



**Czech
Technical University
in Prague**

F3

Faculty of Electrical Engineering
Department of Cybernetics

Named Entity Recognition in Czech

Bachelor's Thesis of
Radek Štulc

Supervisor: **Ing. Herbert Ullrich**
Study program: **Open Informatics**
Specialization: **Artificial Intelligence and Computer Science**
May 2024

I. Personal and study details

Student's name: **Štulc Radek** Personal ID number: **507413**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Cybernetics**
Study program: **Open Informatics**
Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Named Entity Recognition in Czech

Bachelor's thesis title in Czech:

Rozpoznávání pojmenovaných entit v českém jazyce

Guidelines:

The overall task of this thesis is to research the Natural-Language-Processing challenge of Named Entity Recognition, its solutions based on Language Models with Transformer architecture and their application to Czech training data to obtain and publish a re-useable solver for the task.

- 1) Explore state-of-the-art machine learning methods dealing with the task of Named Entity Recognition and other related NLP tasks in Czech language and in English (with the focus on their reproducibility in Czech and other Slavonic languages).
- 2) Perform a comparative analysis of the available data and solution architectures.
- 3) Based on the findings, train multiple classifiers for the task and compare their performance.
- 4) Perform an empirical analysis of how your best-performing solution performs on general real-world data (such as news articles), describe your findings, and propose how to address possible issues.
- 5) Release the model publicly for a streamlined open usage.

Bibliography / sources:

- [1] Jana Straková, Milan Straka, and Jan Hajic: "Neural Architectures for Nested NER through Linearization". In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics
- [2] Ikuyi Yamada et al.: "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention" (2020). Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)
- [3] Besnik Fetahu, Sudipta Kar, Zhiyu Chen, Oleg Rokhlenko, and Shervin Malmasi. 2023. "SemEval-2023 Task 2: Fine-grained Multilingual Named Entity Recognition (MultiCoNER 2)". In Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)

Name and workplace of bachelor's thesis supervisor:

Ing. Herbert Ullrich Department of Computer Science FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **19.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Herbert Ullrich
Supervisor's signature

prof. Dr. Ing. Jan Kybic
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

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

Date of assignment receipt

Student's signature

Acknowledgements

I would like to thank my supervisor Herbert Ullrich for his kind guidance, valuable feedback, and assistance during my work. I also thank Dr. Jana Straková for her helpful e-mail correspondence regarding the used dataset.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics” is also gratefully acknowledged.

Declaration

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

In Prague, 24. May 2024
Radek Štulc

Abstract

Named Entity Recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories, such as person names, ages, dates, organizations, etc. It is one of the main techniques in Natural Language Processing (NLP), and a strong entity classifier is an inseparable part of every research, where the correct representation of entities is essential. In Czech, the biggest publicly available corpus is the Czech Named Entity Corpus. This corpus contains nested entities and therefore all state-of-the-art models trained on this corpus solve the problem of nested NER. In this thesis, I modified this corpus to contain only flat entities to solve the NER problem using the Hugging Face libraries with a focus on the replicability of training and creating a simple yet strong classifier. The classifier is based on a Czech monolingual version of RoBERTa, a robustly optimized BERT pre-training approach, named RobeCzech. The solution proposed in the thesis exceeds the known state-of-the-art results and seems to scale well with real-world data.

Keywords: Named Entity Recognition, NER, nested NER, Transformers, Czech Named Entity Corpus, CNEC

Supervisor: Ing. Herbert Ullrich

Abstrakt

Rozpoznávání pojmenovaných entit (NER) je dílčím úkolem “information extraction”, který se snaží najít a klasifikovat pojmenované entity uvedené v nestrukturovaném textu do předem definovaných kategorií, jako jsou jména osob, věk, data, organizace atd. Je to jedna z hlavních technik “Natural Language Processing” (NLP) a silný klasifikátor entit je nedílnou součástí každého výzkumu, kde je správná reprezentace entit zásadní. V češtině je největším veřejně dostupným korpusem “Czech Named Entity Corpus”. Tento korpus obsahuje vnořené entity, a proto všechny nejmodernější modely natrénované na tomto korpusu řeší problém vnořného NER. V této práci jsem tento korpus upravil tak, aby obsahoval pouze nevnořené entity pro řešení problému NER pomocí knihoven “Hugging Face” se zaměřením na replikovatelnost trénování a vytvoření jednoduchého, ale silného klasifikátoru. Klasifikátor je založen na české jednojazyčné verzi RoBERTa, robustně optimalizovaném tréniku založeném na architektuře “BERT”, jménem RobeCzech. Řešení navržené v této práci překračuje známé state-of-the-art výsledky a zdá se, že je dobře škálovatelné s reálnými daty.

Klíčová slova: Rozpoznávání pojmenovaných entit, NER, nested NER, Transformers, Czech Named Entity Corpus, CNEC

Contents

1 Introduction	1
1.1 Motivations	2
1.2 Main Challenges	2
1.3 Introduction to Transformers	2
1.4 Thesis outline	3
2 State-of-the-Art Overview	5
2.1 Named Entity Recognition in English	5
2.2 Named Entity Recognition in Czech	5
2.2.1 NameTag 2	6
2.2.2 Czert	6
2.2.3 SlavicBERT	7
2.2.4 RobeCzech	7
2.2.5 XLM-RoBERTa	8
2.3 Metrics	8
3 Dataset	11
3.1 Overview	11
3.1.1 CNEC 1.0	11
3.1.2 CNEC 1.1	11
3.1.3 CNEC 2.0	12
3.1.4 CoNLL-based extended	14
3.2 Representation	14
3.2.1 Conversion	14
3.2.2 Labeling	15
4 Model training	19
4.1 Data Preprocessing	19
4.1.1 Conversion	19
4.1.2 Loading Scripts	20
4.1.3 Tokenize and Align	21
4.2 Training Loop	22
4.2.1 Flat NER	22
4.2.2 Nested NER	24
4.3 Evaluation	25
5 Empirical Analysis	29
5.1 CoNLL-based extended	29
5.2 CNEC 2.0 Supertypes	29
5.3 CNEC 2.0 Types	30
5.4 Nested NER	30
5.5 Results of analysis	30
6 Conclusion	35
6.1 Main Contributions	36
6.2 Future Works	36
Bibliography	37
A Acronyms	41
B Repository Structure	43

Figures

1.1 A simple example of raw input ..	1
1.2 A simple example of classified input	1
1.3 A simple example of classified input with nested entities, P stands for person, P-f for first name, P-s for surname, N for number, and G for geographical location	1
1.4 An architecture of the Transformer model (reprinted from [Vaswani et al., 2017])	4
3.1 CNEC 1.1 named entities type hierarchy	12
3.2 CNEC 2.0 named entities type hierarchy	13
3.3 CNEC 1.1 named entities type hierarchy excluding unused entities	17
4.1 A Heaviside step function	25
5.1 A screenshot of an example of RobeCzech fine-tuned on CNEC 2.0 CoNLL extended.	30
5.2 A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 CoNLL extended	31
5.3 A screenshot of an example of RobeCzech fine-tuned on CNEC 2.0 Supertypes	32
5.4 A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 only supertypes	32
5.5 A screenshot of an example of RobeCzech fine-tuned on CNEC 2.0 only types	33
5.6 A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 only types	33
5.7 A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 all nested types	34
6.1 An example of ChatGPT answer on NER problem	36

Tables

4.1 Pre-tokenized labeled input	21
4.2 Tokenized and aligned input ...	21
4.3 An example of one-hot encoded label for one token	22
4.4 F score comparison of models trained to solve the NER problem.	26
4.5 F score comparison of trained models to solve the nested NER problem	26

Chapter 1

Introduction

The human ability to communicate through natural language has fascinated scientists for many years. After all, only because of this ability can humans understand each other, convey their feelings and information, and this was the main reason for our ever-growing evolution [MacWhinney, 2005, Mithen, 1997]. With the invention of computing devices, a new challenge arose including this topic; making machines able to understand language the same way humans do. Natural Language Processing (NLP) is an interdisciplinary subfield of computer science and information retrieval primarily concerned with solving this challenge using the most recent machine learning approaches. The goal is a computer capable of understanding human language, including the contextual nuances that language brings.

For machines to understand the language, we created many tasks to help the computer understand the context. One of these tasks is information extraction (IE), which automatically extracts structured information from unstructured or semi-structured machine-readable inputs.

IE is made up of various subtasks depending on the problem these tasks solve, one being the topic of this thesis **Named Entity Recognition (NER)**. NER is a task that seeks to locate named entities and classify them in unstructured text into pre-defined categories such as person names, organization, numbers, dates, etc. A simple example of this classification can be seen in Figure 1.1 and in Figure 1.2. In Czech NER the current state-of-the-art solution is called **NameTag 2**, introduced in [Straková et al., 2019] paper, fine-tuned on a dataset named **Czech Named Entity Corpus (CNEC)** [Ševčíková et al., 2007a] (explained in Chapter 3), but this solution solves a closely related problem called Nested Named Entity Recognition, where named entities can now be embedded inside each other creating nested entities, the example for this problem is shown in Figure 1.3.

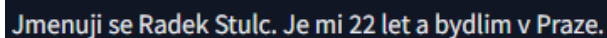


Figure 1.1: A simple example of raw input

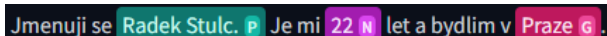


Figure 1.2: A simple example of classified input, P stands for person, N stands for number, and G is for geographical location



Figure 1.3: A simple example of classified input with nested entities, P stands for person, P-f for first name, P-s for surname, N for number, and G for geographical location

1.1 Motivations

In today's world, almost everyone is connected through the Internet and uses it daily to gather information. No one would be surprised that not all this information is factually correct, creating misinformation and the danger of spreading such misinformation. To combat this, many journalists began to fact-check claims, newspaper articles, or videos. The NLP team at the Artificial Intelligence Center¹ led by Jan Drchal decided to help them by creating an automated fact-checking platform [Drchal et al., 2023]. This platform requires several strong classifiers to understand the context of information, including the named entity classifier. NameTag 2 offers such a classifier, but as stated above, this model is trained using nested entities. This may seem like a trivial problem, but it can cause major obstacles, as the embedding can grow quickly and some entities may prove to be more confusing than helpful, for example, in the name of town "Ústí nad Labem" the river entity "Labem" may not be that important for the understanding of the entire sentence. As the models for Czech NER were either not publicly available or solved nested NER, I was tasked with creating a dataset that contains the same data as the CNEC dataset but does not contain nested entities, and creating a machine learning model that produces near-human results fine-tuned in this dataset.

1.2 Main Challenges

The main challenges of this thesis are the already mentioned creation of a dataset containing only flat entities and the creation of a machine learning model solving NER. All available solutions as of now for Czech are complex architectures combining the Transformer architecture with other neural networks. I focused on creating a publicly available model using Hugging Face libraries² using their training API and solving the NER problem in a way that is easily replicable and using a single model based on the Transformers architecture.

Hugging Face is a large machine learning and data science collaboration platform that helps users build, deploy, and train machine learning models. The creation of a complex training loop for NER could be a topic for a whole thesis, and we could still not be as efficient and effective as using a training loop offered by the Hugging Face community that is configured and optimized for the best results. To achieve the best results, I decided to use this platform and all models and datasets were made publicly available.

The offered solution should surpass the available state-of-the-art results, and the model should translate well into real-world situations, as testified by an empirical analysis.

In addition to the NER classifier, I also decided to train a model capable of comprehending nested entities to acquire a deeper understanding of the NER problem.

