encoder+classification layer, Conditional Random Field (CRF), or a combination of both. At the time of writing this thesis, the use of LLMs such as LLaMa 2 [56] or GPT models [59, 44, 4] is a rapidly growing area of research, with new models coming out every couple of months. These models showed great capabilities in a variety of NLP tasks, even for tasks they were not trained for. Mistral 7B [23] LLM will be explored further in this work for NER on emergency call data.

5.1 Implementation Tools

The models and experiments were implemented in Python, mainly with the HuggingFace[1] libraries that provide an assortment of NLP models, classes to train these models, and datasets for various NLP tasks. In addition, the implementation of CRF was taken from the AllenNLP library [17].

Regarding computational hardware, all experiments were conducted on the FIT SGE[2] cluster mainly on 48GB Quadro RTX 8000 GPU. The following experiments on real data from the emergency call center were performed on a separate isolated computer with a 24GB NVIDIA RTX A5000 GPU, provided by the BUT Speech@FIT group.

5.2 Encoder-only transformer models

Token classification with embeddings from pre-trained encoder-only models was chosen as a first approach in this work. It is a fairly common approach for the NER task with state-of-the-art results on various datasets. Multiple transformer-based encoder models were explored with the addition of an optional CRF layer on top of a classifier layer, as shown

[1] https://huggingface.co/

[2] https://www.fit.vut.cz/units/cvt/cluster/.en

in Figure 3.2. Only multilingual encoders were explored as oposed to Czech-only models, because such models are usually trained on much larger corpora. In addition, multilingual models could potentially work better in case of code-switching or different language in real-world scenarios, for example, in the case of an accident where a foreigner calls the emergency line.

XLM-R

One of the models used in this thesis was multilingual XLM-R [8] with 588M parameters. This model is based on the original Transformers [57] trained in a multilingual manner using the Masked Language Modeling approach shown in Figure 2.3, with the difference in not using language embeddings (gray embedding boxes in Figure 2.3), which should improve code-switching abilities [8]. The model used in the experiments was trained on 2.5TB of filtered CommonCrawl data containing 100 languages and has a vocabulary of 250k tokens. Conneau et al. [8] show that such multilingual models can outperform their monolingual BERT counterparts.

LaBSE

Another model tested for the NER was LaBSE (Language-agnostic Bert Sentence Embedding [15]) with 470M parameters. This model was trained as a dual BERT-based encoder model (Figure 5.1) for **sentence-level** embeddings, however it can also produce (at least in HuggingFace implementation) **word-level** embeddings.

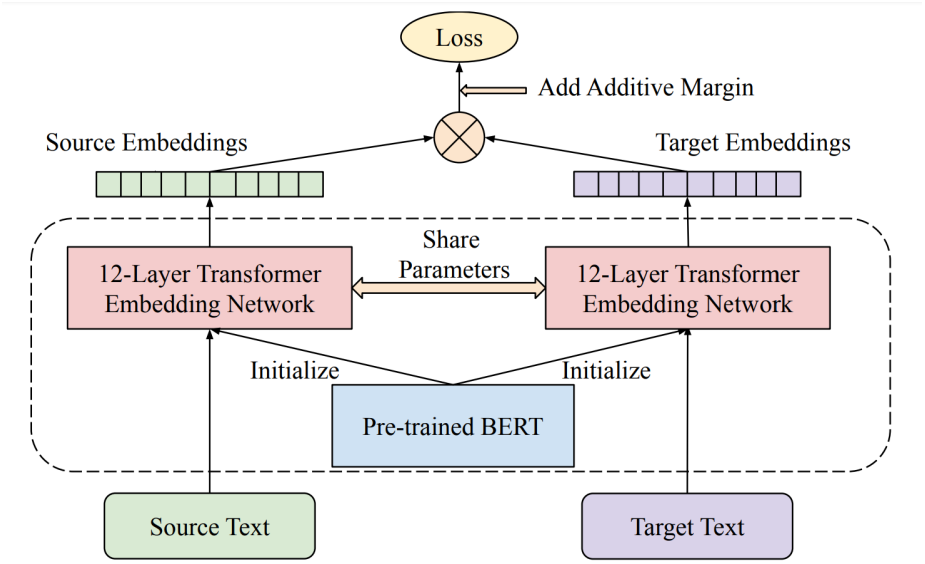

Figure 5.1: LaBSE dual encoder architecture, taken from Feng et al. [15].

Multilingual DeBERTaV3

This model is a multilingual version of the DeBERTaV3 [19] model trained on the same data as the XLM-R model and has 278M parameters. It is a Transformer-based model that improves the results on downstream Natural Language Understanding (NLU) tasks with two major changes in the training method.

First, instead of MLM used for BERT [14] or XLM-R [8], DeBERTaV3 uses the **Replaced Token Detection** method in which the model is trained with two encoders. One encoder (generator) is trained with the MLM objective, whereas the second one (discriminator) is trained as a token-level binary classifier. The generator is trained to produce masked tokens, while the discriminator performs classification of these tokens, determining whether the generated token is the original or not.

Second, the authors propose a method of sharing the embeddings between the generator and the discriminator during the training in order to mitigate the drawbacks of the embedding sharing ("tug-of-war dynamics") [19].

5.3 Encoder-Decoder approach

The second approach was inspired by Paolini's paper [41] with some modifications, as Paolini et al. did the experiments only on the English data with the T5 model. The target dataset of this thesis is in Czech and therefore multilingual T5 (mT5) [58] was chosen. The small version of mT5 has 300M parameters and the base version has 582M parameters. The mT5 training closely follows the training of the T5 model, and, similar to XLM-R, mT5 was trained on 101 languages. During training, languages were sampled with probability $p(L) \propto |L|^{\alpha}$, where $|L|$ is the number of examples for the given language and $\alpha = 0.3$ based on the ablation experiments in the original article [58].

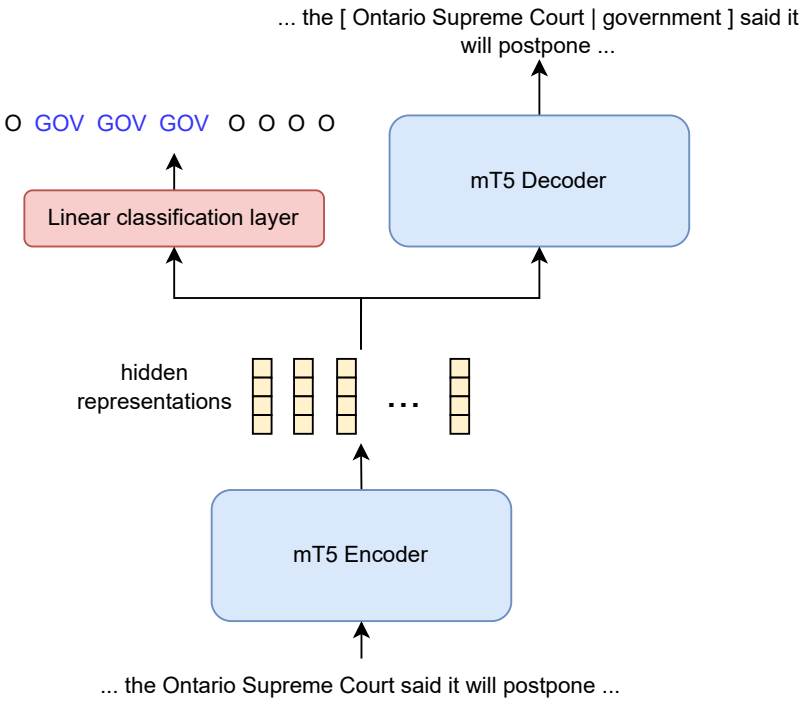

Figure 5.2: mT5 architecture with additional token classification head used in this work.

In addition to the original encoder-decoder architecture approach, a classification layer was added on top of the encoder part of mT5 as shown in Figure 5.2. This allowed the model to be trained in a multitask fashion, where the encoder also performs a token classification task during the training in order to improve the results. The final loss for the multi-task

approach was:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{seq2seq} + (1 - \alpha) \cdot \mathcal{L}_{token}, \tag{5.1}$$

where the α is a hyper-parameter, the $\mathcal{L}_{seq2seq}$ is cross-entropy loss on augmented natural language sentences and $\mathcal{L}_{token}$ is cross-entropy loss computed on NER labels similarly as during the basic token classification task.

