BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DOCUMENT INFORMATION EXTRACTION

EXTRAKCE INFORMACÍ Z DOKUMENTŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. ROMAN JANÍK**
AUTOR PRÁCE

SUPERVISOR **Ing. MICHAL HRADIŠ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

148996

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Janík Roman, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Počítačové vidění
Název:	**Extrakce informací z dokumentů**
Kategorie:	Umělá inteligence
Akademický rok:	2022/23

Zadání:

1. Prostudujte základy zpracování přirozeného jazyka a neuronových sítí.
2. Vytvořte si přehled o současných metodách pro extrakci sémantické informace z textu se zaměřením na metody, které dokáží pracovat s 2D strukturou stránek (např. pojmenované entity, témata, sentiment).
3. Vyberte nebo navrhněte metodu aplikovatelnou pro extrakci informací z českých historických dokumentů jako jsou kroniky, staré noviny, matriční knihy, formuláře nebo rejstříkové lístky.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Sido et al.: Czert - Czech BERT-like Model for Language Representation. arXiv:2103.13031v3, 2021.
- Hubková et al.: Czech Historical Named Entity Corpus v 1.0. Proceedings of The 12th Language Resources and Evaluation Conference, 2020.
- Garncarek et al.: LAMBERT: Layout-Aware (Language) Modeling for information extraction, ICDAR, 2020.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	31.7.2023
Datum schválení:	3.11.2022

Abstract

With development of digitization comes the need for historical document analysis. Named Entity Recognition is an important task for Information extraction and Data mining. The goal of this thesis is to develop a system for extraction of information from Czech historical documents, such as newspapers, chronicles and registry books. An information extraction system was designed, the input of which is scanned historical documents processed by the OCR algorithm. The system is based on a modified RoBERTa model. The extraction of information from Czech historical documents brings challenges in the form of the need for a suitable corpus for historical Czech. The corpora Czech Named Entity Corpus (CNEC) and Czech Historical Named Entity Corpus (CHNEC) were used to train the system, together with my own created corpus. The system achieves 88.85 F1 score on CNEC and 87.19 F1 score on CHNEC, obtaining new state-of-the-art results.

Abstrakt

S rozvojem digitalizace přichází potřeba analýzy historických dokumentů. Důležitou úlohou pro extrakci informací a dolování dat je rozpoznávání pojmenovaných entit. Cílem této práce je vyvinout systém pro extrakci informací z českých historických dokumentů, jako jsou noviny, kroniky a matriční knihy. Byl navržen systém pro extrakci informací, jehož vstupem jsou naskenované historické dokumenty zpracované OCR algoritmem. Systém je založen na modifikovaném modelu RoBERTa. Extrakce informací z českých historických dokumentů přináší výzvy v podobě nutnosti vhodného korpusu pro historickou Češtinu. Pro trénování systému byly použity korpusy Czech Named Entity Corpus (CNEC) a Czech Historical Named Entity Corpus (CHNEC), spolu s mým vlastním vytvořeným korpusem. Systém dosahuje úspěšnosti 88,85 F1 skóre na CNEC a 87,19 F1 skóre na CHNEC. Toto je zlepšení o 1,36 F1 u CNEC a 5,19 F1 u CHNEC a tedy nejlepší známé výsledky.

Keywords

Artificial intelligence, Deep neural networks, Natural Language Processing, Named Entity Recognition, Transformers, Information extraction, historical documents, BERT, RoBERTa, RobeCzech, Czech language processing, Masked language modeling, NER dataset

Klíčová slova

umělá inteligence, hluboké neuronové sítě, zpracování přirozeného jazyka, rozpoznávání pojmenovaných entit, tranformers, extrakce informací, historické dokumenty, BERT, RoBERTa, RobeCzech, zpracování českého jazyka, masked language modeling, NER dataset

Reference

JANÍK, Roman. *Document Information Extraction*. Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Michal Hradiš, Ph.D.

Document Information Extraction

Declaration

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

.....
Roman Janík
July 31, 2023

Acknowledgements

I would like to thank my supervisor Ing. Michal Hradiš, Ph.D., who provided professional help.