1.3 Introduction to Transformers

In the past, dominant models solving NLP tasks were based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best-performing models would also connect the encoder and decoder through an attention mechanism. In the [Vaswani et al., 2017] paper, a new simple network architecture was introduced, able

¹<https://www.aic.fel.cvut.cz/>

²<https://huggingface.co/>

to solve sequence-to-sequence problems based solely on attention mechanisms, dispensing the recurrent and convolutional networks completely; the Transformers. This has been a major breakthrough in the field and gave birth to famous models such as BERT [Devlin et al., 2019], ELMO [Peters et al., 2018] or GPT-3 [Brown et al., 2020], which quickly outperformed other models, and recent research built on these models pushed current state-of-the-art near or even outperforming human results, and this research will use the Transformer architecture as well, so I find appropriate to describe this architecture.

The attention mechanism is a method that simulates how human attention works by assigning varying levels of importance to different words in a sentence. Attention assigns importance to each word by calculating weights for the word's embeddings within a specific sentence section called the context window to determine its significance. These weights can adapt and change with each training.

This architecture offers many advantages over other solutions. The biggest is the increase in performance in many NLP tasks including NER. Another big advantage is the available use of parallelism, as the solution does not treat the input step by step, and this architecture battles one of the biggest disadvantages of recurrent neural networks, which prioritize more recent information over the old.

Masked Language Modeling (MLM) is a method that trains attention by masking some parts of a sentence and forcing the model to depend on other words to predict the masked one and making the model using attention better understand sentences based on the context.

The Transformer architecture is depicted in Figure 1.4.

1.4 Thesis outline

My thesis is divided into 6 chapters. A succinct description of each chapter can be seen below:

- **Chapter 1** introduces the thesis, its motivation, main goals, and challenges
- **Chapter 2** explores state-of-the-art methods for NER and describes the pre-trained models to be used.
- **Chapter 3** explains the CNEC dataset used for fine-tuning and the problem named nested NER which will also be researched.
- **Chapter 4** contains details on the training and evaluation of the pre-trained models.
- **Chapter 5** offers an empirical analysis of the two best performing models
- **Chapter 6** concludes the thesis.

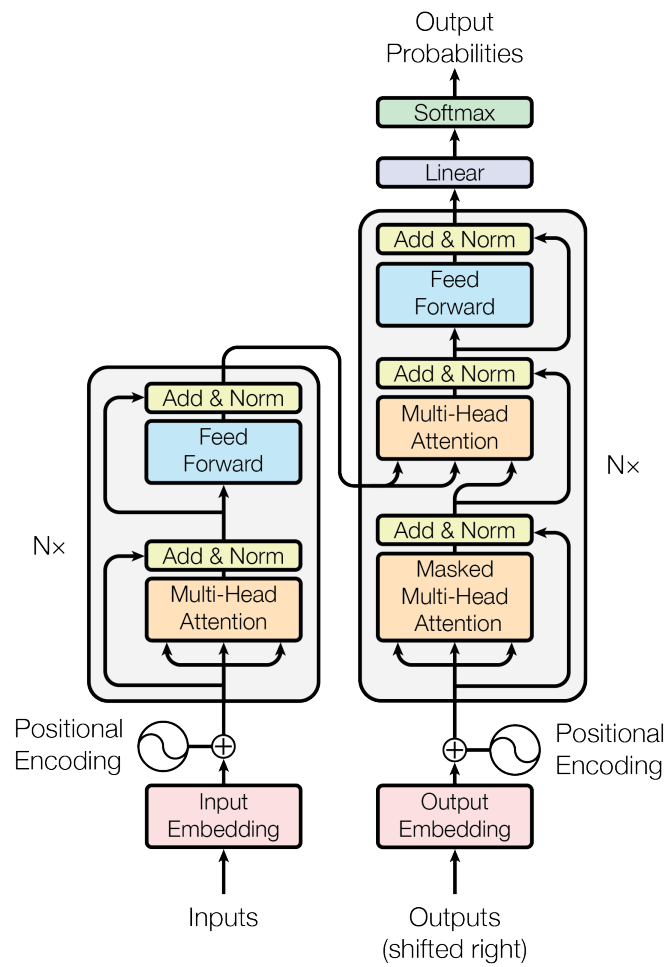


Figure 1.4: An architecture of the Transformer model (reprinted from [Vaswani et al., 2017])

Chapter 2

State-of-the-Art Overview

This chapter explores current state-of-the-art methods used in Named Entity Recognition in English and Czech. It introduces and describes pre-trained models used in this thesis for fine-tuning in Czech. Lastly, I explain the metrics used for the observation of the model performance.

2.1 Named Entity Recognition in English

Named Entity Recognition in English is already a well-researched topic with many fine-tuned models exceeding 90% F_1 score. The application of deep learning methods in NLP was made possible by the introduction of word embeddings [Mikolov et al., 2013], which represent words as vectors in a low-dimensional continuous space, capturing their semantic relations. However, a problem occurred: depending on the context, one word could have more meanings. This word would have the same embedding for all homonyms. Deep neural models based on Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] and Transformer [Vaswani et al., 2017] architecture, such as ELMo [Peters et al., 2018] and BERT [Devlin et al., 2019] have introduced contextualized word representation (CWR)(also known as contextualized embeddings), making word vectors sensitive to context. This allowed the creation of many language models trained using a task equivalent to or similar to MLM, which could be used for a wide range of downstream NLP tasks, namely, the NER task, where these models achieve the F1 scores mentioned. The Transformer architecture is briefly explained in Section 1.3.

Many models use different methods to boost the F1 score as high as possible and beat other models. An interesting model is LUKE [Yamada et al., 2020], which treats not only words (similar to existing CWR models) but also entities as independent tokens so that the entities would have two embeddings, one a token embedding and the other an entity embedding.

Recent works also found better word representations by concatenating different types of embeddings. [Wang et al., 2021] proposed an Automated Concatenation of Embeddings (ACE) to automate the process of finding a better concatenation of embeddings.

2.2 Named Entity Recognition in Czech

In Czech, Named Entity Recognition is a less researched task than in English, and the models do not reach such high F1 scores. In 2024, the state-of-the-art is **NameTag 2** [Straková et al., 2019], which will be my benchmark for my fine-tuned models, and its scores can be seen in Table 4.5. Furthermore, [Sido et al., 2021] trained the first Czech

monolingual language model **Czert** based on the architectures of BERT [Devlin et al., 2019] and ALBERT [Lan et al., 2020] (86.7 F_1 score on the [Konkol et al., 2014] version of CNEC 1.1 dataset). [Sido et al., 2021] also compared their model with mBERT [Devlin et al., 2019] and SlavicBERT [Arkhipov et al., 2019]. The last model, I have found, that achieves state-of-the-art results is **RobeCzech** [Straka et al., 2021] (87.82 F_1 score on CNEC 1.1).

Moreover, a recent large bert-like model, which includes Czech, is **XLM-RoBERTa** [Conneau et al., 2020]. XLM-RoBERTa is a large multilingual language model based on RoBERTa [Liu et al., 2019] architecture. This model can be trained using a downstream dataset described in Chapter 3.

A detailed description of each model I will fine-tune and evaluate can be seen in the following subsections.

2.2.1 NameTag 2

NameTag 2 [Straková et al., 2019] is a neural network architecture for nested NER. Nametag 2 is divided into two different neural models:

- The first model uses a standard LSTM-CRF [Lample et al., 2016] architecture, where the input, which is a concatenation of multiple labels of a nested entity, is sent into an LSTM neural network and then decoded by a conditional random field (CRF). Although this approach is simple and effective, the obvious disadvantage is the fast-increasing number of named entities, which makes this model less effective for the problem of nested NER.
- In the second model, the nested entities are represented as a sequence, viewing the task as a sequence-to-sequence (seq2seq) [Sutskever et al., 2014] task. The most popular architecture used for this task is the encoder-decoder architecture, where the encoder is a bidirectional LSTM [Cui et al., 2019] and the decoder is an LSTM. The tokens are viewed as the input sequence and the encoded labels are predicted one by one by the decoder until the decoder outputs the “<eow>” (end of word) label and moves to the next token.

The network was trained using a variant of Adam optimizer [Kingma and Ba, 2014], which only updates the accumulators for the variables that appear in a current batch¹, with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.98$. They used mini-batches of size 8. For regularization, they applied a dropout rate of 0.5 and the word dropout replaces 20% of words by the unknown token to force the network to rely more on context. They also add contextual word embeddings from ELMo [Peters et al., 2018], BERT [Devlin et al., 2019], and Flair [Akbik et al., 2018] to word- and character-level embeddings to reach even further improvements.

2.2.2 Czert

As mentioned, **Czert** [Sido et al., 2021] is the first Czech monolingual language representation model based on the BERT [Devlin et al., 2019] and ALBERT [Lan et al., 2020] architectures. As stated in Section 2.1, BERT is a model based on Transformer architecture with two pre-training tasks: *Masked Language Modeling (MLM)* and *Next Sentence Prediction (NSP)*. In the original paper [Devlin et al., 2019] the author published BERT_{BASE} and BERT_{LARGE} where BERT_{LARGE} contains 3 times the parameters

¹https://www.tensorflow.org/addons/tutorials/optimizers_lazyadam

of $BERT_{BASE}$. Later, ALBERT [Lan et al., 2020] proposed a way to use the parameters more effectively and reduce their number while maintaining a similar performance, while also dividing into $ALBERT_{BASE}$, $ALBERT_{LARGE}$, $ALBERT_{XLARGE}$, $ALBERT_{XXLARGE}$. Czert also followed this convention and created two models:

- **Czert-A** is very similar to the standard $ALBERT_{BASE}$.
- **Czert-B** is configured exactly as $BERT_{BASE}$.

Both of the models are trained on a text corpus consisting of:

- SYN v4 [Křen et al., 2016], a large corpus of contemporary written Czech, 4,188M tokens.
- Czes [Anonymous, 2011], a collection of Czech newspaper and magazine articles, 432M tokens.
- Plain texts extracted from the Czech Wikipedia dump using WikiExtractor [Attardi, 2015], 123M tokens.

Both models are trained with a learning rate of $1 \cdot 10^{-4}$ and linear decay using the Adam optimizer [Kingma and Ba, 2014].

For my experiments, I fine-tuned only Czert-B and SlavicBERT on a downstream dataset from Chapter 3, as they perform the best as shown in the paper [Sido et al., 2021].

■ 2.2.3 SlavicBERT

SlavicBERT [Arhipov et al., 2019] is based on a multilingual version of the mBERT [Devlin et al., 2019] model focused on four Slavic languages: Russian, Bulgarian, Polish, and Czech. The original BERT embedder is already trained on 104 languages. However, for the four languages, they did not need the entire inventory of multilingual subtokens, and the original WordPiece tokenization may lack Slavic-specific ngrams, which could result in a longer input sequence. Hence, they retrained BERT in the stratified Wikipedia for Czech, Polish, and Bulgarian, and News data for Russian. They initialize the model with the multilingual BERT, as training from scratch would be extremely computationally expensive.

After pre-training and fine-tuning on a downstream dataset, their model outperforms multilingual BERT, and by adding a CRF layer, the model is achieving even better performance.

■ 2.2.4 RobeCzech

RobeCzech [Straka et al., 2021] is a Czech contextualized language representation model based on the Transformer architecture trained solely on Czech data. RobeCzech is more accurately a Czech monolingual version of RoBERTa [Liu et al., 2019], a robustly optimized BERT pre-training approach. RobeCzech was trained on four publicly available texts. Three of them are the same as for Czert described in Section 2.2.2 and one added:

- Documents with at least 400 tokens from the Czech part of the W2C web corpus [Majliš, 2011, Majliš and Žabokrtský, 2012], 16M tokens

Using the concatenated corpus, RoBERTa was trained from scratch using the official code released in the Fairseq library². Adam was used as an optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.98$, the batch size is 8,192 and the learning rate is adapted using the polynomial decay schema with 10,000 warm-up updates and the maximum learning rate set to $7 \cdot 10^{-4}$. The training took approximately three months.

They stated that for the NER problem, they reproduced the current state-of-the-art architecture [Straková et al., 2019] described in Section 2.2.1, but did not share code repository or trained weights.