5.4 Large Language Models

Large Language models (LLMs) are achieving state-of-the-art results in a variety of NLP tasks. Because of their capabilities, they are becoming very popular and new models are released every couple of months. For this work, the Mistral 7B [23] model was chosen for its performance on various NLP tasks, multilingual and fine-tuning abilities. Jiang et al. [23] implemented multiple changes compared to the LLama 2 [56] model. **Sliding Window Attention** (SWA), shown in Figure 5.5, is an attention mechanism in which the hidden state h_i at some position i attends only to W hidden states in the previous layer at the positions between $i - W$ and i. When this mechanism is used across all layers, the hidden state h_i in layer k can attend to previous $W \times k$ hidden states. The Sliding-Window Attention is reportedly two times faster than the vanilla attention. **Rolling Buffer Cache**, shown in Figure 5.5, is an implementation of cache for attention keys and values. Because the SWA has a fixed attention span of size W, the cache also has a fixed size W. The keys and values at timestamp i are stored in position $i \mod W$ effectively reducing the memory usage, without affecting model quality. **Pre-fill and Chunking** method, shown in Figure 5.5, utilizes LLM prompt that is known from the beginning. This means that the keys and values cache can be pre-filled before the tokens are generated one-by-one. However, nowadays the prompts can get very long. The chunking takes care of long prompts by dividing them and pre-filling the cache with each chunk.

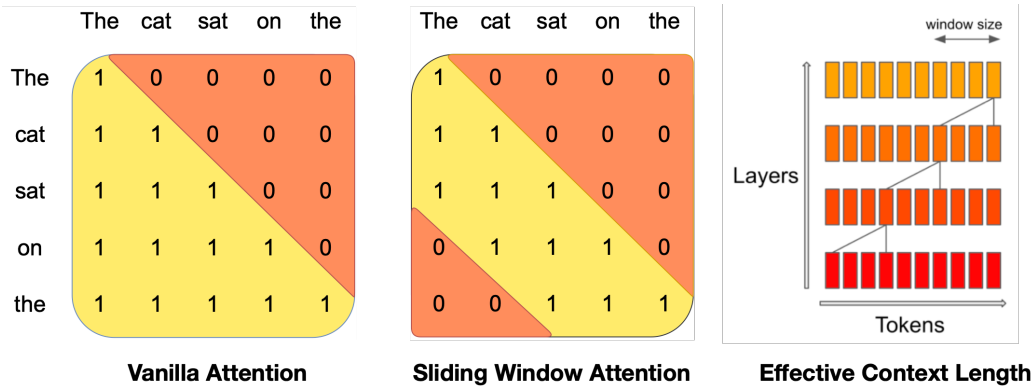

Figure 5.3: Example of **Sliding Window Attention** adopted from [23] with window size $W = 3$. The predicted token can be influenced even by tokens outside of the sliding window through the stacked attention layers (the right-most image).

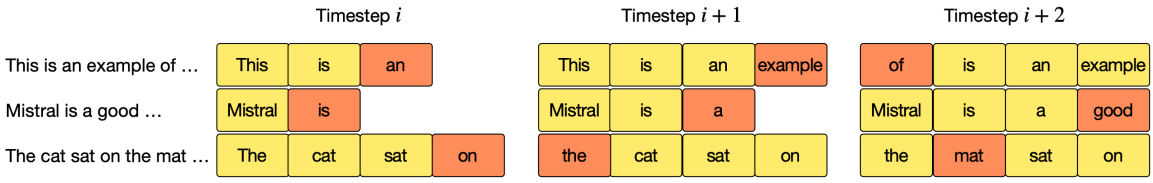

Figure 5.4: Example of the **rolling buffer cache** adopted from [23] with window size $W = 4$. At the timestamp $i + 2$ the input in first and third row are larger than the window size. Therefore the cache is over-written with the new keys and values at the position $i + 2$ mod W.

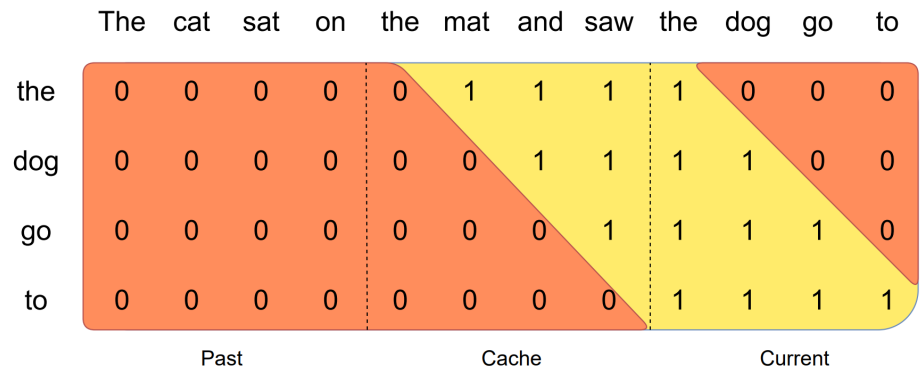

Figure 5.5: Example of **pre-fill and chunking** adopted from [23]. The prompt "The cat sat on the mat and saw the dog go to" is split into three chunks. The example shows the masking for the third chunk (current) where it attends to itself and to the previous chunk (cache) via sliding window.

Chapter 6

Experiments

This chapter describes the experimental setup for the datasets presented in Chapter 4 and for the proposed solutions presented in Chapter 5. It is divided into three parts; exploring different approaches on public Czech NER datasets (cs subset of the WikiANN [40], CoNLL-based CNEC2.0 [27]), creating Czech emergency calls (TCTV112) datasets, and NER on created TCTV112 datasets.

6.1 Named Entity Recognition on Czech datasets

Because the Czech emergency calls are proprietary data and were not available at the beginning of this work, different methods and models were implemented and tested on the Czech part of WikiANN [40], CoNLL-based CNEC2.0 [27] datasets. This section is divided into two parts. The first part consists of experiments with traditional token classification objective, while the second set of experiments insert entity tags directly into the text, as introduced in Section 3.2 (Previous works), and need further post-processing.

6.1.1 Token Classification

This approach leverages models pre-trained on large quantities of data, mainly in self-supervised way and achieving SOTA results for entity classification on various datasets such as CoNLL-2003[1] or OntoNotes Release 5.0[2], WikiANN [46], etc. The architectures in this subsection of experiments follow the process presented in Section 5.2 with a pre-trained tokenizer, and a pre-trained encoder model, followed by a linear classification layer and optional CRF layer. Both, the pre-trained encoder and the classification layer, were fine-tuned on the corresponding datasets. The classification layer was initialized randomly. The transition matrix of the CRF layer was initialized in the following ways:

- **init** - transition matrix was initialized to transitions probabilities computed on the training set of the dataset and fine-tuned with the model
- **train** - every value in transition matrix was initialized to -1000 (as in original implementation) [17] and trained with the model
- **freeze** - same as **init** but the transition matrix was kept frozen during the training.

[1] https://huggingface.co/datasets/conll2003
[2] https://huggingface.co/datasets/tner/ontonotes5

In case of the CoNLL-based CNEC 2.0 [27] and WikiANN [46] datasets, the added CRF slightly improved the final results. However, there is almost no difference between the initialization approaches listed for the CRF transition matrix. An example of transition matrix initialize from the training data is shown in Figure 6.1.