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

Contents

Chapter 1

Introduction

The process of digitization in librarianship is progressing since the beginning of the digital era. Digitization is the process of converting documents from analog to digital form. Converting documents from textual form such as books, newspapers, magazines, papers; from audio form such as phonograph records, phonograph cylinders and from graphical form such as photos, photographic films to digital form subsequently brings all the advantages of digital media. Main advantages are easier access, evidence, storage, searching and editing. The next step is the analysis of the document content.

While the process of digitization is satisfactorily resolved for textual data by optical character recognition nowadays, the analysis of document content is still an open problem. Text analysis, also referred to as text mining, is the process of deriving information from text. This includes tasks such as classification (language, format, category, etc.), Sentiment Analysis, Text Summarization, Tagging, Relationship extraction, Question Answering and Named Entity Recognition. The latter is a main interest of this thesis. The ability to understand text and extract information from text by computers has many useful applications. For example Semantic Search – searching with meaning to obtain more relevant answers, targeted advertising – showing relevant advertisement to increase profits and Machine Translation – translation from one natural language to another.

Today's state-of-the-art methods usually solve text analysis tasks by machine learning. For this, a suitable text datasets are needed. These datasets are also referred to as text corpora. This thesis deals with historical Czech documents, such as newspapers, chronicles and registry books. As Czech is language with relatively small number of speakers, the amount of available data is smaller than for world languages, e.g. English. Also historical Czech documents are written in historical Czech, which is different from contemporary Czech, so the corpus needs to be adjusted. There is also a problem with data scarcity of historical documents due to lesser amount of data created, preserved until today and digitized.

The main goal of this thesis is to develop a system for extraction of information from Czech historical documents. The system takes scanned historical documents processed by the OCR algorithm as input and outputs named entities. To achieve this goal, contemporary methods of natural language processing, and neural networks for information extraction needs to be studied. Another goal is to gather a suitable corpus, part of the corpus needs to be created manually and automatically. Further goals are designing, training and experimenting with the system. Final goal is to evaluate achieved results and discuss possible development.

This thesis is a part of project PERO, which aims to create technology and tools which would improve accessibility of digitized historic documents. The tools created during this project will enable existing digital archives and libraries to provide full-text search and content extraction for low quality historic printed and all hand written documents.

This thesis is organized as follows: chapter 2 contains an introduction to natural language processing, contemporary algorithms for information extraction, training of models and metrics used for evaluation. Chapter 3 discusses datasets for Named Entity Recognition. System is designed in chapter 4, including description of input data and proposed datasets. Chapter 5 describes the implementation of the system. Experiments and their results are described in chapter 6. Chapter 7 discusses results and possible development.

Chapter 2

Information extraction

Information extraction is the task of extracting structured information from unstructured data. In context of this thesis, the unstructured data are Czech historical documents, such as newspapers, chronicles and registry books. The goal will be achieved by Named Entity Recognition, which is a task that detects occurrences of named entities and classifies them into predefined types. The outputs then can be used for other natural language processing tasks, e.g. Semantic Search, Text Summarization and Question Answering.

This chapter explains necessary background needed to solve the task. Starting with distributed word representations, description of language models follows. Named entity recognition is elaborated further, and last is metrics and evaluation.

2.1 Distributed word representations

In human languages, words are the smallest units that have a meaning. Important part of language understanding is the understanding of meanings of words [32]. Naturally, the same applies for natural language processing (NLP) models, as they need to represent words in order to work.

Neural network models use representation learning to learn word representations, also referred to as embeddings. These word representations are learned from vast amounts of text in an unsupervised manner. Usual way to represent words is as distributed representations, which are low dimensional real-valued dense vectors [30]. The meaning of dense vectors is that one vector dimension represents multiple concepts and one concept is represented by multiple dimensions. Distributed word representations encode both syntactic and semantic word relationships [36].

Representation learning can be viewed as learning a mapping from words to distributed word representations. Words are placed (embedded) into a multi-dimensional vector space. The number of dimensions varies from 50 into earliest works (e.g. [20]) to usual maximum 1 024 in recent works (e.g. [58]). Words with similar meaning should be placed close to each other.