2.2.5 XLM-RoBERTa

XLM-RoBERTa is a cross-lingual language model (XLM) [Lample and Conneau, 2019] based on RoBERTa [Liu et al., 2019], which is, as said in Section 2.2.4, a BERT-based model with an optimized training procedure³. XLM-RoBERTa was trained in one hundred languages, using 2.5 terabytes of filtered CommonCrawl data. In [Conneau et al., 2020] paper, they discuss the *curse of multilinguality*: More languages lead to better cross-lingual performance in low-resource languages up to a point, after which the overall performance on monolingual and cross-lingual benchmarks degrades. This curse can be alleviated by increasing the model capacity, that is why the authors made two models, which are very large; XLM-RoBERTa_{BASE} with 270M parameters and XLM-RoBERTa_{LARGE} with 550M parameters, which is, for example, over 4 times the size of RobeCzech from Section 2.2.4.

XLM-RoBERTa_{LARGE} significantly outperforms other multilingual models and performs well compared to other Czech monolingual models, as shown in the papers [Straka et al., 2021, Sido et al., 2021] and for this reason, I will also experiment with this model.

2.3 Metrics

In this Section, I will explain all primary metrics used to evaluate the models.

- **Accuracy** is the base metric used for model evaluation, which describes the number of correct predictions over all predictions:

$$\frac{\text{Number of Correct Predictions}}{\text{Number of all Predictions}}$$

- **Precision** is a measure of how many of the positive predictions are correct (true positives):

$$\frac{\text{True positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall** is a measure of how many of the positive cases the classifier correctly predicted over all the positive cases in the data:

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F_1 score** is a measure combining precision and recall. It is generally described as the harmonic mean [Ferber, 1931] of the two:

$$2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

²<https://github.com/facebookresearch/fairseq/tree/main/examples/roberta>

³For example, RoBERTa uses only MLM.

The F_1 score is a special case of the F_β score, where β is a positive real factor, chosen so that recall is considered β times as important as precision is:

$$(1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

Chapter 3

Dataset

This chapter describes the dataset used for fine-tuning and how the dataset is divided into different versions, introduces a problem related closely to NER, and how I represent the problem for the pre-trained models.

3.1 Overview

For the Czech NER, the largest public dataset is the **Czech Named Entity Corpus** [Ševčíková et al., 2007]. This dataset contains nested entities, which introduces the problem of nested NER. Although the main problem of this thesis is the solution to the NER problem, I decided to fine-tune models on this dataset and create models that solve the nested NER. Nested NER is described in Section 3.1.3. All versions are available for download from their official website [Ševčíková et al., 2007b].

The CNEC dataset is divided into different versions:

3.1.1 CNEC 1.0

This is the original version of the corpus. This version is outdated, as stated on the official website of the corpus version [Ševčíková et al., 2007a], and it is highly recommended to use the version described in Section 3.1.2, which contains the same data but offers fixes.

3.1.2 CNEC 1.1

The CNEC 1.1 is a corpus of 5868 Czech sentences with 33662 Czech named entities manually annotated, classified according to a two-level hierarchy (explained in Section 3.1.3) of 62 two-letter named entities and 10 one-letter supertypes shown in Figure 3.1. This is a minor update to its previous version.

The difference between CNEC 1.0 and CNEC 1.1 are the following:

- fixed some misannotated entities and typos
- make all formats contain the same data
- replaced tmt format by treex format
- split all formats into train, dtest, and etest

All data were created and annotated as stated in [Ševčíková et al., 2007] by this procedure:

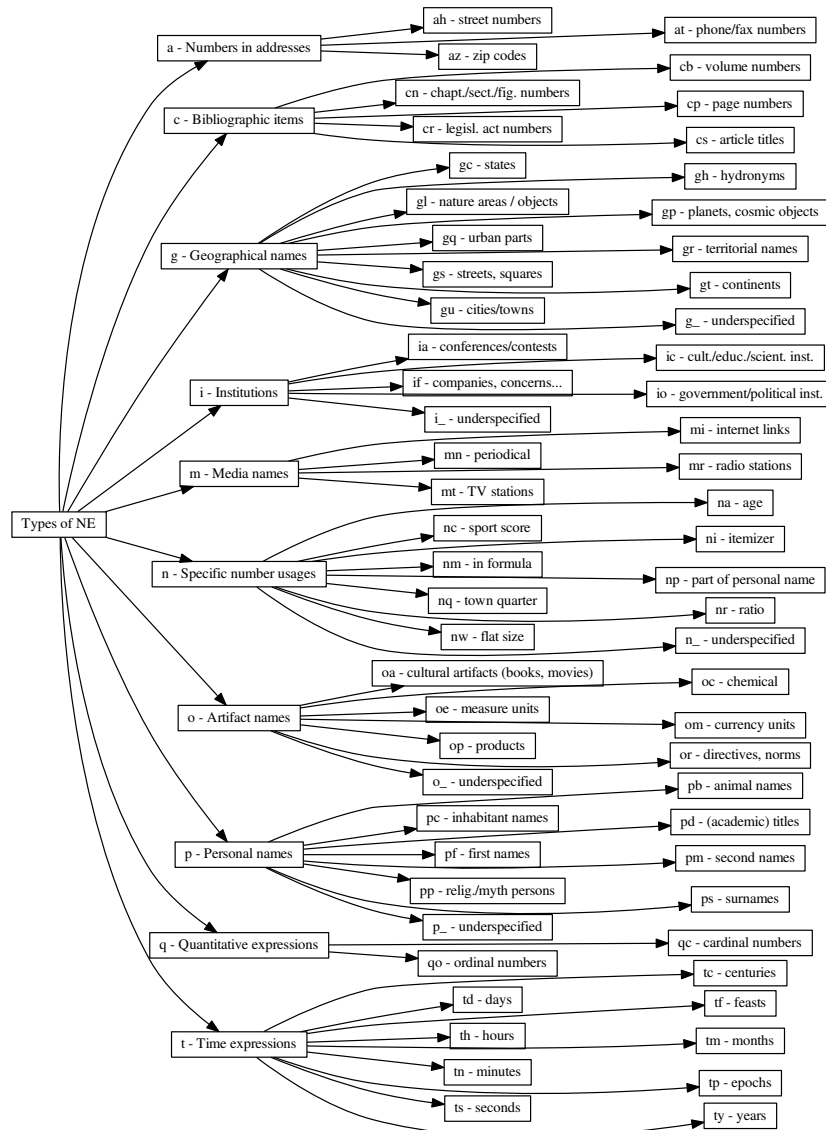


Figure 3.1: CNEC 1.1 named entities type hierarchy (reprinted from [Ševčíková et al., 2007])

- Sentences were randomly selected from the Czech National Corpus¹.
- The data have been manually annotated by two annotators in parallel and differently annotated instances have been checked and decided by a third person².
- The data have been divided into training, development test, and evaluation test parts in the 8:1:1 ratio.

■ 3.1.3 CNEC 2.0

This is the most recent version of the corpus. It contains 8993 Czech sentences with 35220 Czech named entities manually annotated. It is a major expansion of the previous version.

¹<https://ucnk.ff.cuni.cz/cs/>

²Magda Ševčíková, Zdeněk Žabokrtský, and Oldřich Krůza

The corpus uses 46 two-letter named entity types, which can be embedded creating the nested NER problem; e.g., the river name can be a part of the name of a city as in:

<gu Ústí nad <gh Labem>>

There are also 4 named entity containers: two or more named entities are parts of a named entity container, such as in:

<P <pf Jan><ps Novák>>

Containers are marked with a capital one-letter tag. All types of named entity can be seen in Figure 3.2, there can also be seen one-letter tags, which are not capital, these are called supertypes and are much more general, as opposed to the fine-grained two-letter types.

The changes from the previous version are stated on the official website of the version [Ševčíková et al., 2014b]:

- overhaul the number entities, some entities were merged, moved, or removed
- a new entity *me* for email was added
- other entities that showed similarities or were too fine-grained were merged
- new data was annotated and added, mainly for the corpus to better represent the density of named entities, so a lot of sentences with no named entities were added

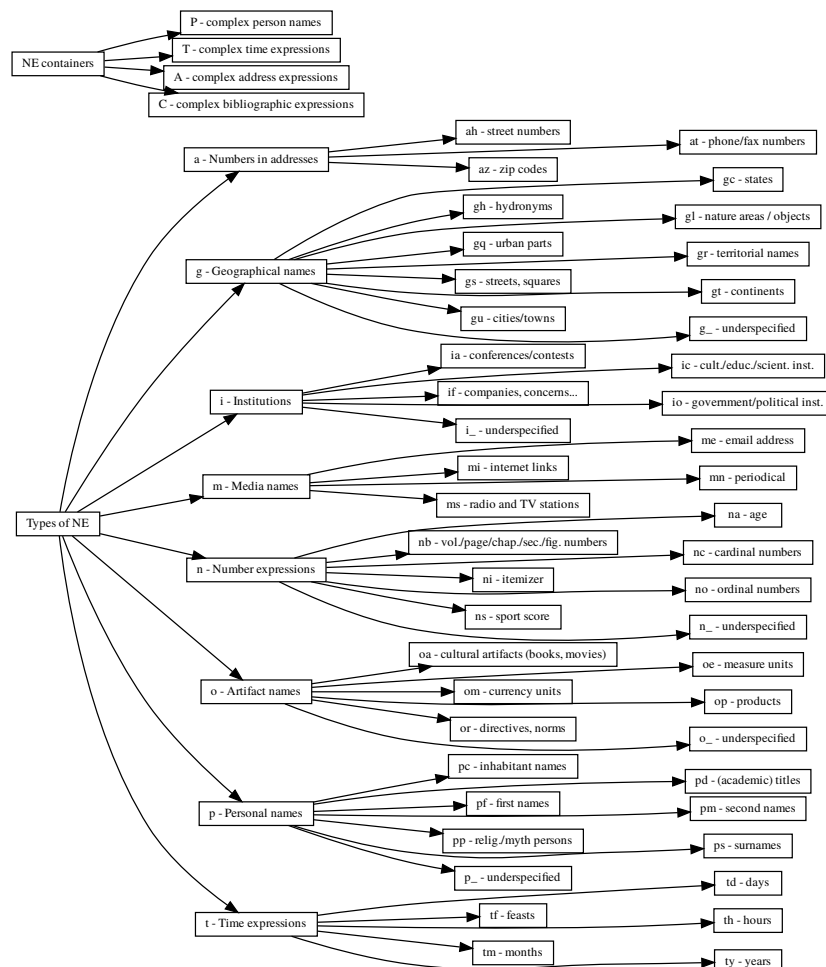


Figure 3.2: CNEC 2.0 named entities type hierarchy (reprinted from [Ševčíková et al., 2014a])

3.1.4 CoNLL-based extended

In the [Konkol and Konopík, 2013] paper, the authors introduced the CoNLL-based version of CNEC. Konkol and Konopík have taken both versions, removed the nested entities, and labeled all entities to only 7 supertypes³. This has transformed the problem of a nested NER into the problem of a flat NER.

3.2 Representation

The dataset, as explained in Section 3.1.3, can be embedded. This leads to the problem of a nested NER. However, we can also transform the data into that of a NER problem, as shown, for example, in Section 3.1.4. Both problems must be represented for the model to understand the dataset and train itself on it. As the main challenge of this thesis is the NER problem, I decided to convert even the version from Sections 3.1.2 and 3.1.3 from nested NER to flat NER and create a modified dataset containing the same data to achieve even stronger and diverse classifiers, which could help in future work.

3.2.1 Conversion

First, we need to look at how the dataset is represented and how we can work with it. The dataset is saved in Treex format. We would like to transform this format into a more readable ConLL format. To achieve this, I modified a Perl script from [Straková et al., 2016] paper.