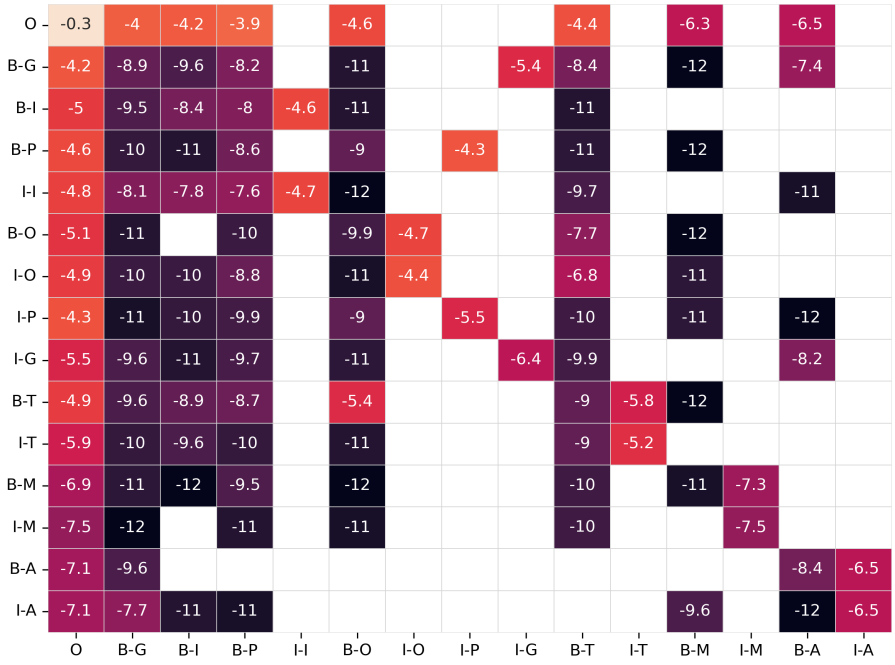

Figure 6.1: Example of the CRF transition matrix. Transition values were initialized from the training data as log-probabilities of transitions from the label on the y axis to the label on the x axis. The labels are in BIO2 format (described in Section 3.1.1). The empty transitions are set to -1000 as in the original implementation of CRF [25].

For token classification experiments, XLM-R [8], mDeBERTaV3 [19] and LaBSE [15] models were chosen with the size of 588M, 278M and 470M parameters, respectively. All models were trained for 20 epochs. The early stopping was used during the training to prevent model from over-fitting because the Czech NER datasets are fairly small. In this stage, the model checkpoint with the highest overall F1 score (computed over the 7 entity types) was chosen. Event though, the focus in the following experiments will be only on **location** and **person** entity types. Token classification models were trained on a single GPU as it took only approximately 1 hour to train on these datasets. The batch size was set to 64 with learning rate $1e^{-5}$ and no gradient accumulation.

The results are shown in Table 6.1 with the XLM-R model achieving the best scores on both datasets. However, the mDeBERTa-v3 model achieved comparable results, and is half the size.The WikiANN dataset was created from Wikipedia data, and the pre-trained encoder model saw Wikipedia data during the pre-training with a high chance of seeing also the WikiANN test data. Because of this, the experiments on the WikiANN dataset serve mainly as a "sanity check" to ensure the model is working. For the CoNLL-based CNEC 2.0 dataset, XLM-R achieved a better F1 score than the baseline results reported in [54].

Table 6.1: Performance of different pre-trained encoder models on Czech entity recognition datasets. The CRF column indicates if the CRF was used and type of its initialization. * - results taken from original paper [54].

Model	CRF	coNLL-based CNEC2.0				wikiann cs			
		Acc	**F1**	**P**	**R**	**Acc**	**F1**	**P**	**R**
CZERT*	-	-	0.866	-	-	-	-	-	-
XLM-Roberta	-	0.973	0.877	0.861	0.893	0.976	0.929	0.924	0.934
	train	0.973	0.885	0.875	0.894	-	-	-	-
	init	0.973	**0.888**	0.878	0.897	0.977	**0.939**	0.939	0.940
	freeze	0.973	0.885	0.876	0.893	-	-	-	-
LaBSE	-	0.969	0.863	0.846	0.881	0.975	0.924	0.919	0.928
mDeBERTaV3	-	0.968	0.855	0.833	0.878	0.973	0.919	0.912	0.926
	train	0.97	0.875	0.864	0.886	0.972	0.925	0.921	0.929
	init	0.969	0.876	0.865	0.886	0.972	0.925	0.921	0.929

6.1.2 Seq2Seq NER

The sequence-to-sequence approach proposed by Paolini et al. [41] and introduced in Section 3.2.2, was explored for the Czech NER, with some modifications for the experiments presented in this section. The mT5 model [58] was used, as its multilingual pre-training enables its use on data in Czech language. The architecture with additional encoder token classification loss is described in Section 5.3.

The CoNLL-based CNEC 2.0 dataset (Section 4.2) was originally prepared for the token classification NER approach, where each word has an entity label assigned. This had to be updated for the seq2seq task by creating the `augmented natural language` from text utterances and the corresponding entity labels.

Two ways of augmenting natural language were explored. The examples are shown below. In the first approach, new special tokens are added to the text, tokenizer, and model itself. Every entity is encapsulated between start and end tokens with the corresponding entity type. In the CoNLL-based CNEC2.0 [27] dataset, 7 entity types are used that correspond to 14 new special tokens, as shown in the example below (**person** is labeled as words between the **<P>** and **</P>** tags).

The second approach follows the idea of the original paper [41] more closely. In this case, every entity is encapsulated in the same brackets "[ ]". The entity type is present as a word in natural language after the "|" delimiter. An example of this augmentation is shown for the same sentence from the CoNLL-based CNEC2.0 dataset [27].

Augmentation with special tokens

Výrazným oslabením jsou odchody brankáře **<P>** Čapka **</P>** do **<G>** Bratislavy **</G>**, útočníka **<P>** Poukara **</P>** do **<I>** Slavie **</I>** a **<P>** Romana a Petra Kaňkovských **</P>** do **<G>** Znojma **</G>**.

Augmentation with natural language

Výrazným oslabením jsou odchody brankáře [Čapka | **person**] do [Bratislavy | **location**] útočníka [Poukara | **person**] do [Slavie | **institution**] a [Romana a Petra Kaňkovských | **person**] do [Znojma| **location**].

Experimental results

This approach was trained and evaluated on CoNLL-based CNEC2.0 dataset because the WikiANN dataset seems too artificial compared to real emergency line data. These experiments were carried out similarly to token classification. Only one 49GB GPU was used, as the mT5 [58] model versions are similar in size to XLM-R [8] and mDeBERTaV3 [19]. Small and base versions of mT5 were used in these experiments with 300M and 582M parameters, respectively. The batch size had to be reduced to 16 because the seq2seq model generates natural language sequences instead of producing only the labels for each word. To compensate for a smaller batch size, the gradient accumulation was set to 8 steps, effectively simulating a larger batch size. Based on the initial experiments, the learning rate was set to $1e^{-3}$ with a cosine learning rate scheduler. Furthermore, various loss weights α were explored for multitask learning with token classification as an auxiliary task. The final loss was computed with Equation 5.1.

The results of NER with the mT5 [58] encoder-decoder model are shown in Table 6.2. In general, the seq2seq approach performed worse than the token classification. The introduction of the second task, token classification, improved performance of the model. Experiments with different values of weight α were carried out to balance both losses to maximize performance on the NER task. Although this approach still yielded worse results than the encoder-based token classification technique, the final result is highly dependent on the post-processing and evaluation method. In the case of CoNLL-based CNEC2.0, the dataset is quite "dirty" and even after removing redundant characters such as: `„.,[]` etc., some model outputs that are marked as wrong are questionable.

As an example, there are occurrences of **person** entities in the form of **person conjunction person** for which the true labels are **B-P I-P I-P** counting the **conjunction** also as part of the entity. The model, on the other hand, tags only the **person** entities producing the **B-P O B-P** output, which is evaluated as completely wrong as shown in an example from the CoNLL-based CNEC2.0 dataset:

```
LABELS: ...zmátla policii Dvojčata<P> Ray a Jay Nugentovi</P>...
MODEL : ...zmátla policii Dvojčata<P> Ray</P> a<P> Jay Nugentovi</P>...
```

There are some other observations of this behavior. For example, one is in geographical locations, where the model omits "v" (meaning "in") between two geographic names.

Both augmentation methods yielded comparable results, showing that the model performed better on the natural language-augmented version of the dataset. Control experiments were run for the natural language augmentation, where instead of English entity types, **person** for example, Czech translations (**person**) were used, however the English version performed slightly better.

6.1.3 Summary

Experiments in this section explored pre-trained encoder-based and sequence-to-sequence approaches for Named Entity Recognition task on publicly available Czech datasets. The pre-trained encoder-based token classification approach with XLM-R [8] as multilingual encoder

Table 6.2: Performance of mT5 small and base version on sequence-to-sequence NER with optional token classification task introduced during the multi-task training. Argument α is used as a weight for seq2seq loss and $1 - \alpha$ for token classification loss.