Distributed word representations are produced by language models described in subsection 2.2. Once learned they can be used as input to task-specific model or used it the original language model modified to down-stream task. In both cases, distributed word representations can be either frozen (fixed) or fine-tuned. Most significant advantage of using pre-trained word representations is that they model the language much better than randomly initialized word representations. As most NLP tasks require manually annotated

data, task-specific supervised datasets are far smaller in size than unsupervised datasets for language modeling. Because of that, most models in last decade use pre-trained distributed word representations. Another advantage of language model approach is that word representations can be trained once and then used in many different NLP task. It also saves a computational time, as the training of language model is expensive, but done only once and task-specific models are far cheaper to train. I.e. many experiments can be run on top of pre-trained word representations.

2.1.1 Context-free word representations

Context-free word representations, also called traditional pre-trained word representations (embeddings), represent a word by single vector. Most notable examples are Word2vec [36], GloVe [38] and fastText [19]. In Word2vec, word representations are learned by training Skip-gram and Continuous bag-of-words models. Skip-gram learns word representations that can predict the nearby words. Contrary to Skip-gram, Continuous bag-of-words model predict the center word representation in a context window. When was Word2vec published, it was a groundbreaking work and since then context-free pre-trained word representations are used as integral part of NLP systems. An interesting fact is that Skip-gram word vectors can be added and subtracted; and resulting vectors have corresponding meaning. This can be illustrated in Figure 2.1: vec("Italy") + vec("capital") is close to vec("Rome").

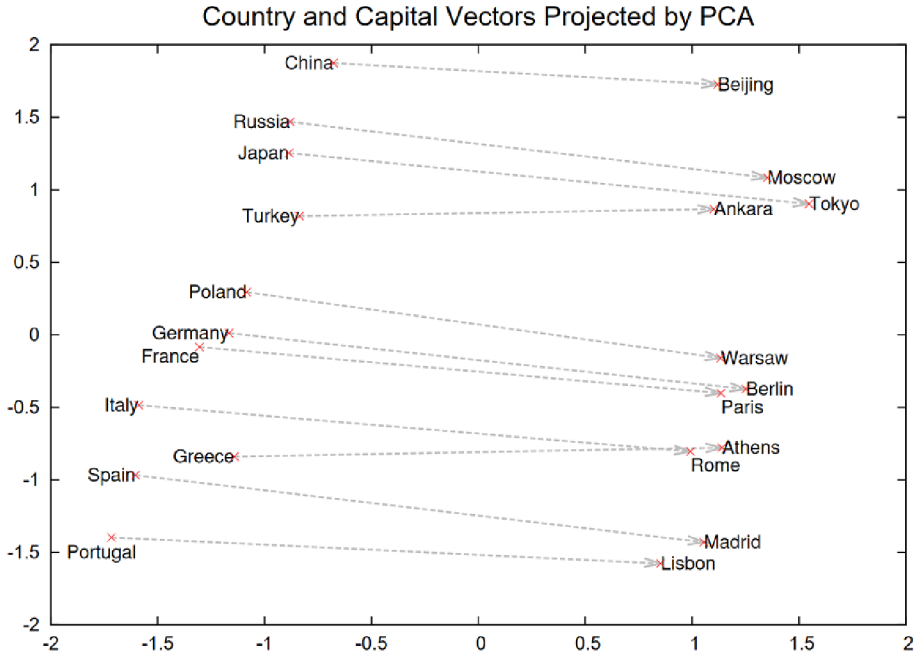


Figure 2.1: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. This illustrates that learned word vectors capture concepts and the relationships between them [36].

The main limit of context-free word representations is that a word has only one vector. However, some words have multiple meanings based on linguistic context (polysemy). This problem is addressed by contextualized word representations (CWRs). Nevertheless,

context-free representations are still used in NLP models, either as input for CWRs language models or as one of multiple embedding types.

2.1.2 Contextualized word representations

Contextualized word representations (CWRs) are able to distinguish between multiple meanings of word based on entire word context. One of the first works, which introduced CWRs, is ELMo (Embeddings from Language Models) [39]. Unlike traditional context-free word representations that assign one static vector to a word, in ELMo words are assigned a representation that is a function of the entire input sentence. To get a word representation, word and its sentence is fed into the ELMo deep neural network [32]. Table 2.1 shows the comparison of results of nearest neighbors search in GloVe and ELMo word representations.