A Treex format file is an XML file with specific rules, making it possible to use a script and convert this file to other formats. Treex format can be seen in the following example:

```

1 ...
2   <bundles>
3     <LM id="s1">
4       ... other tags and rules
5       <LM id="a_tree...">
6         <form>Asii</form>
7         <lemma>Asie_;G</lemma>
8         <tag>NNFS6-----A-----</tag>
9         <ord>19</ord>
10      </LM>
11      ...
12    <n_tree id="...">
13      ...
14      <LM id="...">
15        <ne_type>gt</ne_type>
16        <normalized_name>Asii</normalized_name>
17        <a.rf>a_tree...</a.rf>
18      </LM>
19      ...
20    </LM>
21 ...

```

Listing 3.1: An example of the data represented in Treex format

³Which are the supertypes shown in Figure 3.3

The script transforms the file into the CoNLL format, which is more human-readable. The example is shown below:

```

1 ...
2 Jihovýchodní jihovýchodní AAFS6----1A---- 0
3 Asii Asie_;G NNFS6-----A---- B-gt
4 a a-1 J^----- 0
5 you ten PDFS7----- 0
6 ...

```

Listing 3.2: An example of the data in CoNLL format for flat entities

The first column contains words, including punctuation, the second column contains lemmas of these words, the third column contains morphological tags, and the last fourth column contains NEs.

For the nested entities, it has the same structure with an added entity:

```

1 ...
2 jmenuje jmenovat_:T_:W VB-S---3P-AA--- 0
3 see se_^(zvr._zájmeno|'částice') P7-X4----- 0
4 Ann Ann_;Y NNFSX-----A---- B-P|B-pf
5 Suba Suba X@----- I-P|B-ps
6 . . Z:----- 0
7 ...

```

Listing 3.3: An example of the data in CoNLL format for nested entities

We can observe the usage of the Inside-outside-beginning (IOB) tag. In this instance, the usage of the IOB2 tagging is shown. This format of tagging was presented in the [Ramshaw and Marcus, 1995] paper and is commonly used for tagging tokens. The format can be described as:

- I- prefix indicates that the token is inside an entity
- B- prefix indicates that the token is the beginning of an entity
- O indicates that this token is not part of any entity

The difference between IOB and IOB2 is, in IOB, the **B-** prefix is used only after another token that is not tagged as **O**

■ 3.2.2 Labeling

Only having the file transformed into the CoNLL format will not be enough for the model to understand the input. Hugging Face models must have each entity labeled according to its corresponding ID number. Example given: An entity labeled ‘**O**’ will have an ID of **0**. Thanks to the Hugging Face library, I was able to create a simple loading script, following a template from Hugging Face⁴, in which I can prepare all the features that the dataset should have; I will explain this script more in detail in Section 4.1.2. The loading script loads the dataset into DatasetDict; this DatasetDict consists of three datasets, each representing the train, validation, and test split.

Each split has features, which are 'tokens' and 'ner_tags'. 'Tokens' are words, from each sentence, and 'ner_tags' are tags, for each token in the ID form. Simple example:

⁴https://huggingface.co/docs/datasets/dataset_script

```

1 {'tokens': ['Vede', 'ji', 'žena', ',', 'jmenuje', 'se', 'Ann', 'Suba', '.
   ↪ '],
2 'ner_tags': [0, 0, 0, 0, 0, 0, 75, 81, 0]}
3 '''
4 75: 'B-pf' (person's first name)
5 81: 'B-ps' (person's surname)
6 '''

```

Listing 3.4: An example of the labeling for flat entities

This example is for the flat NER problem, for nested NER, 'ner_tags' look slightly different, as the token 'Ann', has two labels.

```

1 {'tokens': ['Vede', 'ji', 'žena', ',', 'jmenuje', 'se', 'Ann', 'Suba', '.
   ↪ '],
2 'ner_tags': [[0], [0], [0], [0], [0], [0], [1, 83], [2, 89], [0]]}
3 '''
4 1: 'B-P',
5 2: 'I-P',
6 83: 'B-pf',
7 89: 'B-ps'
8 '''

```

Listing 3.5: An example of the labeling for nested entities

We can see that for the nested NER, I have to put all labels in an array because features expect all variables to be the same type.

The number of entities represented in the code for the model differs depending on the version used for fine-tuning and the type of problem we are trying to solve. It must be now noted that the number of 'ner_tags' used for fine-tuning and the number of named entities for the same version, as stated in Section 3.1.2 differs. After a short correspondence with Dr. Straková⁵, I was informed that the evaluation results from the [Ševčíková et al., 2007] paper onward were achieved on only 42 named entities, not 62 (shown in Figure 3.3), and due to backward compatibility, all results for this version were reported on these entities. It should now be explained what number of entities I used for each version:

- For the version from the Section 3.1.4, [Konkol and Konopík, 2013] used 7 supertypes, from the original supertypes, they excluded: 'c', 'n', and 'q'.
- For the version from Section 3.1.2, which solved the NER problem, I fine-tuned the model on 42 types, which can be seen in Figure 3.3. Furthermore, I fine-tuned and evaluated the model only on supertypes, which now used all supertypes, even from the excluded entities, as excluding supertypes now would make the corpus into the version from Section 3.1.4. When solving the nested NER problem, 4 containers are added, as shown in Figure 3.2, making the total number of entities 46.
- For the version from Section 3.1.3, the number of types and supertypes matches the one used for fine-tuning, which is 46 types and 8 supertypes. Again, for nested NER, 4 containers are added, making the total 50.

⁵Straková, personal communication, March 12, 2024

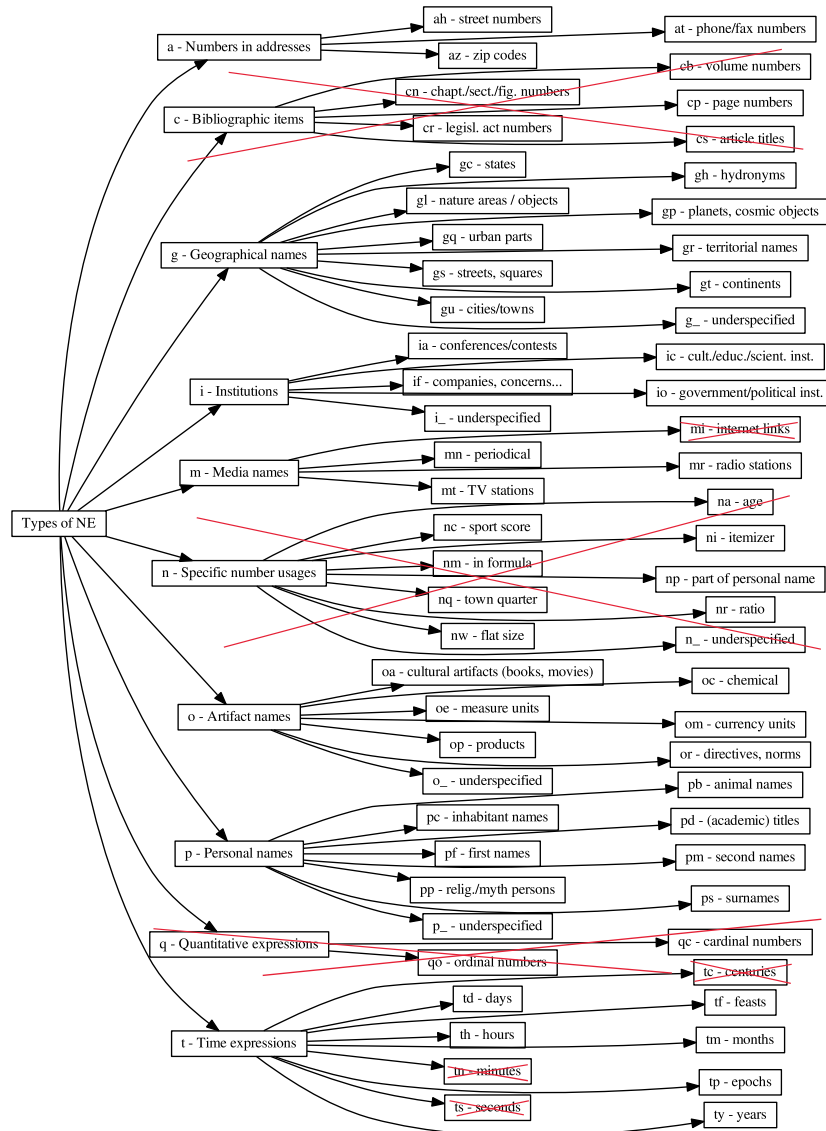


Figure 3.3: CNEC 1.1 named entities type hierarchy excluding unused entities (reprinted and edited from [Ševčíková et al., 2007])

Chapter 4

Model training

This chapter contains all the details on the training and evaluation of different models used for fine-tuning on the CNEC dataset described in Chapter 3. In this chapter, I also describe the data preprocessing required before every training. Training methods and data preprocessing may vary depending on whether the solution is for NER or nested NER. So, I will explain both solutions that I used separately. All notebooks, scripts, and data are in the enclosed repository explained in Appendix B.

4.1 Data Preprocessing

Just having a dataset is not enough to fine-tune the selected model on it. Different models have different tokenizers that expect the data to be in a suitable form to be passed to the model. In Section 3.2 I touched on this topic, but in this Section, I will explain all the steps more thoroughly.

4.1.1 Conversion

In Section 3.2.1 I showed how the raw data is represented in Listing 3.1. I also mentioned Perl scripts from [Straková et al., 2016] paper, which I modified to better suit my solution. There are in total four Perl scripts for conversion and one bash script. Each script converts the raw data into the resulting CoNLL formatted file and different versions depending on the arguments received for the bash script.

- `treex2conll2003.pl` converts the file with all types and flat entities, the result can be seen in Listing 3.2.
- `treex2conllsuper.pl` converts the file, but only with supertypes and flat entities.
- `treex2conll2003_nested.pl` converts the file again with all types, but for nested entities, the result can be seen in Listing 3.3.
- `treex2conllsuper_nested.pl` converts the file to supertypes and nested entities.

Each script has a hashmap for all the labels for both versions, and the hashmap used depends on the arguments passed into the script, so each script can handle both version CNEC 1.1 and 2.0. All versions were organized in their files for easier access and separated as raw data into train, test, and validation splits.

4.1.2 Loading Scripts

The data must be tagged and split into tokens to take advantage of the benefits the Hugging Face libraries offer. I mentioned this in Section 3.2.2, where the tagged data can be seen in Listing 3.4 for flat entities and Listing 3.5 for nested entities. To achieve this, I created loading scripts following the template¹ from Hugging Face and created the logic for the generation of samples, where each sample has features that were already mentioned 'ner_tags' and 'tokens'.

Thanks to the CoNLL format of the data, I can split each row into four columns, the first column is used as a token, the next two columns are ignored, as they are not needed for solving the NER problem, and the last column, which is the labeled entity, is used to tag the token. This tag has to be converted from the label to the assigned ID number. Keeping in mind that the data use IOB tagging, the number of labels is doubled, since we now have a label for the beginning of an entity and inside of an entity and one more for the nonentity type.

There is a small difference in the loading scripts for nested and flat entities. For flat entities, all 'ner_tags' are in a single array, but for nested entities, I need to split the last column using '|' and all the tags belonging to one token add to one array, so all 'ner_tags' are an array. This behavior is needed, as the Hugging Face library for datasets requires a feature to be of one predetermined type; here, the feature 'ner_tag' is a sequence of sequences that hold integers.

The whole purpose of the loading script is to load the dataset into the Hugging Face DatasetDict, where each split is represented as the Hugging Face Dataset type and can be used and easily accessed by other methods from Hugging Face libraries. An example of the loaded dataset can be seen below:

```

1 DatasetDict({
2   train: Dataset({
3     features: ['tokens', 'ner_tags'],
4     num_rows: 4695
5   })
6   validation: Dataset({
7     features: ['tokens', 'ner_tags'],
8     num_rows: 587
9   })
10  test: Dataset({
11    features: ['tokens', 'ner_tags'],
12    num_rows: 586
13  })
14 })

```

Listing 4.1: An example of the Hugging Face DatasetDict. The example is CNEC 1.1.