Model	$\boldsymbol{\alpha}$	**natural language augment**			**special tokens augment**		
		F1	**P**	**R**	**F1**	**P**	**R**
mT5-small	0.5	0.662	0.692	0.634	0.767	0.758	0.775
	0.7	0.779	0.787	0.770	0.770	0.77	0.767
	0.9	-	-	-	0.735	0.735	0.735
	1.0	-	-	-	0.774	0.770	0.779
mT5-base	0.5	0.816	0.823	0.808	0.754	0.740	0.768
	0.7	**0.830**	0.831	0.828	0.810	0.810	0.810
	1.0	0.812	0.817	0.807	0.788	0.776	0.800

achieved the best results outperforming the state-of-the-art overall F1 score on CoNLL-based CNEC 2.0 [27] dataset at the time of writing this thesis.

Experiments showed that even the sequence-to-sequence approach with encoder-decoder architecture is a viable option with a bit of performance degradation depending on the model size. Taking into account the points raised in the original seq2seq paper [41], this approach can be beneficial when used in multi-task settings instead of pure entity recognition. Another disadvantage of seq2seq, taking into account the goal of this thesis, is the need for a heavy post-processing of the output in comparison to encoder-based token classification. The best results and alignment between the output entity types and inputs to the model are the reason why encoder-based models will be used for experiments done on real emergency line data. In addition, the performance of the LLMs models will be tested on the created emergency line datasets.

6.2 TCTV112 - from transcriptions to labels

This section describes the main pipeline and additional methods for creating multiple labeled NER datasets from the available TCTV112 data that are used in experiments in the following Section 6.3. Some parts presented in this section were implemented in parallel or after some initial experiments described in the following Section 6.3. Therefore, the information from those experiments was used during the creation of some subsets described here.

Following the initial data pre-processing described in Section 4.3, the available data from various sources were linked to create several dataset files that consist of ASR transcriptions of selected calls, corresponding audio recording, and multiple columns containing selected metadata that could be important not only for NER. This leads to multiple datasets, shown in Table 6.3. The sets differ in size and the labeling process.

6.2.1 Creating Named Entity Recognition Labels

As mentioned in Section 4.3, the data for the TCTV112 project came only in the mp3 format with some structured metadata (example shown in Appendix A), which is not optimal for NER or the extraction of other information from calls. This brought a secondary goal, that

is, creation of better structured dataset for future work. This is done with information extracted from audio transcriptions by:

1. rule-based approach using information available in the metadata
2. neural network approach using pre-trained models with optional manual correction

The creation of NER labels from the metadata proved to be a bit more tricky than it seemed at first glance. The metadata contains a lot of useful information. Although it is arguably not that large compared to the amount of audio and its transcription, such as the name of the caller, the location of the emergency, or the type of emergency. The metadata comes in a structured form and it is manually created by the emergency call center agent. This means that a lot of information is gathered by the agent with tools available in the call center (for example, the locations), therefore, it may not be present in the corresponding transcriptions. For the 2022 data, caller names were not available at all. Moreover, the audio transcriptions consist of tens of thousands of utterances, while only a fraction of them carry information that is aimed at in this thesis.

The dataset creation pipeline is presented in Figure 6.2 and starts after the selection of data and transcriptions described in Section 4.3. This pipeline consists of multiple stages and was used for both 2020 and 2022 datasets and their variations. First, there are two automated methods for creating labels that are run in parallel:

1. The transcriptions are run through the token classification model pre-trained on coNLL-based CNEC2.0 [27] dataset providing NER labels in BIO2 format.
2. The location of emergency and callers' names presented in metadata files associated with the recording are matched with words in the transcriptions. This is done via a simple method, where both metadata information and transcriptions are lower-cased, and punctuation and diacritics are removed. After this, if the words stored in the metadata are present in the transcription, they are marked with the corresponding labels. All the other words are marked with "**O**" representing no entity.

Entities were labeled in BIO2 format with either **P** for person (caller's name) or **G** for location of emergency. A more common tag for location is usually the **LOC**, but the **G** tag (stands for geography) was used in the CoNLL-based CNEC2.0 [27] dataset on which the token classification model was trained. For further processing and evaluation, the dataset was reduced even more by selecting mainly the utterances containing callers' name and location information that represent only a fraction of all the transcribed utterances. This was done with the intention to make the system reliable in detecting names and locations.

The "Annotated datasets", shown in Figure 6.2, stand for the data annotated with two previously described methods. These datasets were used for further analysis to assess the quality of the labels from both sources and are present in Table 6.3 as `2020_full` and `2020_names` subsets. It was found that the metadata alone, in their current form, are not sufficient to create reliable labels. Moreover, the pre-trained models had a high ratio of false positive classifications. Examples and a more detailed description are given in Subsection 6.3.1.

In the second stage, the labels generated by the model and the labels matched from the metadata were combined. If some words were labeled by both methods with different labels, the metadata label was taken as correct, and the labels provided by the model were overwritten. The labels in these datasets are then manually corrected and split for

the training/test set. This was followed by the augmentation of the training set based on the analysis performed on the "**Annotated datasets**". The augmentation of the training split was done by adding words or phrases that were the most frequently classified as false positives. This augmentation and frequently miss-classified entities are also described in Subsection 6.3.1.

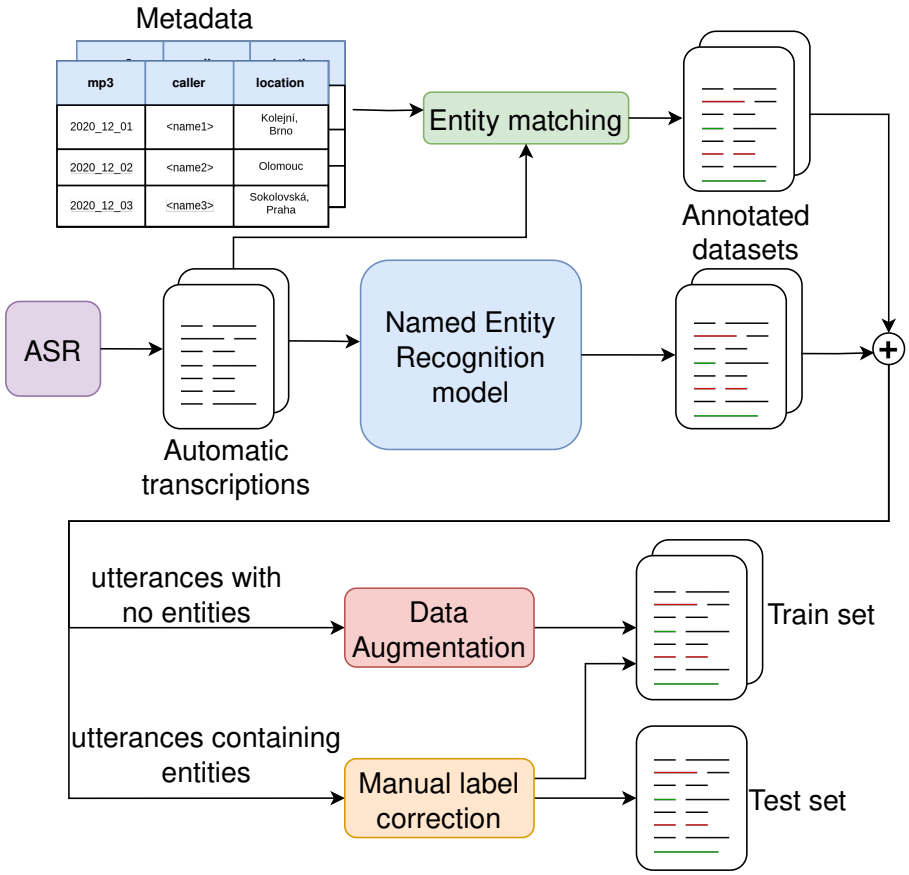

Figure 6.2: TCTV112 pipeline for creating structured dataset with proper NER labels. The ASR transcriptions are labeled for NER task based on the model predictions and content of the metadata. Portion of the annotated data is manually corrected and augmentation is added, creating clean train and test splits.

Tornado subset

The TCTV112 tornado subset was created from two data sources. One consists of a small number of emergency calls made during a tornado in the Czech Republic in 2021. The second is a subset selected based on the emergency event type from 2022, specifically the spikes highlighted in Figure 4.5. These calls are mainly about strong winds and the damage they caused. Therefore, the content is similar to the emergency calls recorded during the tornado.