In recent years, the use of CWRs significantly improved the results on all NLP tasks. Therefore, state-of-the-art models use CWRs either as features or for fine-tuning.

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
ELMo	Chico Ruiz made a spectacular play on Alusik's grounder {. . . }	Kieffer, the only junior in the group, was commended for his ability to hit in the clutch, as well as his all-round excellent play.
	Olivia De Havilland signed to do a Broadway play for Garson {. . . }	{. . . } they were actors who had been handed fat roles in a successful play, and had talent enough to fill the roles competently, with nice understatement.

Table 2.1: Demonstration of nearest neighbors search in GloVe and ELMo word representations for a word "play" [39].

2.2 Language modeling

Language models learn distributed word representations by training on vast amounts of text data in an unsupervised manner. They do so by predicting a probability of given sequence of words. Recent neural network models are based on either long short-term memory (LSTM) or Transformer [56] units. This section describes recent important language models.

2.2.1 ELMo

ELMo (**E**mbeddings from **L**anguage **Mo**dels) [39] is an important language model that produces CWRs. It is based on two language models (LM): forward and backward language model. In forward LM, probability of given sequence $(t_1, t_2, \ldots, t_N)$ of N tokens (words) is computed as:

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k \,|\, t_1, t_2, ..., t_{k-1}) \tag{2.1}$$

where t_k is one of the sequence tokens and $(t_1, t_2, ..., t_{k-1})$ are previous tokens. In this manner, LM learns to predict next token t_{k+1} of given context.

The backward LM is similar to the forward LM. As can be expected, it differs from forward LM in that the input sequence is reversed and tokens are predicted given the future context:

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, ..., t_N) \tag{2.2}$$

The contextual representation of each token is the concatenation of forward LM and backward LM representations. The architecture is shown in Figure 2.2, each ELMo language model consists of two bidirectional LSTM (Bi-LSTM) layers, the input word representations are computed by convolutional neural network (CNN). Word representations are computed from representations of each character. This approach enables the model to infer representations for word that are not in the training set [39]. The top layer of both LMs is a softmax. Bi-LSTM layers have 4 096 LSTM units and compute 512 dimension representations. ELMo adopts feature-based approach for solving specific-tasks.

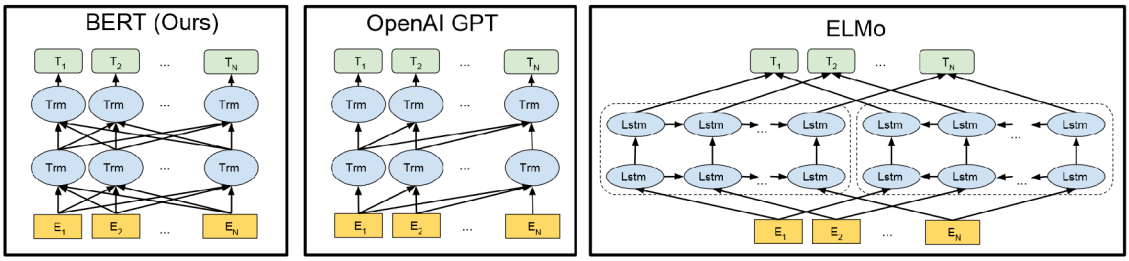


Figure 2.2: Comparison of architectures of BERT, OpenAI GPT 1 and ELMo models (*Trm* denotes transformer) [22].

2.2.2 BERT

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) [22] is the first deep bidirectional language model. Major importance of this LM is this ability to learn from left and right context at the same time. Contrary to ELMo, which uses concatenation of left-to-right (forward) and right-to-left (backward) LMs, BERT learns deep bidirectional representations in all layers. To achieve this, BERT is trained by newly introduced pre-training task, called "Masked Language Model". BERT advanced the state-of-the-art results for eleven NLP tasks. Since publishing, many works on improving BERT were proposed.