I pushed all datasets created from these scripts to Hugging Face Hub² to be accessible, with credit to the original authors, in the standard form most Hugging Face datasets for NER are and for easy loading with the Hugging Face `load_dataset("stulcrad/dsName")` function.

¹https://huggingface.co/docs/datasets/dataset_script

²<https://huggingface.co/stulcrad>

4.1.3 Tokenize and Align

For the models to make sense of the input, the tokens must be converted to token IDs. All models have their tokenizer API, which does this job so that the model can recognize each token ID based on a predefined vocabulary. We have the tokens already pre-tokenized and let the tokenizer further tokenize the input, as shown in the example below:

```

1  ['I', 's', 'Dubenkou', ',', 'na', 'kterou', 'U', 'tygra', 'ted', '
   ↪ myslím', '.', '.', '.']
2  ['[CLS]', 'I', 's', 'Dub', '##enko', '##u', ',', 'na', 'kterou', 'U',
   ↪ 'tyg', '##ra', 'ted', 'myslím', '.', '.', '.', '[SEP]']
3  [2, 45, 87, 13829, 22930, 1025, 16, 1939, 2897, 57, 28258, 1931,
   ↪ 3460, 5747, 18, 18, 18, 3]

```

Listing 4.2: An example of pre-tokenized inputs, tokenized inputs and their ids model accepts

As we can see, the tokenizer adds special tokens used by models at the beginning [CLS] and at the end [SEP], and words that were not tokenized to the desired form are further tokenized into subwords as we can see in word Dubenkou for example. This introduces a mismatch between our input and the labels (`ner_tags`), where the list of labels has 13 elements, whereas our new input has 18 tokens. We now need to align the labels correctly with the tokenized input. The tokenizers can easily map each token to the original word with `word_ids`, and it can be seen how they work in Table 4.2. The way in which I align the labels differs here, depending on whether we align to flat NER or nested NER.

Flat NER

For flat NER, the main challenge of this thesis, this process is more straightforward, as each token will have exactly one label. The first rule I applied is to label special tokens with -100. That is because, by default, -100 is an index ignored by cross-entropy loss, which the training uses. Then, each token gets the same label as the token that started the word since they are part of the same entity, I just changed B- to I- for these tokens. This process shows how the pre-tokenized input in Table 4.1 is converted into the tokenized and aligned input in Table 4.2.

Tokens	I	s	Dubenkou
NER tags	O	O	B-P

Table 4.1: Pre-tokenized labeled input

Tokens	[CLS]	I	s	Dub	##enko	##u	[SEP]
Word ids	None	0	1	2	2	2	3
NER tags	None	O	O	B-P	I-P	I-P	None
Labels	-100	0	0	15	16	16	-100

Table 4.2: Tokenized and aligned input

■ Nested NER

For nested NER, aligning labels with tokens is a bit more complicated. The problem is that one token can have more than one label. My first thought was to follow the solution for flat NER, but again, put all labels into arrays so that they would be of one type. However, this created inconsistencies, as the Hugging Face trainer API required labels to be a sequence of integers, which was now an impossible task. This required a different approach to solving the nested NER, where I will explain the training loop more in Section 4.2.2. The cross-entropy loss was replaced with `BCEWithLogitsLoss`³ and to fit this loss, I need to one-hot encode my labels.

One-hot encoding is a technique used to represent categorical variables as numerical values. In our case, each token is treated separately, and the labels are recognized with 0 or 1. A very simple example of one token “Labe”, with only three tags (person, geographical location, and institution), can be seen in Table 4.3, where it is a river, but in an entity “Hostinec u Labe” it is a part of an institution.

O	B-P	I-P	B-G	I-G	B-I	I-I
0	0	0	1	0	0	1

Table 4.3: An example of one-hot encoded label for one token

Special tokens will have at all indexes -100, so they can be ignored.

■ 4.2 Training Loop

After our data is properly prepared, the next step is the actual training of the models on the data. Here, the methods used to solve each problem again differ and will be explained in their respective sections.

■ 4.2.1 Flat NER

The Hugging Face community created a `Trainer`⁴ class that goes hand in hand with `TrainingArguments`⁵ class and together they provide a complete training API, which I use to train the models for flat NER.

The `Trainer` class is an already prepared training loop, which is optimized to work correctly and efficiently with Hugging Face Transformers models and its use is justified in Section 1.2. Using the `TrainingArguments` class, I can further tweak different metaparameters, such as batch size, learning rate, weight decay, number of epochs, etc. For each model, I decided to find the best metaparameters using the trial-and-error method that maximizes the F_1 score. I mostly played with batch sizes, trying the sizes 64, 32, 16, 8, 4, and 2. The starting learning rate remained the same for all models as $2 \cdot 10^{-5}$; for weight decay, I chose between 0.01 and 0.001, and the number of epochs varied between 8 and 20 epochs. The reasoning behind this is that even though one model can perform well with certain metaparameters, the other model can perform better with different arguments. `Trainer` uses an AdamW [Loshchilov and Hutter, 2019] optimizer, which is an

³<https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html>

⁴https://huggingface.co/docs/transformers/main_classes/trainer

⁵https://huggingface.co/docs/transformers/v4.40.2/en/main_classes/trainer#transformers.TrainingArguments

Adam optimizer with a weight decay fix with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and the dropout rate for the model again varies depending on the model and the version of the dataset. I choose dropout rates from values between 0.3 and the default 0.1.

It should be noted that for the `Trainer` to accept data in a batch, I must first form the batches with a `DataCollator`⁶. This class pads the inputs including the labels to be of the same length, which can be seen in the Listing 4.3.

```

1 [-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, 0, -100]
2 [-100, 1, 2, -100]
3 tensor([
4 [-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, 0, -100],
5 [-100, 1, 2, -100, -100, -100, -100, -100, -100, -100, -100, -100]])

```

Listing 4.3: An example of padding, first two arrays are unpadded labels and the tensor is the padded labels

To observe how the model performs, metrics are computed after an epoch or a given number of steps. For this purpose, `Trainer` accepts a function that computes the desired metrics and outputs them to us to see. The metrics are evaluated with `sequeval` [Nakayama, 2018], which is a framework for sequence labeling evaluation. The framework needs predictions and references, which must be converted back into the string form, and I added optional arguments to force the framework to compute the metrics with a strict mode, which will ensure that one entity is correctly predicted if all the tokens have the correct reference labels. An example for better clarification is shown in Listing 4.4, where it can be seen that the model must correctly predict not only labels but also the span of the entity.

```

1 >>> from sequeval.metrics import classification_report
2 >>> from sequeval.scheme import IOB2
3 >>> y_true = [['B-NP', 'I-NP', 'O']]
4 >>> y_pred = [['I-NP', 'I-NP', 'O']]
5 >>> classification_report(y_true, y_pred)
6           precision    recall  f1-score   support
7      NP           1.00      1.00      1.00         1
8  micro avg           1.00      1.00      1.00         1
9  macro avg           1.00      1.00      1.00         1
10 weighted avg          1.00      1.00      1.00         1
11 >>> classification_report(y_true, y_pred, mode='strict', scheme=IOB2)
12           precision    recall  f1-score   support
13      NP           0.00      0.00      0.00         1
14  micro avg           0.00      0.00      0.00         1
15  macro avg           0.00      0.00      0.00         1
16 weighted avg          0.00      0.00      0.00         1

```

Listing 4.4: A difference between using `sequeval` with strict mode or with default mode

The use of strict mode is required as all benchmarks report the scores based on correct entity prediction, not token prediction. The framework, by default, ignores non-entity labels.

With this, we can start the training with a simple function call `Trainer.train()` and I uploaded the best performing models that can be seen in Section 4.3 in Table 4.4 to the Hugging Face Hub for easy access and simple showcase of the application.

⁶https://huggingface.co/docs/transformers/main_classes/data_collator#datacollatorfortokenclassification

4.2.2 Nested NER

Encouraged by the state-of-the-art results of my flat NER solutions shown in Section 4.3 I have further experimented with nested NER. The training for nested NER posed more of a challenge because this problem is much less researched than flat NER. From the beginning, I thought of reusing the same training method as for flat NER, but I was quickly stopped by the Hugging Face trainer API requiring the labels to be a sequence of integers. After doing some research⁷, I found out that nested NER requires the use of a different loss function, namely the `BCEWithLogitsLoss`.

Using this loss, I created a custom training loop with the help of different guides to fine-tuning⁸ and other similar solutions⁹. This training loop uses the AdamW optimizer again with a starting learning rate of $2 \cdot 10^{-5}$, the number of training epochs differs between 10 and 40 epochs, depending on the models to maximize the scores. Unfortunately, the batch size is set to 1, as I could not make the training loop work with a larger batch size. The weight decay is fixed to 0.01 and the dropout rate of the model is chosen from values of 0.25 and the default 0.1.

For model performance observations, the metrics were now calculated with a scikit-learn classification report function¹⁰ instead of `seqeval`, due to the use of one-hot encoded labels, which `seqeval` would not be able to recognize, of course again on the validation split of the dataset. However, this evaluation calculates the metrics with non-entity types and with token prediction. For that reason the reported metrics are only indicative, and the final results shown in Table 4.5 are evaluated using the same script as my benchmark NameTag2 was evaluated from paper [Straková et al., 2019]¹¹ to make sure the scores can be compared correctly on the entity prediction level.

The training of the models for the nested NER took more time than for flat NER and especially the training of XLM-RoBERTa_{LARGE} lasted the longest, mainly because the model had often faced problems with overfitting and learning to predict only empty labels for each token, this forced me to restart the training every time such thing happened as the model would get stuck and could not learn anymore. Interestingly, no other model suffered from this issue, the reason behind that can be discussed, maybe XLM-RoBERTa_{LARGE} is too large, or maybe because the model is less focused on Czech than other used models, which can cause some irregularities.

Activation function

I mentioned that `seqeval` would not be able to recognize one-hot encoded labels and I had to use a different classification function. It should now be noted that I had to use even a different activation function. It is known that the output of a trained model for any classification task must be used in an activation function. The most widely used function today is the Softmax function¹², which converts a vector of numbers into a probability distribution. For nested NER this function would not work, since now we could have more correct predictions, because of that I suggest the use of Heaviside step function [Baowan et al., 2017]. It is a function whose value is zero for negative arguments and one for positive

⁷<https://discuss.huggingface.co/t/multi-label-token-classification/16509/5>

⁸<https://huggingface.co/docs/transformers/training>

⁹https://github.com/lstickert/seq2seq-slr/blob/main/model_trainers/classifier/model.py

¹⁰https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

¹¹https://github.com/ufal/ac12019_nested_ner

¹²<https://pytorch.org/docs/stable/generated/torch.nn.Softmax.html>

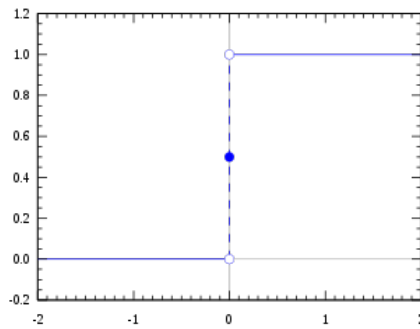


Figure 4.1: A Heaviside step function (reprinted from [Baowan et al., 2017])

arguments, as illustrated in Figure 4.1. There is a special case where an argument is zero and the value is 0.5, but this case can also be modified to equal zero.

4.3 Evaluation

After the successful training, all models must be evaluated and compared on the test split of the CNEC dataset. The benchmark used for both the flat and nested NER problem will be NameTag 2 from Section 2.2.1, but for flat NER the scores are not reported apart from the version described in Section 3.1.4 and for other models, I collected all results I could find from their attached papers.