This subset was anonymized and manually transcribed by colleagues from Speech@FIT group in the firefighters' center in Brno Líšeň. Anonymization was performed on the names and locations of the calls. The names of the callers were completely anonymized by replacing the surname and/or first name with the Czech keywords "(jméno)", "(příjmení)"

meaning “first name” and “surname”, respectively. The locations were anonymized in a similar fashion by replacing the addresses with the Czech keyword “(adresa)”, which means “address”.

The anonymized name tags were then replaced by randomly generated names, while simultaneously labeled with **P** labels in BIO2 format, to match the format of the CoNLL-based CNEC2.0 dataset [27]. The names were randomly generated by combining the top 10 most common names and surnames (both male and female) in the Czech Republic in 2007. The names and surnames were taken from the Czech Statistical Office[3], which resulted in 100 possible combinations of male and female names each. In addition, sometimes only the surname was chosen and the order of surname and first name was randomly shuffled. This was done to prevent the model from learning just these names, but rather to learn also the context in which the names are given by the caller.

Similarly, addresses were randomly generated using ChatGPT [39] with the instruction prompt to generate 40 random Czech addresses consisting of the street name with the street number and the city. Furthermore, the chosen address sometimes misses some components to create more variations and avoid creating the same pattern for locations. All names and addresses were added to the dataset in the nominative case. However, this may cause problems like model over-fitting in the future. As this dataset will grow, more advanced augmentation will be needed.

Final TCTV112 datasets

Several different subsets were created with different labeling methods and even manual correction. The `2020_names` and `2022` subsets, which were manually corrected, and the tornado subset were combined to the final clean TCTV112 dataset containing 1149 person and 1476 location entities. An overview of these datasets is given in Table 6.3.

Table 6.3: Various NER datasets created as a part of this thesis with corresponding number of words and entities. The dataset with the extension “names” is just the subset of the “full” containing all utterances with person entity. The Labels column shows the method of label creation. **Metadata** means that the labels were created based on the information from the metadata. **Manual** means that the entities were labeled manually. **Semi-manual** is present only for the tornado dataset, where the names and locations were tagged manually but the content of the entities was automatically generated.

Dataset	**Labels**	**Words**	**Number of entities**	
			Person	**Location**
`2020_full`	metadata	3845376	3436	23626
`2020_names`	metadata	969878	3436	9121
	manual	5399	465	178
`tornado`	semi-manual	40440	94	775
2022	manual	6740	466	289
`TCTV112_NER`	semi-manual + manual	55750	1149	1476

[3]https://www.czso.cz/csu/czso/nejoblibenejsi_detska_jmena

Table 6.4: Person and location entity classification performance with different variations of XLM-R model on created TCTV112 datasets. For the **tc** (truecase), the model was pre-trained on the CoNLL-based CNEC 2.0, the **tc, lc** variation is a model trained additionally on lowercase version of the dataset. **CRF** means that the model contained the CRF layer on top of the classification head. The Labels column represents the manner in which the labels for the particular subset were created, where **manual** is manual labeling, **semi-manual** stands for automatic creation of names and locations with labels in place of manually anonymized information (described in more detail in Subsection 6.2.1). For the **metadata** subset, the labels were created by automatically matching metadata with text transcriptions.

Dataset	Labels	XLM-R variation	Person			Location		
			F1	P	R	F1	P	R
`tornado`	semi-manual	tc	**0.703**	0.727	0.681	0.160	0.463	0.096
		tc, lc	0.505	0.385	0.734	**0.687**	0.706	0.668
		tc, lc, CRF	0.470	0.343	0.745	0.656	0.713	0.6077
`2020 names subset`	metadata	tc	0.188	0.162	0.225	0.172	0.383	0.111
		tc, lc	0.300	0.219	0.476	0.250	0.204	0.321
	manual	tc	0.449	0.841	0.307	0.143	0.517	0.082
		tc, lc	**0.711**	0.704	0.719	**0.761**	0.855	0.685
		tc, lc, CRF	0.705	0.745	0.670	0.639	0.748	0.558
`2022 subset`	manual	tc	0.533	0.772	0.408	0.138	0.710	0.076
		tc,lc	**0.717**	0.699	0.736	**0.563**	0.740	0.455

6.3 Named Entity Recognition on TCTV112 dataset

This section describes the entity recognition on real emergency call centers data from Czech Republic, on various version of datasets created as part of this thesis. Experiments are carried out with encoder-only token classification models selected based on the results obtained on the publicly available Czech NER datasets. In addition, with the rising popularity and capabilities of large language models (LLM), some experiments were done to assess the performance of selected LLMs on this task.

6.3.1 Inference with pre-trained models

First, the objective was to establish some baseline performance numbers and statistics on real data mainly focused on classification of **person** and **location** entities. The XLM-R model trained with and without the CRF layer on CoNLL-based CNEC 2.0 [27] was selected as a baseline. The model was used for inference on `tornado`, `2020_names`, and `2022` subsets of TCTV112 dataset. The results of the initial experiments are shown in Table 6.4.

The metrics presented in Table 6.4 combined with an analysis of the utterances on which the model performed poorly provide a lot of information on how to increase its performance. Information about the dataset quality can be assessed based on model performance across different subsets. The following paragraphs will elaborate on these results.

The model pre-trained on CoNLL-based CNEC2.0 [27] dataset performed poorly across all subsets after initial tests (tc variation). This was caused by the fact that the text in the CoNLL-based CNEC2.0 dataset is originally in truecase, and the ASR transcriptions contain all utterances in lowercase with the capital letter at the beginning of the sentence.

Because of this, there is one exception for the tornado subset. Here, the person entity is often at the beginning of the sentence, which makes the majority of names start with a capital letter, causing inflation in the F1 score. Based on these baseline results, the fastest and easiest way to increase performance with the training pipeline that was already implemented is to adapt it to the output format of the used ASR. The CoNLL-based CNEC 2.0 [27] dataset was normalized to lowercase with the capital letter at the beginning of the sentence and combined with the original dataset. This resulted in a dataset twice as large as the original one, where each utterance had a truecase and a lowercase version. The XLM-R model was trained on this dataset with the same settings as described in Section 6.1. The results are shown in Table 6.4 as **tc, lc** variation.

This model achieved better performance across all subsets (with the expected exception in the tornado subset), with a significant difference in location tagging. Compared to the truecase-only model, the difference in F1 was ranging from ≈ 0.4 to ≈ 0.6 for manually labeled datasets.

The greatest increase in performance can be seen on the subsets that are labeled manually. However, the model still achieved poor results for the 2020 subset, where labels were created from the metadata. This was expected due to the nature of the provided metadata and the drawbacks of this simple labeling method, as described in Section 6.2.

Some areas in which the model could improve were discovered by closer examination of precision and recall scores in combination with the model output. The miss-classifications made by the model are mainly caused by the following factors:

- **exclamations in real life scenarios** - the model achieved very low precision scores compared to the recall for the person entity for the tornado subset. This problem arises from the difference between the training and test data, where in the TCTV112 data people in real life situations use a lot of exclamations such as (transcribed to English) “Oh God”, “Jesus Christ”, “Jesus Mary” and so on, probably because the area affected by the tornado is a strongly catholic part of Moravia. The model classified these exclamations as a **person** entity.

- **poor metadata quality** - even though the model classifies many entities correctly, a portion of correct classifications is still considered False Positive during the evaluation due to the fact that not all names were correctly labeled with the simple automatic matching of the metadata against the transcribed text. However, part of this could also be due to the accumulation of errors created by the ASR model causing the content of transcriptions and metadata to not match.

- **lowercase text** - even though fine-tuning the model on lowercase text increased the performance, it also introduced new errors. The model still cannot correctly classify all entities due to the lowercase text plus sometimes there is an occasional mix-up between the street names and name of the person, where the only difference is in the context.

In conclusion, the problems revealed in the baseline experiments are caused by the difference between real data and public NER dataset combined with accumulation of errors in the ASR+NER cascade system. In addition, the metadata alone are not sufficient enough to create good NER labels, and manual correction is needed.