Model architecture

Architecture of BERT is a multi-layer bidirectional Transformer, respectively the encoder part of Transformer. There are two model sizes: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. The number of Transformer layers is 12, the number of word representations dimensions is 768 and there 12 attention heads for $\text{BERT}_{\text{BASE}}$. $\text{BERT}_{\text{LARGE}}$ has 24 layers, 1 024 dimensions and 16 attention heads. The architecture is shown in Figure 2.3. The input word representations are calculated as a sum of token (word), position and segment representations. Position representations encodes position of token in the input sequence, segment representations denote if token belongs to sentence A or B in a pair sentence input scenario. Here, sentence

refers to word sequence, not a linguistic sentence. Sentence pair is at the input for pair-sentence tasks, such as BERT's next sentence prediction and Question Answering.

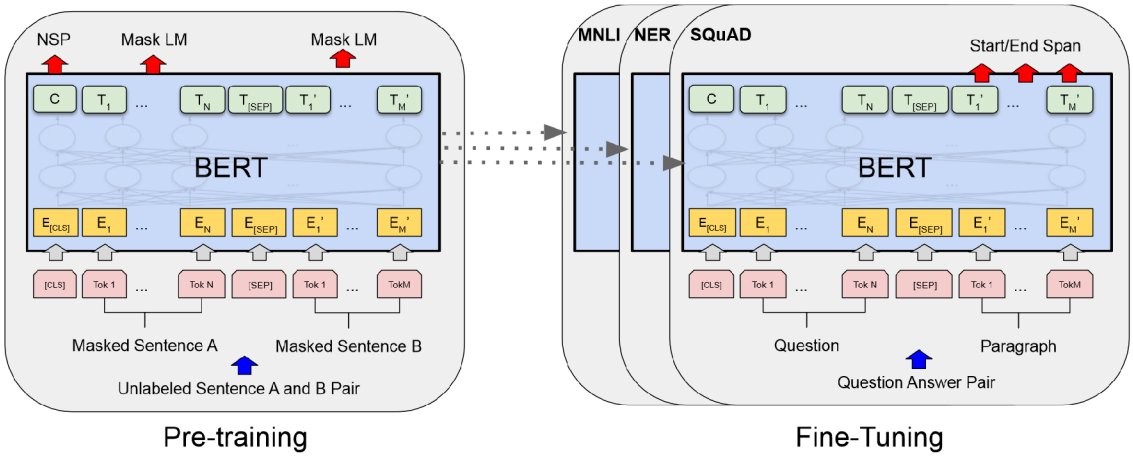


Figure 2.3: Architecture of BERT model [22].

Pre-training

BERT is pre-trained with two unsupervised tasks. First is Masked Language Model (MLM), second is next sentence prediction (NSP). Unlike standard language models, which are either left-to-right (forward) or right-to-left (backward), bidirectional language model cannot be trained both ways because a word would indirectly "see itself" [22]. In MLM task, given an input sequence, some tokens are randomly masked and the model predicts the masked tokens. In BERT, 15% of tokens in a sequence are randomly masked. When a token is chosen to be masked, in 80% of the time the token is replaced by special `[MASK]` token, in 10% of the time the token is replaced by random token and in 10% of the time the token is unchanged. This distribution helps the model learning. Similarly to standard LM, last hidden vectors of the masked tokens are fed into a softmax layer. Model is pre-trained with a cross-entropy loss.

Next sentence prediction task is important for sentence-level downstream tasks, such as Question Answering and natural language inference (textual entailment). With NSP, model learns to understand a relationship between two sentences. Given two sentences `A` and `B` sampled from a text corpus, the model is trained to predict if sentence `B` follows sentence `A`. A pre-training example consist of two sentences `A` and `B`, where in 50% of the time `B` is an actual next sentence of `A` and in 50% of the time `B` is a random sentence from the corpus. The sentence information is encoded in segment embeddings.

The model is pre-trained on BooksCorpus and English Wikipedia datasets, with total 3.3 billion words. The pre-training example has ≤ 512 tokens and both pre-training tasks are used at the same time. Adam optimizer is employed.