As an evaluation method, I used entity-level evaluation. This method is commonly used on all different models that report their F scores. The model will predict labels and the span of an entity, the prediction is then correct only if both the span and the labels are correct. For flat entities, I showed the evaluation method in Listing 4.4. For nested entities, I used the same scripts for evaluation from the NameTag 2 paper [Straková et al., 2019] GitHub repository¹³. For the input word *Michaela Talirova*, the golden data, which the script requires, would have the form shown in Listing 4.5 and the prediction should look the same.

```
1 178,179 P Michaela Talirova
2 178 pf Michaela
3 179 ps Talirova
```

Listing 4.5: An example of golden data for nested entities

However, the model does not output the predictions in this form, but after the activation function in the one-hot encoded labels, which I explained in Section 4.1.3. The conversion from one-hot encoding to an array of labels is a trivial task, I just iterate through one label, and for each 1 I add into an array the index in which the 1 stands. After this conversion, I created a simple Python script that takes predictions and true labels and saves them in a CoNLL format in a new file. The new file is then converted using a script from the NameTag2 paper, and as a last step, a different script compares the gold data and the prediction and reports the final F score.

After ensuring all the evaluation methods are the same, I can start reporting all F scores. The F score is an F_1 score explained in Section 2.3 and is a metric that is most used to explore the performance of the NER models. In Table 4.4 are the F scores of the models

¹³https://github.com/ufal/ac12019_nested_ner

Model	CNEC 1.1 CoNLL	CNEC 2.0 CoNLL	CNEC 1.1 Super- types	CNEC 2.0 Super- types	CNEC 1.1 Types	CNEC 2.0 Types
NameTag2	83.27*	86.39*	-	-	-	-
Czert-B	86.27*	-	-	-	-	-
SlavicBert	86.57*	-	-	-	-	-
RobeCzech	87.47*	-	-	-	-	-
XLM-RoBERTa	86.86*	-	-	-	-	-
Czert-B	83.95	83.72	82.39	82.78	82.84	82.22
SlavicBert	86.95	86.44	85.31	85.45	84.53	83.32
RobeCzech	87.78	88.19	86.20	86.85	86.81	85.17
XLM-RoBERTa	87.41	88.10	86.64	85.91	86.20	85.26

Table 4.4: F score comparison of models trained to solve the NER problem. All scores are reported on the test set. The scores indexed with * are reprinted from their papers and websites. NameTag2 is from the official CNEC website [Ševčíková et al., 2007b], Czert-B and SlavicBert are from [Sido et al., 2021] and RobeCzech with XLM-RoBERTa are from [Straka et al., 2021]. The best scores are highlighted with bold text.

Model	CNEC 1.1 Supertypes	CNEC 2.0 Supertypes	CNEC 1.1 Types	CNEC 2.0 Types
NameTag2	89.91*	88.02*	86.88*	84.66*
Czert-B	81.83	83.10	80.28	78.96
SlavicBert	82.21	82.74	81.21	79.49
RobeCzech	83.62	83.86	82.52	81.70
XLM-RoBERTa	82.59	83.69	81.12	80.92

Table 4.5: F score comparison of trained models to solve the nested NER problem. All scores are reported on the test set. Scores for NameTag are reprinted from the official CNEC website [Ševčíková et al., 2007b] and are indexed with *. The best scores are highlighted with bold text aside from NameTag.

trained on flat entities and in Table 4.5 are the F scores for nested entities. For clarification, when I am writing XLM-RoBERTa, I am actually using the XLM-RoBERTa_{LARGE} version, and all other models are used in their “BASE” size. The best scores are highlighted with bold text and the scores indexed with * are reprinted, not my results.

From Table 4.4 I can see that RobeCzech and XLM-RoBERTa are very close to each other in terms of performance and both models surpass the current state-of-the-art results for the version of the dataset from Section 3.1.4 and for the unreported version, the models achieve very good results. For empirical analysis, I will use and compare both models and discuss if there are differences in classification, and I will recommend one model for use afterwards.

The scope of this thesis was the training and creation of a model that solves the NER problem and the later application of this solution would be used in the fact-checking platform. For this reason and after consultation with the supervisor, I focused on the strong and reliable solution of NER, and in this challenge, I have achieved state-of-the-art using a model of reasonable size using only trainable parameters and the efficient and effective Hugging Face trainer API. Furthermore, for a deeper understanding of the NER problem, I decided to train a model that solves the nested NER. From Table 4.5 I can see

that my scores do not reach the state-of-the-art solutions, but are coming close to it. In the NameTag 2 paper [Straková et al., 2019], the proposed solution is a complex and well-made training combining state-of-the-art methods and creating a very strong classifier for nested entities. However, I would like to emphasize that Transformers can learn to solve this problem with a simple training loop and, with more thorough research, can achieve strong results.

We can see that the RobeCzech model outperforms other models, and for this model, I wrote a very simple demo notebook in my GitLab repository to showcase the functionality.

Chapter 5

Empirical Analysis

In this chapter, I will perform an empirical analysis of general real-world data from a random news article. This chapter will contain screenshots of the RobeCzech and XLM-RoBERTa fine-tuned models on CNEC 2.0 with flat entities divided by the number of entities used and discussions of issues and positives. I decided to use only models fine-tuned on CNEC 2.0 as it contains more data and better represents the density of named entities in different sentences as stated on their website [Ševčíková et al., 2014b]. For nested NER I created a small example.

5.1 CoNLL-based extended

The screenshots for the CoNLL-based version from Section 3.1.4 can be seen in Figure 5.1 for RobeCzech and for XLM-RoBERTa in Figure 5.2. The predicted entities are the one-letter supertypes shown in Figure 3.3.

From the screenshots, we can see that both models predict correctly all entities. There are no visible differences in the classifications in this given example. This version of the dataset trains the models very well, but it lacks the number entities, which I find important because they appear often in real-world data, not just in dates. The solution to this would be adding this entity to the corpus, but that would transform this version of the dataset into the version of the dataset described in Section 3.1.3 with only supertypes.

5.2 CNEC 2.0 Supertypes

The screenshots for the CNEC 2.0 with only supertypes are shown in Figure 5.3 for RobeCzech and in Figure 5.4 for XLM-RoBERTa. The predicted entities are the one-letter supertypes shown in Figure 3.2.

Again, the predicted entities are almost identical for both models. Still, we can now notice that for the e-mail entity, XLM-RoBERTa has a problem with recognizing it completely, as it leaves out the dot in between the e-mail and the region recognizer. The reason for that may be that XLM-RoBERTa, being much larger than RobeCzech, is overfitting to memorizing that the dot should always be a non-entity type in longer texts. Maybe the enlargement of the learning dataset might solve this issue, or better choice of metaparameters could help, but in both cases, RobeCzech seems to perform better, both in the test split of the dataset, as shown in Table 4.4, and on real-world data as shown here.

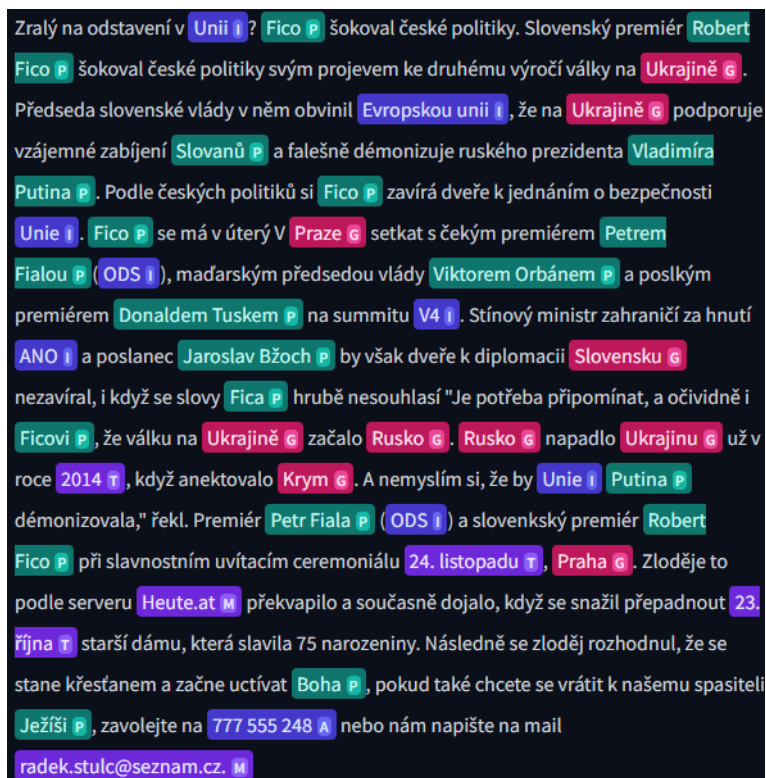


Figure 5.1: A screenshot of an example of RobeCzech fine-tuned on CNEC 2.0 CoNLL extended.

5.3 CNEC 2.0 Types

The screenshots for the CNEC 2.0 with all types are shown in Figure 5.5 for RobeCzech and in Figure 5.6 for XLM-RoBERTa. The predicted entities are the two-letter types shown in Figure 3.2.

For the examples, we can now notice a few changes. The RobeCzech did not classify the name Putin in the part of a sentence ""Unie Putina démonizovala"", and incorrectly classified the word serveru in the next sentence. XLM-RoBERTa again has a problem with the e-mail entity, but aside from that, it classified everything else correctly.

5.4 Nested NER

For nested entities, a small demo notebook was created, to display the functionality of the models on a small input. The notebook shows the classification of individual tokens for RobeCzech and XLM-RoBERTa with supertypes and all types. The classification of a sentence “Jmenuju se Radek Štulc a žiju v Praze. Narodil jsem se v Ústí nad Labem v roce 2001.” can be seen in Figure 5.7

5.5 Results of analysis

The models fine-tuned from Section 5.1 seem to work correctly on this example, but the absence of a number entity may become problematic, as a claim “Radkovi je 25 let” could

Zralý na odstavení v Unii I? Fico P šokoval české politiky. Slovenský premiér Robert Fico P šokoval české politiky svým projevem ke druhému výročí války na Ukrajině G. Předseda slovenské vlády v něm obvinil Evropskou unii I, že na Ukrajině G podporuje vzájemné zabíjení Slovanů P a falešně demonizuje ruského prezidenta Vladimíra Putina P. Podle českých politiků si Fico P zavírá dveře k jednáním o bezpečnosti Unie I. Fico P se má v úterý v Praze G setkat s českým premiérem Petrem Fialou P (ODS I), maďarským předsedou vlády Viktorom Orbánem P a poslkým premiérem Donaldem Tuskem P na summitu V4 I. Stínový ministr zahraničí za hnutí ANO I a poslanec Jaroslav Bžoch P by však dveře k diplomacii Slovensku G nezavíral, i když se slovy Fica P hrubě nesouhlasí "Je potřeba připomínat, a očividně i Ficovi P, že válku na Ukrajině G začalo Rusko G. Rusko G napadlo Ukrajinu G už v roce 2014, T když anektovalo Krym G. A nemyslím si, že by Unie I Putina P demonizovala," řekl. Premiér Petr Fiala P (ODS I) a slovenský premiér Robert Fico P při slavnostním uvítacím ceremoniálu 24. listopadu T, Praha G. Zloděje to podle serveru Heute.at M překvapilo a současně dojalo, když se snažil přepadnout 23. října T starší dámu, která slavila 75 narozeniny. Následně se zloděj rozhodnul, že se stane křesťanem a začne uctívat Boha P, pokud také chcete se vrátit k našemu spasiteli Ježíši P, zavolejte na 777 555 248 A nebo nám napište na mail nejsemscam@seznam.cz M.

Figure 5.2: A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 CoNLL extended

not be proven as factual or not, because the entity for age would not exist.

The models in Section 5.3 as expected from the F scores in Table 4.4 perform the worst on real-world data. It seems that fine-tuning the models on fine-grained types poses more of a challenge, as the model has a lot of labels to choose from and can get confused more easily; because of that I would recommend using a less fine-grained version.