Mistral inference

Mistral was expected to mitigate some of the drawbacks of the XLMR token classification approach. Few-shot inference evaluation was done with the expectation of better context understanding for entity extraction and some level of internal normalization. Several prompts in both English and Czech were tried with a couple of examples. The expected output was a json format containing extracted location and person entities. The example of the used prompt is:

```
Extract the entities in Czech language for the following labels from
the given input sentence text and provide the results in JSON format
   - Entities must be extracted exactly as mentioned in the text.
   - Return each entity under its label without creating new labels.
   - Provide a list of entities for each label, ensuring that if no entities
     are found for a label, an empty list is returned.
   - Accuracy and relevance in your responses are key.
   - Czech words such as: jo, Jo, jojo, naschle, boze are not entities.
   Lables and their Descriptions:
      - PERSON : Extract names
      - LOCATION : Extract location information such as cities
        and street addresses
Input sentence to extract from: INPUT.
```

The **INPUT** was replaced by the actual utterance from the dataset. In addition, a couple of examples of the input and corresponding output were given. Inference was done with the model quantized to 4 bits. Mistral-7B achieved very poor results on this task for several reasons. It was not very consistent with the output JSON format to the point where it was not possible to automatically parse the output into JSON. In such cases, the empty JSON was used as a prediction of the model for the F1 scoring. However, the model did not work very well in these settings as can be seen in Table 6.5 below.

Table 6.5: Evaluation of pre-trained Mistral 7B LLM quantized to 4bits in few-shot settings with prompts and exampels in either Czech or English language. The output was expected to be in JSON format. An empty JSON was used for the evaluation of utterances where the output format was wrong and could not be reconstructed. In these settings a high amount of hallucination was present which corresponds with the obtained results.

Mistral prompts	**Person**			**Location**		
	F1	**P**	**R**	**F1**	**P**	**R**
English + few shot	0.23	0.261	0.205	0.088	0.083	0.093
Czech + few shot	0.179	0.221	0.151	0.139	0.111	0.187

6.3.2 Improving baseline results

The next step is to fine-tune the models on the created in-domain `TCTV112_NER` dataset that is present in Table 6.3, with 56k words and almost 3000 entities. It was created by combining manually annotated subsets and `tornado` subset as described in Section 6.2.1. Creating such dataset brought some technical drawbacks. The sensitivity of the TCTV112 data and how time-consuming it is to manually annotate the data by one or two people. This results

to only a small amount of data divided into training and testing splits. Furthermore, data augmentation was added to the train split to increase performance more even with a limited number of data utterances by addressing some findings made during the baseline evaluation. The augmentation consisted of adding the exclamations into no entity utterances. This process corresponds to the bottom part of Figure 6.2.

The XLM-R model trained on the truecase and lowercase versions of CoNLL-based CNEC 2.0 [27] dataset (tc, lc version of XLM-R in Table 6.6) was further fine-tuned on the created `TCTV112_NER` datatset. Fine-tuning was also done for the CRF version. The model was trained in the same way as in the previous experiments with the learning rate set to $1e^{-5}$, for 10 epochs and early stopping.

The Table 6.6 compares the performance of the XLM-R model variations pre-trained on CoNLL-based CNEC2.0 [27] and models fine-tuned on the real TCTV112 data. Fine-tuning brought significant improvement in F1 score. For both entity types, person and location, it is ≈ 0.2 increase in F1. The XLM-R with additional CRF layer obtained slightly better results similar to the experiments on CoNLL-based CNEC2.0 [27] dataset showed in Table 6.1.

Table 6.6: XLM-R model performance on `TCTV112_NER`. The first three rows represent the XLM-R baselines. These models were pre-trained on CoNLL-based CNEC2.0 [27]. Last two rows show the performance of the model fine-tuned on the created `TCTV112_NER` dataset.

XLM-R variation	**Person**			**Location**		
	F1	**P**	**R**	**F1**	**P**	**R**
tc	0.497	0.767	0.367	0.112	0.4545	0.063
tc,lc	0.677	0.641	0.718	0.658	0.695	0.624
tc,lc, CRF	0.676	0.579	0.812	0.626	0.709	0.560
tc,lc, fine-tuned	0.883	0.895	0.872	0.866	0.848	0.885
tc,lc, CRF fine-tuned	**0.887**	0.903	0.872	**0.870**	0.844	0.898

Mistral fine-tuning

Because of the poor performance of the pre-trained Mistral model (inference results present in Table 6.5), the model was also fine-tuned on the `TCTVT112_NER` dataset. This model was fine-tuned with the QLoRA (described in Section 2.5) approach for 400 steps with a batch size of 4 and the gradient accumulation steps set to 16. It was quantized to 4 bits and the size of LoRA rank was set to 64. The model achieved poor results that were even worse than the pre-trained inference baseline results. These results are shown in Table 6.8. The output of the fine-tuned model was further inspected to clarify what went wrong. The model successfully learned to produce the correct JSON format; however, it suffers from a huge amount of hallucination and effectively stopped performing NER. Instead, the model randomly generates entities seen in the training data or no entities. Examples of the input (TEXT) and output (JSON) of the model are shown below.

```
TEXT: takže.
JSON: {‘‘LOCATION’’: [‘‘hodonína.’’], ‘‘PERSON’’: []}

TEXT: Děkuju za oznámení pane <CENSORED NAME> já to takhle předám.
JSON: {‘‘LOCATION’’: [‘‘lužice.’’], ‘‘PERSON’’: []}
```

The hypothesis is that this was caused by a combination of several factors; too many fine-tuning steps and the small size of the `TCTVT112_NER` dataset with a lot of utterances that do not contain any entities. Assuming a larger dataset would be needed, the fine-tuned XLM-R from Table 6.6 was used as it has sufficient enough performance to automatically label a larger dataset.

A new dataset was created by taking all the transcribed text available at the time, labeling it with the fine-tuned XLM-R model, and removing all utterances that do not contain any entities. The new dataset, shown in Table 6.7, contains 2 million words, 91k **person** entities and 137k **location** entities. Mistral 7B was fine-tuned on this dataset with the same settings as in the previous experiment, besides the number of steps which was set to 1400 steps (trained in $\approx$ 11 hours). The model achieved better results (Table 6.8), especially compared to the first failed experiment. Although performance is nowhere near the token classification approach with XLM-R, it was shown that the LLM of this size can be trained for named entity recognition on emergency calls in a reasonable time. The aim of future experiments will be to improve the fine-tuning hyperparameters, and train the model on all of the available data.

Table 6.7: Comparison of NER datasets created from emergency call recordings used in the fine-tuning experiments. The Labels column shows the method for label creation. **Manual** means that the entities were labeled manually. **Semi-manual** was present only in the tornado subset, where the names and locations were tagged manually but the content of the entities was automatically generated. Labels for `all_transcriptions` (**automatic**) were created solely with the fine-tuned XLM-R model shown in Table 6.6.

Dataset	**Labels**	**Words**	**Number of entities**	
			Person	**Location**
`TCTV112_NER`	semi-manual + manual	55750	1149	1476
`all_transcriptions`	automatic	$\approx$ 2000000	91000	137 000

Table 6.8: Evaluation of fine-tuned 7B Mistral LLM quantized to 4bits. The output was expected to be in the JSON format as in previous baseline experiments. The model overfitted when trained on the `TCTV112_NER` dataset producing a high amount of hallucinations. The model fine-tuned on the dataset created from all available transcriptions slightly improved the results compared to baseline.

Dataset (# words)	**Steps**	**Person**			**Location**		
		F1	**P**	**R**	**F1**	**P**	**R**
`TCTV112_NER` (55k)	0 (baseline)	0.23	0.261	0.205	0.088	0.083	0.093
`TCTV112_NER` (55k)	400	0.079	0.500	0.043	0.036	0.214	0.019
`all transcriptions` (2m)	1400	**0.300**	0.429	0.231	**0.283**	0.317	0.255

Chapter 7

Conclusion

The motivation of this work is to build a system that could help an emergency call center agent process information in a shorter period of time during emergency calls. This involves implementing and training models for the extraction of information from emergency calls. Because real-world emergency calls data are not publicly available or used, the need for a clean dataset created a secondary goal to create such dataset from available metadata and recordings.

7.1 Summary of work

Two different approaches for Named Entity Recognition (NER) on text data were explored. The first is the token classification, as the model classifies every token. The second approach was formulated as the sequence-to-sequence (seq2seq) task. In the seq2seq task, the output of the model is a sequence of tokens instead of class labels. These approaches were evaluated on publicly available Czech NER datasets and served as baselines for the following experiments. The token classification approach with the pre-trained XLM-R [8] model as the backbone achieved a 0.89 F1 score on the CoNLL-based CNEC2.0 [27] dataset, improving previously reported state-of-the-art results.