Named entity recognition experiments

Results for NER are reported on CoNLL-2003 [53] dataset. BERT paper presents fine-tuning and feature based approach for NER task. In the case of fine-tuning, a linear classification layer is added on top of the pre-trained model and all parameters are fine-

tuned. A multi-class cross entropy loss is used. BERT achieves 92.4% F-score with BASE model and 92.8% F-score with LARGE model on test ConLL-2003 dataset.

In the case of feature-based approach, model parameters are extracted without fine-tuning. Many different methods were explored with $\text{BERT}_{\text{BASE}}$ model and concatenation of last four hidden layers yields best result 96.1% F-score on development CoNLL-2003 dataset [22]. These contextual word representations are then fed into a randomly initialized two-layer BiLSTM model. Token entity types are produced by a linear classification layer on top of BiLSTM model. Test dataset results are not reported in the case of feature-based approach; however with fine-tuning approach, $\text{BERT}_{\text{BASE}}$ achieved 96.4% F-score on development dataset, which is only 0.3% better.

2.3 Named entity recognition

Named entity recognition (NER) is the task of detecting and classifying named entities in text into predefined entity categories [30]. "A named entity is defined as word or phrase that clearly identifies one item from a set of other items that have similar attributes" [48]. In general domain, such named entities are names of persons, locations, organizations and artifacts. Some works also include time, currency and percentage expressions. In [30], NER is formally defined as function that takes a sequence of tokens $s = \langle w_1, w_2, \ldots, w_N \rangle$ as input, and outputs list of tuples $\langle I_s, I_e, t \rangle$. Tuple represents a named entity mention, where $I_s \in [1, N]$ is start index and $I_e \in [1, N]$ is end index and t is an entity type (category). Example of a NER system is shown in Figure 2.4. The definition of a named entity types depends on the problem domain and the task goal. Classification of named entities into a small set of types (typically < 10) is called as coarse–grained classification. When more entity types are needed, classification is called fine-grained. The evaluation of NER is described in section 2.4.

Named entity recognition is an important task of natural language processing (NLP). NER is by itself an information extraction tool and also serves as pre-processing step for many NLP applications, such as Semantic Search, Text Understanding, Question Answering, Machine Translation, Text Summarization, etc. [30]. Real-world applications examples include search engines, content recommendation, customer support and digital libraries [23].

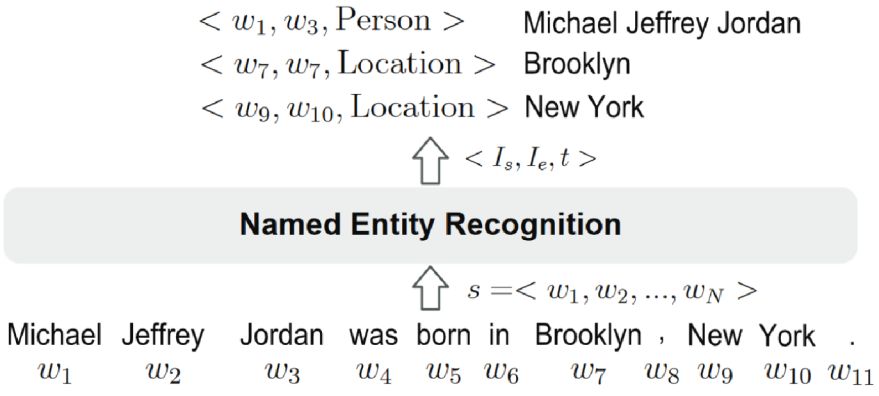


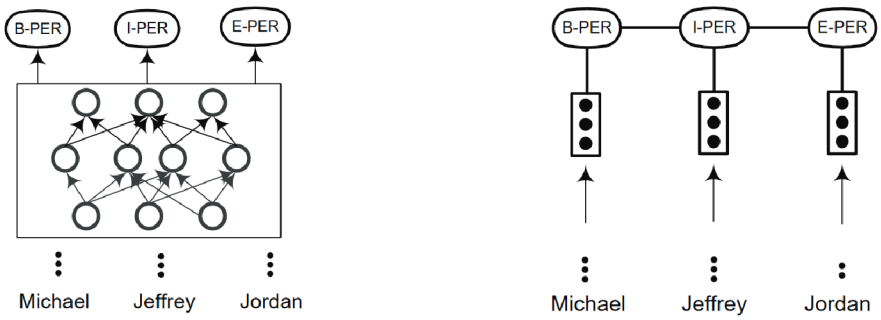
Figure 2.4: An illustration of the Named Entity Recognition task [30].