Section 5.2 offers the best results with real-world data, the example is classified correctly, and the version contains all entities that we could encounter in everyday life. Between the two models, I would recommend using RobeCzech as it performed slightly better both in F scores as seen in Table 4.4 and adapts better to the NER problem.

Zralý na odstavení v Unii i? Fico P šokoval české politiky. Slovenský premiér Robert Fico P šokoval české politiky svým projevem ke druhému výročí války na Ukrajině G. Předseda slovenské vlády v něm obvinil Evropskou unii i, že na Ukrajině G podporuje vzájemné zabíjení Slovanů P a falešně demonizuje ruského prezidenta Vladimíra Putina P. Podle českých politiků si Fico P zavírá dveře k jednáním o bezpečnosti Unie i. Fico P se má v úterý V Praze G setkat s českým premiérem Petrem Fialou P (ODS i), maďarským předsedou vlády Viktorom Orbánem P a poslkým premiérem Donaldem Tuskem P na summitu V4 i. Stínový ministr zahraničí za hnutí ANO i a poslanec Jaroslav Bžoch P by však dveře k diplomacii Slovensku G nezavíral, i když se slovy Fica P hrubě nesouhlasí "Je potřeba připomínat, a očividně i Ficovi P, že válku na Ukrajině G začalo Rusko G. Rusko G napadlo Ukrajinu G už v roce 2014 T, když anektovalo Krym G. A nemyslím si, že by Unie i Putina P demonizovala," řekl. Premiér Petr Fiala P (ODS i) a slovenský premiér Robert Fico P při slavnostním uvítacím ceremoniálu 24. listopadu T, Praha G. Zloděje to podle serveru Heute.at M překvapilo a současně došlo, když se snažil přepadnout 23. října T starší dámu, která slavila 75 N narozeniny. Následně se zloděj rozhodnul, že se stane křesťanem a začne uctívat Boha P, pokud také chcete se vrátit k našemu spasiteli Ježíši P, zavolejte na 777 555 248 A nebo nám napište na mail nejsemscam@seznam.cz. M

Figure 5.3: A screenshot of an example of RobeCzech fine-tuned on CNEC 2.0 Supertypes

Zralý na odstavení v Unii i? Fico P šokoval české politiky. Slovenský premiér Robert Fico P šokoval české politiky svým projevem ke druhému výročí války na Ukrajině G. Předseda slovenské vlády v něm obvinil Evropskou unii i, že na Ukrajině G podporuje vzájemné zabíjení Slovanů P a falešně demonizuje ruského prezidenta Vladimíra Putina P. Podle českých politiků si Fico P zavírá dveře k jednáním o bezpečnosti Unie i. Fico P se má v úterý V Praze G setkat s českým premiérem Petrem Fialou P (ODS i), maďarským předsedou vlády Viktorom Orbánem P a poslkým premiérem Donaldem Tuskem P na summitu V4 i. Stínový ministr zahraničí za hnutí ANO i a poslanec Jaroslav Bžoch P by však dveře k diplomacii Slovensku G nezavíral, i když se slovy Fica P hrubě nesouhlasí "Je potřeba připomínat, a očividně i Ficovi P, že válku na Ukrajině G začalo Rusko G. Rusko G napadlo Ukrajinu G už v roce 2014, T když anektovalo Krym G. A nemyslím si, že by Unie i Putina P demonizovala," řekl. Premiér Petr Fiala P (ODS i) a slovenský premiér Robert Fico P při slavnostním uvítacím ceremoniálu 24. listopadu T, Praha G. Zloděje to podle serveru Heute.at M překvapilo a současně došlo, když se snažil přepadnout 23. října T starší dámu, která slavila 75 N narozeniny. Následně se zloděj rozhodnul, že se stane křesťanem a začne uctívat Boha P, pokud také chcete se vrátit k našemu spasiteli Ježíši P, zavolejte na 777 555 248 A nebo nám napište na mail nejsemscam@seznam M. cz M.

Figure 5.4: A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 only supertypes

Zralý na odstavení v Unii io ? Fico ps šokoval české politiky. Slovenský premiér Robert pf Fico ps šokoval české politiky svým projevem ke druhému výročí války na Ukrajině gc. Předseda slovenské vlády v něm obvinil Evropskou unii io, že na Ukrajině gc podporuje vzájemné zabíjení Slovanů pc a falešně demonizuje ruského prezidenta Vladimíra pf Putina ps. Podle českých politiků si Fico ps zavírá dveře k jednáním o bezpečnosti Unie io. Fico ps se má v úterý v Praze gu setkat s českým premiérem Petrem pf Fialou ps (ODS io), maďarským předsedou vlády Viktorem pf Orbánem ps a poslkým premiérem Donaldem pf Tuskem ps na summitu V4 io. Stínový ministr zahraničí za hnutí ANO io a poslanec Jaroslav pf Bžoch ps by však dveře k diplomacii Slovensku gc nezavíral, i když se slovy Fica ps hrubě nesouhlasí "Je potřeba připomínat, a očividně i Ficovi ps, že válku na Ukrajině gc začalo Rusko gc. Rusko gc napadlo Ukrajinu gc už v roce 2014 ty, když anektovalo Krym gr. A nemyslím si, že by Unie io Putina ps demonizovala," řekl. Premiér Petr pf Fiala ps (ODS io) a slovenský premiér Robert pf Fico ps při slavnostním uvítacím ceremoniálu 24. td listopadu tm, Praha gu. Zloděje to podle serveru mn Heute.at mn překvapilo a současně dojalo, když se snažil přepadnout 23. td října tm starší dámu, která slavila 75 na narozeniny. Následně se zloděj rozhodnul, že se stane křesťanem a začne uctívat Boha pp, pokud také chcete se vrátit k našemu spasiteli Ježíši pp, zavolejte na 777 555 248 at nebo nám napište na mail nejsemscam@seznam.cz. me


Figure 5.5: A screenshot of an example of RobeCzech fine-tuned on CNEC 2.0 only types

Zralý na odstavení v Unii io ? Fico ps šokoval české politiky. Slovenský premiér Robert pf Fico ps šokoval české politiky svým projevem ke druhému výročí války na Ukrajině gc. Předseda slovenské vlády v něm obvinil Evropskou unii io, že na Ukrajině gc podporuje vzájemné zabíjení Slovanů pc a falešně demonizuje ruského prezidenta Vladimíra pf Putina ps. Podle českých politiků si Fico ps zavírá dveře k jednáním o bezpečnosti Unie io. Fico ps se má v úterý v Praze gu setkat s českým premiérem Petrem pf Fialou ps (ODS io), maďarským předsedou vlády Viktorem pf Orbánem ps a poslkým premiérem Donaldem pf Tuskem ps na summitu V4 io. Stínový ministr zahraničí za hnutí ANO io a poslanec Jaroslav pf Bžoch ps by však dveře k diplomacii Slovensku gc nezavíral, i když se slovy Fica ps hrubě nesouhlasí "Je potřeba připomínat, a očividně i Ficovi ps, že válku na Ukrajině gc začalo Rusko gc. Rusko gc napadlo Ukrajinu gc už v roce 2014, ty když anektovalo Krym gr. A nemyslím si, že by Unie io Putina ps demonizovala," řekl. Premiér Petr pf Fiala ps (ODS io) a slovenský premiér Robert pf Fico ps při slavnostním uvítacím ceremoniálu 24. td listopadu tm, Praha gu. Zloděje to podle serveru Heute.at mn překvapilo a současně dojalo, když se snažil přepadnout 23. td října tm starší dámu, která slavila 75 na narozeniny. Následně se zloděj rozhodnul, že se stane křesťanem a začne uctívat Boha pp, pokud také chcete se vrátit k našemu spasiteli Ježíši pp, zavolejte na 777 555 248 at nebo nám napište na mail nejsemscam@seznam me. CZ me.

Figure 5.6: A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 only types

```
_Rad      ['B-P', 'B-pf']  
ek        ['I-P', 'I-pf']  
_š        ['I-P', 'B-ps']  
tul       ['I-P', 'I-ps']  
c         ['I-P', 'I-ps']  
_Praze   ['B-gu']  
_Ústí    ['B-gu']  
_nad     ['I-gu']  
_Labem   ['B-gh', 'I-gu']  
_2001.   ['B-ty']
```

Figure 5.7: A screenshot of an example of XLM-RoBERTa fine-tuned on CNEC 2.0 all nested types



Chapter 6

Conclusion

In this Bachelor's thesis, I explored the state-of-the-art methods used for Named Entity Recognition (NER) in English and Czech, together with a description of all models used for fine-tuning in this thesis and the metrics used for performance observation (Chapter 2).

I explained the dataset, used for fine-tuning, named the Czech Named Entity Corpus (CNEC), how the dataset is divided into different versions CNEC 1.0, which is an outdated version, CNEC 1.1, which includes corrections to 1.0, and CNEC 2.0, which is the most recent version of the corpus, contains the most data, and new entities (Chapter 3). I also explained how the dataset is represented for the model, converting from Treex format to human-readable CoNLL format using scripts from [Straková et al., 2016], and labeling the inputs.

In Chapter 4, I followed this with an explanation of the training and evaluation of the different models used for fine-tuning on the CNEC dataset. This chapter also describes the essential data preprocessing steps before training and the variations in preprocessing and training depending on whether the solution is for flat NER or nested NER. For flat NER training, I used the Hugging Face training API for its efficiency and effectiveness. The metaparameters, such as batch size, number of epochs, dropout rate, and weight decay, varied depending on the model to achieve the best performance, but the starting learning rate for the AdamW optimizer remained constant at $2 \cdot 10^{-5}$. Training for nested NER required one-hot encoding of all labels and a change in the loss function for the training loop. Again, the metaparameters varied while the learning rate remained at $2 \cdot 10^{-5}$. Lastly, using entity-level evaluation, I evaluated all models on all versions of CNEC for both flat and nested NER, where the entity must have all tokens correctly labeled, not just its subwords. I ensured that all solutions were correctly evaluated and then explored the reported F scores. From the scores, it can be seen that XLM-RoBERTa and RobeCzech are the two best-performing models for flat NER, surpassing the current state-of-the-art, with RobeCzech also outperforming other models for nested NER.

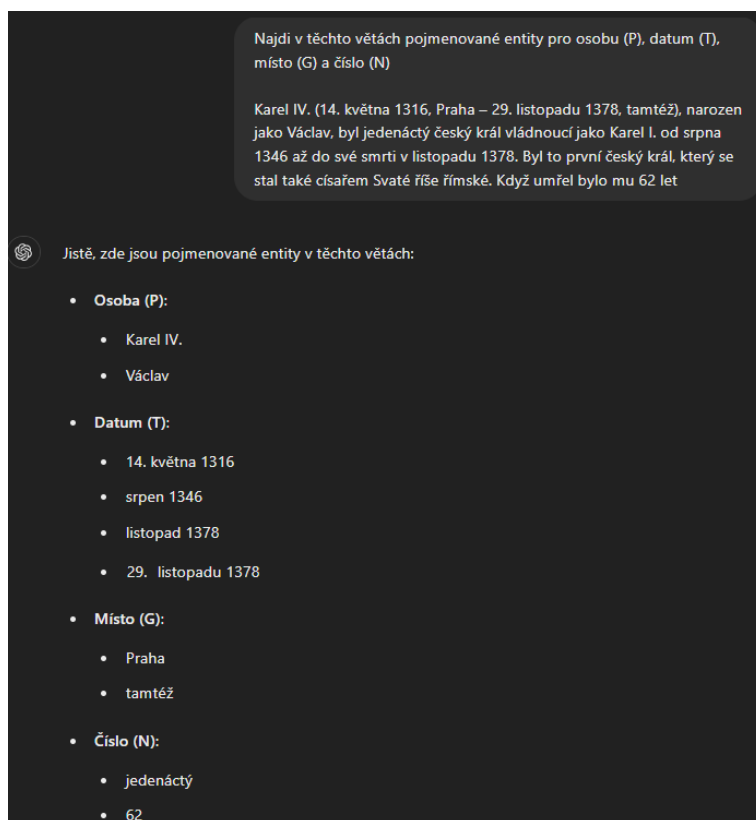
Lastly, I conducted an empirical analysis of RobeCzech and XLM-RoBERTa on real-world data from a media website (Chapter 5). The models used were fine-tuned on CNEC 2.0 as they reported the best F scores and the corpus was the largest and the most recent. The analysis has shown us that using the RobeCzech model fine-tuned only on supertypes captures the real-world situation the best, as the CoNLL version lacks the number entity and the version with all types performs worse. In the case of nested NER RobeCzech performed fairly well, but I did not conduct any complex training or analysis on real-world data. Nevertheless, the results indicate that Transformers can build a strong solution for this task and can reduce training time together with the complexity of the solution, and future research could potentially surpass current state-of-the-art solutions.