Next, multiple subsets of the labeled data were created from real emergency call recordings and the corresponding metadata. This included running an ASR model to create transcriptions and creating NER labels. The quality of the data provided was not the best and a manual label correction was needed to create a clean dataset suited for neural network fine-tuning. The implemented pipeline is ready to process more data if available and utilize reliable fine-tuned models to automatically create labels. The final clean dataset consists of 56k words and almost 3000 entities. It was used for fine-tuning of the chosen models on real world data. I successfully managed to improve the performance of the XLM-R based model for **name** classification from 0.49 F1 to 0.89 F1 and for **location** classification from 0.11 F1 to 0.87 F1. Additionally, a couple of LLM fine-tuning experiments were conducted on created emergency calls datasets. It is important to have a large enough dataset to fine-tune LLMs. Because of this, a new dataset containing 2 million words was created. The Mistral 7B model was successfully fine-tuned, improving its baseline results, but still lacking in performance compared to the XLM-R model.

7.2 Future work

For future work, the goal is to create a larger dataset from all available data, run more experiments and continue to improve Mistral 7B or other LLM. This should be possible by tuning the training hyperparameters in combination with the larger dataset. In addition, the plan is to implement multi-modal models to take advantage of speech and text data, potentially improving the results and be able to do NER directly from speech.

This work serves as a stepping stone for the deployment of neural network models in areas such as emergency services. Even if these models are currently not used in emergency call centers, they present a great tool to create and expand the current dataset in this domain. With more data and LLM capabilities, it is possible to build a model or system that will not only perform NER, but could be a foundation for a conversational system that will assist the emergency line agents in real time.

Bibliography

[1] Aggarwal, C. *Neural Networks and Deep Learning: A Textbook*. 1st ed. January 2018. ISBN 978-3-319-94462-3.

[2] Arkhipov, M., Trofimova, M., Kuratov, Y. and Sorokin, A. Tuning Multilingual Transformers for Language-Specific Named Entity Recognition. In: Erjavec, T., Marcińczuk, M., Nakov, P., Piskorski, J., Pivovarova, L. et al., ed. *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*. Florence, Italy: Association for Computational Linguistics, August 2019, p. 89–93. DOI: 10.18653/v1/W19-3712. ISBN 978-1-945626-45-6. Available at: https://aclanthology.org/W19-3712.

[3] Bishop, C. *Pattern Recognition and Machine Learning*. 1st ed. Springer, January 2006. ISBN 978-0-387-31073-2. Available at: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/.

[4] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D. et al. Language Models are Few-Shot Learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. and Lin, H., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020, vol. 33, p. 1877–1901. ISBN 9781713829546. Available at: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[5] Chang, C.-H., Kayed, M., Girgis, M. and Shaalan, K. A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering*. 1st ed. 2006, vol. 18, no. 10, p. 1411–1428. DOI: 10.1109/TKDE.2006.152.

[6] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G. et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*. 2023, vol. 24, no. 240, p. 1–113.

[7] Chung, J., Gülçehre, Ç., Cho, K. and Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*. 1st ed. 2014, abs/1412.3555, no. 1. Available at: http://arxiv.org/abs/1412.3555.

[8] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G. et al. Unsupervised Cross-lingual Representation Learning at Scale. In: Jurafsky, D., Chai, J., Schluter, N. and Tetreault, J., ed. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, p. 8440–8451. DOI:

10.18653/v1/2020.acl-main.747. Available at: https://aclanthology.org/2020.acl-main.747.

[9] Cowie, J. and Lehnert, W. Information Extraction. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. jan 1996, vol. 39, no. 1, p. 80–91. DOI: 10.1145/234173.234209. ISSN 0001-0782. Available at: https://doi.org/10.1145/234173.234209.

[10] Dai, A. M. and Le, Q. V. Semi-supervised Sequence Learning. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M. and Garnett, R., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, vol. 28. ISBN 9781510825024. Available at: https://proceedings.neurips.cc/paper_files/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

[11] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. et al. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In: Korhonen, A., Traum, D. and Màrquez, L., ed. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, p. 2978–2988. DOI: 10.18653/v1/P19-1285. Available at: https://aclanthology.org/P19-1285.

[12] Derczynski, L. Complementarity, F-score, and NLP Evaluation. In: Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M. et al., ed. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, p. 261–266. ISBN 978-2-9517408-9-1. Available at: https://aclanthology.org/L16-1040.

[13] Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L. QLoRA: Efficient Finetuning of Quantized LLMs. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2023, vol. 36, p. 10088–10115. Available at: https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf.

[14] Devlin, J., Chang, M., Lee, K. and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Burstein, J., Doran, C. and Solorio, T., ed. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, p. 4171–4186. DOI: 10.18653/V1/N19-1423. ISBN 978-1-950737-13-0. Available at: https://doi.org/10.18653/v1/n19-1423.

[15] Feng, F., Yang, Y., Cer, D., Arivazhagan, N. and Wang, W. Language-agnostic BERT Sentence Embedding. In: Muresan, S., Nakov, P. and Villavicencio, A., ed. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, p. 878–891. DOI: 10.18653/v1/2022.acl-long.62. Available at: https://aclanthology.org/2022.acl-long.62.

[16] Frantar, E., Ashkboos, S., Hoefler, T. and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *ArXiv preprint arXiv:2210.17323*. 2022.

[17] Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P. et al. AllenNLP: A Deep Semantic Natural Language Processing Platform. In: Park, E. L., Hagiwara, M., Milajevs, D. and Tan, L., ed. *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, p. 1–6. DOI: 10.18653/v1/W18-2501. Available at: https://aclanthology.org/W18-2501.

[18] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. 1st ed. MIT Press, 2016. ISBN 978-0262035613. http://www.deeplearningbook.org.

[19] He, P., Gao, J. and Chen, W. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. *CoRR*. 2021, abs/2111.09543. Available at: https://arxiv.org/abs/2111.09543.

[20] Hobbs, J. R. and Riloff, E. Information Extraction. *Handbook of natural language processing*. Citeseer. 2010, vol. 15, p. 16.

[21] Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. 1997, vol. 9, no. 8, p. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[22] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y. et al. LoRA: Low-Rank Adaptation of Large Language Models. *CoRR*. 2021, abs/2106.09685. Available at: https://arxiv.org/abs/2106.09685.

[23] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S. et al. Mistral 7B. *ArXiv preprint arXiv:2310.06825*. 2023.

[24] Jordan, M. I. and Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science*. American Association for the Advancement of Science. 2015, vol. 349, no. 6245, p. 255–260.

[25] Jurafsky, D. and Martin, J. H. *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2023. Available at: https://web.stanford.edu/~jurafsky/slp3/ed3bookfeb3_2024.pdf.

[26] Karafiát, M., Veselý, K., Černocký, J., Profant, J., Nytra, J. et al. Analysis of X-Vectors for Low-Resource Speech Recognition. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE Signal Processing Society, 2021, p. 6998–7002. DOI: 10.1109/ICASSP39728.2021.9414725. ISBN 978-1-7281-7605-5. Available at: https://www.fit.vut.cz/research/publication/12525.

[27] Konkol, M. and Konopík, M. CRF-Based Czech Named Entity Recognizer and Consolidation of Czech NER Research. In: Habernal, I. and Matoušek, V., ed. *Text, Speech, and Dialogue*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, p. 153–160. ISBN 978-3-642-40585-3.

[28] Konkol, M. and Konopík, M. Segment Representations in Named Entity Recognition. In: Král, P. and Matoušek, V., ed. *Text, Speech, and Dialogue*. Cham: Springer International Publishing, 2015, p. 61–70. ISBN 978-3-319-24033-6.

[29] Lample, G. and Conneau, A. Cross-lingual Language Model Pretraining. *CoRR*. 2019, abs/1901.07291. Available at: http://arxiv.org/abs/1901.07291.

[30] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. Available at: https://openreview.net/forum?id=H1eA7AEtvS.

[31] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A. et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv preprint arXiv:1910.13461*. 2019.

[32] Li, J., Sun, A., Han, J. and Li, C. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*. IEEE. 2020, vol. 34, no. 1, p. 50–70.