2.3.1 Methods

The evolution of NER methods is no different to other NLP tasks. Starting with rule-based methods, research then moved to unsupervised learning methods and subsequently to supervised learning methods based on handcrafted features. In recent years, state-of-the-art NER methods are based on deep learning with neural networks [30]. First influential work using neural networks for NER is the work of Collobert et. al. [20]. The authors proposed a neural model, where its architecture is based on temporal convolutional networks [30]. The model uses representation learning and can be also applied to other NLP task such as part-of-speech Tagging, chunking and semantic role labeling. Research then moved to recurrent neural networks, as they yield better results on sequential data. Since the groundbreaking work of Vaswani et al. [56], that introduced transformers, state-of-the-art NER models are based on transformer architecture. For detailed description of word representations and model architectures, see sections 2.1 and 2.2, respectively.

Neural NER models consist of three parts: distributed word representations for input, context encoder and tag decoder [30]. First two parts are usually common for many other NLP task as well. There are two possible approaches to utilize pre-trained language models: *feature-based* and *fine-tuning* [22]. In the case of the feature-based approach (e.g. ELMo [39]), pre-trained distributed word representations are used as input in task-specific architectures. In the fine-tuning approach, pre-trained language model and its word representations are fine-tuned to each specific task with minimal changes to model architecture (mostly adding classification layer on top of model), for example OpenAI GPT [16]. Context encoder architectures mostly use bidirectional LSTM or transformers.

The final stage of NER model is the tag decoder. Its input is context-dependent representations and output is a sequence of named entity tags [30]. There are two prevalent tag decoder architectures: multi-layer perceptron + softmax layer, and conditional random fields (CRF). NER can be considered as sequence labeling problem. With a multi-layer perceptron + softmax layer, the problem is simply a multi-class classification, where each word tag is independently computed from context-dependent representations. Neighbor representations are not taken into account [30]. An example work that uses a multi-layer perceptron + softmax layer as tag decoder is BERT [22]. "A conditional random field (CRF) is a random field globally conditioned on the observation sequence" [30]. Often used together with LSTM and Bi-LSTM, CRFs are mostly used for models following feature-based approach. Both architectures are shown in Figure 2.5.



(a) MLP+Softmax (b) CRF

Figure 2.5: Multi-layer perceptron + softmax and conditional random fields tag decoder architectures [30].

Models using pre-trained contextualized word representations (CWRs) achieve excellent results, for example original BERT model [22] achieves 92.8% F-score on CoNLL-2003 dataset [53]. However, there are some issues: many named entities span multiple tokens in the model, and CWRs models provide representations only for each word (token), therefore making reasoning about relationships between entities difficult. Relationship reasoning is a vital part of tasks such as Question Answering and Relation Classification. Additionally, CWRs pre-training is word-based, and the task of predicting a single masked word in a multi-word entity is easier than predicting the entire entity [58]. Because of that, CWRs model does not learn multi-word entity representations. To solve these problems, entity representations models were introduced. Such models can be split into two categories: *static entity representations* and *contextualized word representations enhanced by knowledge injection*. In static entity representations scheme, each entity in the knowledge base is assigned a fixed embedding vector [58]. There are two main disadvantages of this approach, entities, that do not exist in the knowledge base, cannot be represented; and they require entity linking to represent entities in a text [58]. CWRs enhanced by knowledge injection do not have such disadvantages. An example of this category is LUKE [58], described in next subsection.

2.3.2 LUKE

LUKE (**L**anguage **U**nderstanding with **K**nowledge-based **E**mbeddings) is a recent neural language model that learns not only contextualized word representations, but also contextualized entity representations [58]. At the time of publishing, LUKE was a state-of-the-art model for NER, with 94.3% F-score reported on CoNLL-2003 [53] dataset. LUKE results on NER were surpassed by only one work, as stated on NLP-progress website [42]. The ability of learning entity representations helps the model to solve entity-related tasks, such as Named Entity Recognition, Relation Classification, Entity Typing and Question Answering. The model is based on transformer architecture [56] and is trained using a new pre-training task, where entities are taken into account.