6.1 Main Contributions

As stated in Section 1.2 I succeeded in the main challenge of my thesis, which was the creation of a strong classifier that solves the NER problem in Czech together with the creation of a modified version of the CNEC dataset consisting solely of flat entities. The proposed solution achieves near-human performance on the dataset and translates well to a real-world situation, as demonstrated in the empirical analysis. I also created a model capable of comprehending nested entities and able to solve easy examples, again shown in empirical analysis.

6.2 Future Works

The CNEC dataset consists of 8,993 Czech sentences compared to CoNLL-2003 [Tjong Kim Sang and De Meulder, 2003] contains 22,137 English sentences, which is almost 2.5 times more or OntoNotes dataset for NER that contains over 60 000 sentences, which is more than 6.5 times more. We could discuss that in future works, the extension of a publicly available dataset could help further improve classifiers for named entities, especially given the complexity of the Czech language. Additionally, with the recent rise of the ChatGPT service based on GPT-3 [Brown et al., 2020] and the increasing emphasis on Large Language Models, it is worth discussing whether these models could also report state-of-the-art results. As shown in Figure 6.1 it could provide some interesting results.



Najdi v těchto větách pojmenované entity pro osobu (P), datum (T), místo (G) a číslo (N)

Karel IV. (14. května 1316, Praha – 29. listopadu 1378, tamtéž), narozen jako Václav, byl jedenáctý český král vládnoucí jako Karel I. od srpna 1346 až do své smrti v listopadu 1378. Byl to první český král, který se stal také císařem Svaté říše římské. Když umřel bylo mu 62 let

Jistě, zde jsou pojmenované entity v těchto větách:

- **Osoba (P):**
 - Karel IV.
 - Václav
- **Datum (T):**
 - 14. května 1316
 - srpen 1346
 - listopad 1378
 - 29. listopadu 1378
- **Místo (G):**
 - Praha
 - tamtéž
- **Číslo (N):**
 - jedenáctý
 - 62

Figure 6.1: An example of ChatGPT answer on NER problem



Bibliography

- [Akbik et al., 2018] Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In Bender, E. M., Derczynski, L., and Isabelle, P., editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- [Anonymous, 2011] Anonymous (2011). czes. <http://hdl.handle.net/11858/00-097C-0000-0001-CCCF-C>. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.
- [Arkipov et al., 2019] Arkhipov, M., Trofimova, M., Kuratov, Y., and Sorokin, A. (2019). Tuning multilingual transformers for language-specific named entity recognition. In Erjavec, T., Marcińczuk, M., Nakov, P., Piskorski, J., Pivovarov, L., Šnajder, J., Steinberger, J., and Yangarber, R., editors, *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93, Florence, Italy. Association for Computational Linguistics.
- [Attardi, 2015] Attardi, G. (2015). Wikiextractor. <https://github.com/attardi/wikiextractor>.
- [Baowan et al., 2017] Baowan, D., Cox, B. J., Hilder, T. A., Hill, J. M., and Thamwattana, N. (2017). Chapter 2 - mathematical preliminaries. In Baowan, D., Cox, B. J., Hilder, T. A., Hill, J. M., and Thamwattana, N., editors, *Modelling and Mechanics of Carbon-Based Nanostructured Materials*, Micro and Nano Technologies, pages 35–58. William Andrew Publishing.
- [Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.
- [Conneau et al., 2020] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale.
- [Cui et al., 2019] Cui, Z., Ke, R., Pu, Z., and Wang, Y. (2019). Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction.

- [Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.
- [Drchal et al., 2023] Drchal, J., Ullrich, H., Mlynář, T., and Moravec, V. (2023). Pipeline and dataset generation for automated fact-checking in almost any language. *arXiv preprint arXiv:2312.10171*.
- [Ferber, 1931] Ferger, W. F. (1931). The nature and use of the harmonic mean. *Journal of the American Statistical Association*, 26(173):36–40.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.
- [Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.
- [Konkol and Konopík, 2013] Konkol, M. and Konopík, M. (2013). Crf-based czech named entity recognizer and consolidation of czech ner research. In Habernal, I. and Matoušek, V., editors, *Text, Speech, and Dialogue*, pages 153–160, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Konkol et al., 2014] Konkol, M., Konopík, M., Ševčíková, M., Žabokrtský, Z., Straková, J., and Straka, M. (2014). CoNLL-based extended czech named entity corpus 2.0. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.
- [Křen et al., 2016] Křen, M., Cvrček, V., Čapka, T., Čermáková, A., Hnátková, M., Chlumská, L., Jelínek, T., Kovářiková, D., Petkevič, V., Procházka, P., Skoumalová, H., Škrabal, M., Truneček, P., Vondříčka, P., and Zásina, A. (2016). SYN v4: large corpus of written czech. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.
- [Lample et al., 2016] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition.
- [Lample and Conneau, 2019] Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining.
- [Lan et al., 2020] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.
- [Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.
- [Loshchilov and Hutter, 2019] Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization.
- [MacWhinney, 2005] MacWhinney, B. (2005). Language evolution and human development. *Origins of the social mind: Evolutionary psychology and child development*, pages 383–410.

- [Majliš, 2011] Majliš, M. (2011). W2C – web to corpus – corpora. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.
- [Majliš and Žabokrtský, 2012] Majliš, M. and Žabokrtský, Z. (2012). Language richness of the web. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2927–2934, Istanbul, Turkey. European Language Resources Association (ELRA).
- [Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality.
- [Mithen, 1997] Mithen, S. (1997). The prehistory of the mind. *Cambridge Archaeological Journal*, 7:269–269.
- [Nakayama, 2018] Nakayama, H. (2018). seqeval: A python framework for sequence labeling evaluation. Software available from <https://github.com/chakki-works/seqeval>.
- [Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.
- [Ramshaw and Marcus, 1995] Ramshaw, L. A. and Marcus, M. P. (1995). Text chunking using transformation-based learning.
- [Ševčíková et al., 2007] Ševčíková, M., Žabokrtský, Z., and Krůza, O. (2007). Named entities in czech: Annotating data and developing NE tagger. In Matoušek, V. and Mautner, P., editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 188–195, Berlin / Heidelberg. Springer.
- [Sido et al., 2021] Sido, J., Pražák, O., Přebáň, P., Pašek, J., Seják, M., and Konopík, M. (2021). Czert – czech bert-like model for language representation.
- [Straka et al., 2021] Straka, M., Náplava, J., Straková, J., and Samuel, D. (2021). Robeczech: Czech roberta, a monolingual contextualized language representation model. In Ekštejn, K., Pártl, F., and Konopík, M., editors, *Text, Speech, and Dialogue*, pages 197–209, Cham. Springer International Publishing.
- [Straková et al., 2016] Straková, J., Straka, M., and Hajič, J. (2016). Neural networks for featureless named entity recognition in czech. In Sojka, P., Horák, A., Kopeček, I., and Pala, K., editors, *Text, Speech, and Dialogue: 19th International Conference, TSD 2016, Brno, Czech Republic, September 12-16, 2016, Proceedings*, pages 173–181, Cham. Springer International Publishing.
- [Straková et al., 2019] Straková, J., Straka, M., and Hajic, J. (2019). Neural architectures for nested NER through linearization. In Korhonen, A., Traum, D., and Màrquez, L., editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.
- [Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.

- [Tjong Kim Sang and De Meulder, 2003] Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv: Computation and Language*.
- [Wang et al., 2021] Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., and Tu, K. (2021). Automated concatenation of embeddings for structured prediction.
- [Yamada et al., 2020] Yamada, I., Asai, A., Shindo, H., Takeda, H., and Matsumoto, Y. (2020). LUKE: Deep contextualized entity representations with entity-aware self-attention. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.
- [Ševčíková et al., 2007a] Ševčíková, M., Žabokrtský, Z., and Straková, J. (2007a). Czech named entity corpus 1.0 | Úfal. <https://ufal.mff.cuni.cz/cnec/cnec1.0>. Accessed 04.04.2024.
- [Ševčíková et al., 2007b] Ševčíková, M., Žabokrtský, Z., Straková, J., and Straka, M. (2007b). Czech named entity corpus. <https://ufal.mff.cuni.cz/cnec>. Accessed: 05.10.2024.
- [Ševčíková et al., 2014a] Ševčíková, M., Žabokrtský, Z., Straková, J., and Straka, M. (2014a). Cnec 2.0 named entities type hierarchy. <https://ufal.mff.cuni.cz/~strakova/cnec2.0/ne-type-hierarchy.pdf>. Accessed: 04.04.2024.
- [Ševčíková et al., 2014b] Ševčíková, M., Žabokrtský, Z., Straková, J., and Straka, M. (2014b). Czech named entity corpus 2.0. <https://ufal.mff.cuni.cz/cnec/cnec2.0>. Accessed: 05.10.2024.



Appendix A

Acronyms

NER	Named Entity Recognition
CNEC	Czech Named Entity Corpus
NLP	Natural Language Processing
LSTM	Long Short-Term Memory
MLM	Masked Language Modeling
CWR	contextualized word representation
ACE	Automated Concatenation of Embeddings
CRF	conditional random field
NSP	Next Sentence Prediction
XLM	cross-lingual language model
IOB	Inside-outside-beginning
seq2seq	sequence-to-sequence
IE	information extraction

Appendix B

Repository Structure

I created a git repository on GitLab <https://gitlab.fel.cvut.cz/factchecking/ner-radek-stulc>. A snapshot of this repository is enclosed and described below:

Description of repository

```
ner-radek-stulc-main
├── CNEC .....prepared CNEC data for fine-tuning and scripts
│   ├── CNEC1_1 ..... version with all types nested and flat
│   ├── CNEC1_1_Supertypes .....version with only supertypes nested and flat
│   ├── CNEC1_1_ext ..... [Konkol et al., 2014] version for 1.1
│   ├── CNEC2_0 .....version with all types only flat
│   ├── CNEC2_0_Supertypes .....version with only supertypes nested and flat
│   ├── CNEC2_0_ext ..... [Konkol et al., 2014] version for 2.0
│   ├── CNEC2_0_nested .....version for nested 2.0
│   ├── eval_scripts .....scripts used for evaluation
│   ├── Czert_results .....results for the Czert model
│   ├── RobeCzech_results ..... results for the RobeCzech model
│   ├── SlavicBert_results ..... results for the SlavicBert model
│   ├── utils .....scripts for evaluation and results
│   ├── Xlm-roberta_results ..... results for the XLM-RoBERTa model
│   └── treex2conll_scripts ..... scripts for the conversion and data
│       ├── data ..... raw data for all version
│       │   ├── CNEC_1.0
│       │   ├── CNEC_1.1
│       │   ├── CNEC_1.1_konkol
│       │   ├── CNEC_2.0
│       │   └── CNEC_2.0_konkol
│       ├── data_tagged .....data in the CoNLL format
│       │   ├── CNEC_1.1 .....contains different data dependant on the desired output
│       │   └── CNEC_2.0 .....contains different data dependant on the desired output
│       └── utils ..... sripts for the conversion
├── Notebooks ..... all notebooks used for training
├── slurm ..... slurm scripts for RCI cluster
└── utils ..... helper python scripts
```