[33] Li, S., Ning, X., Wang, L., Liu, T., Shi, X. et al. Evaluating Quantized Large Language Models. *ArXiv preprint arXiv:2402.18158*. 2024.

[34] Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M. et al. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In: *MLSys*. 2024.

[35] Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z. et al. Self-Supervised Learning: Generative or Contrastive. *IEEE Transactions on Knowledge and Data Engineering*. 2023, vol. 35, no. 1, p. 857–876. DOI: 10.1109/TKDE.2021.3090866.

[36] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*. 2019, abs/1907.11692. Available at: http://arxiv.org/abs/1907.11692.

[37] Mikheev, A., Moens, M. and Grover, C. Named entity recognition without gazetteers. In: Thompson, H. S. and Lascarides, A., ed. *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. 1999, p. 1–8.

[38] Mikolov, T., Yih, W.-t. and Zweig, G. Linguistic Regularities in Continuous Space Word Representations. In: Vanderwende, L., Daumé III, H. and Kirchhoff, K., ed. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, p. 746–751. Available at: https://aclanthology.org/N13-1090.

[39] OpenAI. *GPT-3 Language Model* [Response generated by GPT-3]. February 2024. Available at: https://openai.com/index/gpt-3-apps.

[40] Pan, X., Zhang, B., May, J., Nothman, J., Knight, K. et al. Cross-lingual Name Tagging and Linking for 282 Languages. In: Barzilay, R. and Kan, M.-Y., ed. *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers). Vancouver, Canada: Association for Computational Linguistics, July 2017, p. 1946–1958. DOI: 10.18653/v1/P17-1178. Available at: https://aclanthology.org/P17-1178.

[41] Paolini, G., Athiwaratkun, B., Krone, J., Ma, J., Achille, A. et al. Structured Prediction as Translation between Augmented Natural Languages. *CoRR*. 2021, abs/2101.05779. Available at: https://arxiv.org/abs/2101.05779.

[42] Pascanu, R., Mikolov, T. and Bengio, Y. On the difficulty of training recurrent neural networks. In: Dasgupta, S. and McAllester, D., ed. *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, vol. 28, no. 3, p. 1310–1318. Proceedings of Machine Learning Research. Available at: https://proceedings.mlr.press/v28/pascanu13.html.

[43] Piskorski, J. and Yangarber, R. Information Extraction: Past, Present and Future. In: Poibeau, T., Saggion, H., Piskorski, J. and Yangarber, R., ed. *Multi-source, Multilingual Information Extraction and Summarization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, p. 23–49. DOI: 10.1007/978-3-642-28569-1_2. ISBN 978-3-642-28569-1. Available at: https://doi.org/10.1007/978-3-642-28569-1_2.

[44] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. et al. Language models are unsupervised multitask learners. *OpenAI blog*. 2019, vol. 1, no. 8, p. 9.

[45] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S. et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 1st ed. JMLR.org. jan 2020, vol. 21, no. 1. ISSN 1532-4435.

[46] Rahimi, A., Li, Y. and Cohn, T. Multilingual NER Transfer for Low-resource Languages. *CoRR*. 2019, abs/1902.00193. Available at: http://arxiv.org/abs/1902.00193.

[47] Ringland, N., Dai, X., Hachey, B., Karimi, S., Paris, C. et al. NNE: A Dataset for Nested Named Entity Recognition in English Newswire. In: Korhonen, A., Traum, D. and Màrquez, L., ed. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, p. 5176–5181. DOI: 10.18653/v1/P19-1510. Available at: https://aclanthology.org/P19-1510.

[48] Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization (1958). In: *Deas That Created the Future: Classic Papers of Computer Science*. February 2021, p. 183–190. DOI: 10.7551/mitpress/12274.003.0020. ISBN 9780262363174.

[49] Ruder, S. An overview of multi-task learning in deep neural networks. *ArXiv preprint arXiv:1706.05098*. 2017.

[50] Ruis, L. *Structured Prediction part one - Deriving a Linear-chain CRF* [online]. 2021 [cit. 2024-05-11]. Available at: https://lauraruis.github.io/2021/01/25/crfpt1.html.

[51] Sennrich, R., Haddow, B. and Birch, A. Neural Machine Translation of Rare Words with Subword Units. In: Erk, K. and Smith, N. A., ed. *Proceedings of the*

54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, August 2016, p. 1715–1725. DOI: 10.18653/v1/P16-1162. Available at: https://aclanthology.org/P16-1162.

[52] Ševčíková, M., Žabokrtský, Z. and Krůza, O. Named Entities in Czech: Annotating Data and Developing NE Tagger. In: Matoušek, V. and Mautner, P., ed. *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*. Berlin / Heidelberg: Springer, 2007, vol. 4629, XVII, p. 188–195. Lecture Notes in Computer Science. ISBN 978-3-540-74627-0.

[53] Shaw, P., Uszkoreit, J. and Vaswani, A. Self-Attention with Relative Position Representations. *CoRR*. 2018, abs/1803.02155. Available at: http://arxiv.org/abs/1803.02155.

[54] Sido, J., Pražák, O., Přibáň, P., Pašek, J., Seják, M. et al. Czert – Czech BERT-like Model for Language Representation. In: Mitkov, R. and Angelova, G., ed. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Held Online: INCOMA Ltd., September 2021, p. 1326–1338. Available at: https://aclanthology.org/2021.ranlp-1.149.

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

[56] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A. et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint arXiv:2307.09288*. 2023.

[57] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is All you Need. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, vol. 30. ISBN 9781510860964. Available at: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[58] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R. et al. MT5: A massively multilingual pre-trained text-to-text transformer. *CoRR*. 2020, abs/2010.11934. Available at: https://arxiv.org/abs/2010.11934.

[59] Yenduri, G., M, R., G, C. S., Y, S., Srivastava, G. et al. *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. 2023.

[60] Zhang, Y. and Yang, Q. An overview of multi-task learning. *National Science Review*. 1st ed. september 2017, vol. 5, no. 1, p. 30–43. DOI: 10.1093/nsr/nwx105. ISSN 2095-5138. Available at: https://doi.org/10.1093/nsr/nwx105.

Appendix A

TCTV112 data sentence

```
<s:Envelope>
  <hlavicka xmlns="">
    <idDatovaVeta>s8g9k-3w7j4p1x-2q6r5y0z1m-2n3b4</idDatovaVeta>
    <datumVytvoreni>2020-02-17T05:52:56.127+01:00</datumVytvoreni>
    <odesilatel>
      <id>s8g9k-3w7j4p1x-2q6r5y0z1m-2n3b4</id>
      <kod>TOLK</kod>
    </odesilatel>
    <adresat>
      <id>s8g9k-3w7j4p1x-2q6r5y0z1m-2n3b4</id>
      <kod>HOLK</kod>
    </adresat>
  </hlavicka>
  <teloDatovaVeta xmlns="">
    <idUdalost>s8g9k-3w7j4p1x-2q6r5y0z1m-2n3b4</idUdalost>
    <typUdalosti>NEM</typUdalosti>
    <oznamovatel>
      <jmeno>ANONYMIZED</jmeno>
      <prijmeni>ANONYMIZED</prijmeni>
    </oznamovatel>
    <poznamka>
      <timestamp>2020-02-17T05:52:50.233+01:00</timestamp>
      <text>V blizkosti se nachazi dalsi OA,
      uvnitr obytneho privesu by se nemel nikdo nachazet.
      </text>
    </poznamka>
    <mistoUdalosti>
      <dopresneniMista>pod mostem mezi ANONYMIZED </dopresneniMista>
      <poloha>
        <x>ANONYMIZED</x>
        <y>ANONYMIZED</y>
        <srid>ANONYMIZED</srid>
      </poloha>
      <urceniPolohy>DBA</urceniPolohy>
```

```
    </mistoUdalosti>
  </teloDatovaVeta>
<s:Envelope>
```

Listing A.1: Example of data sentence in XML format provided as metadata with the emergency call recordings. The most important elements are; **oznamovatel** - contains name of the caller, **mistoUdalosti** - containing information on the location of the emergency, **text** for additional information, and **typUdalosti** which is an event type used to select recordings of interest. The event types are 3-letter abbreviations, e.g., NEM for "nemoc" meaning illness. Data sentences stored in CSV files contain similar information stored in CSV format.