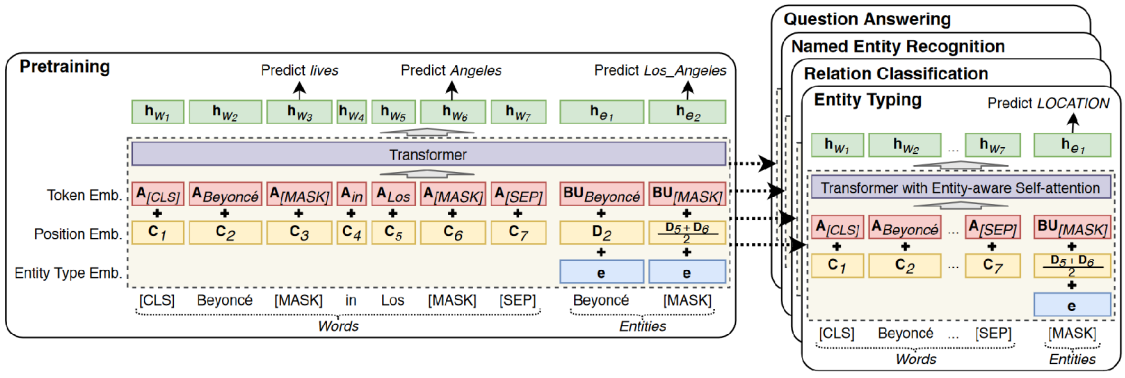


Figure 2.6: Architecture of LUKE using the input sentence "*Beyoncé lives in Los Angeles.*" LUKE outputs contextualized representation for each word and entity in the text. The model is trained to predict randomly masked words (e.g., *lives* and *Angeles* in the figure) and entities (e.g., *Los_Angeles* in the figure). Downstream tasks are solved using its output representations with linear classifiers. [58].

Model architecture

Architecture of LUKE is based on RoBERTa$_{\text{LARGE}}$ model [31], which is a variant of BERT [22]. The architecture is shown in Figure 2.6. Words and entities in the input sequence are treated as input tokens, and a representation for each token is computed. Input representations of a word consist of a token embedding and position embedding. Token embedding is decomposed into two matrices $\mathbf{B}$ and $\mathbf{U}$, for computational efficiency. Token position is represented by position embedding. In the case of multi-word entity, an average of the corresponding positions is taken. Entity tokens are represented by an entity embedding. For word token, the input representation is calculated as sum of token and position embeddings, whereas for entity token, it is as sum of token, position and entity embeddings. Masked tokens are represented by `[MASK]`. First and last words in the word sequence are marked by special tokens `[CLS]` and `[SEP]`. There is also a special entity token `[UNK]` for entities that do not exist in the model vocabulary. The configuration of the model is following: transformer has $D = 1024$ hidden dimensions, 24 hidden layers, $L = 64$ attention heads, and 16 self-attention heads. Entity embeddings dimensions are $H = 256$. The model uses RoBERTa's tokenizer. Size of vocabulary is set to $50K$ words, while the entity vocabulary size is set to $500K$.

Entity-aware self-attention

The authors proposed an entity-aware self-attention mechanism. It is a modification of the original transformer self-attention. As can be seen in model architecture in Figure 2.6, both words and entities are inputs to transformer model. This means that both word and entity tokens attend to word and entity tokens in the previous layer. In the original self-attention mechanism, the i-th output vector $\mathbf{y}_i$ is computed as:

$$\mathrm{y}_i = \sum_{j=1}^{k} \alpha_{ij} \mathbf{V} \mathbf{x}_j \tag{2.3}$$

$$e_{ij} = \frac{\mathbf{K}\mathbf{x}_j^\top \mathbf{Q}\mathbf{x}_i}{\sqrt{L}} \tag{2.4}$$

$$\alpha_{ij} = \mathrm{softmax}(e_{ij}) \tag{2.5}$$

where $\mathbf{x}_i \in \mathbb{R}^D$ are input vectors, $\mathbf{y}_i \in \mathbb{R}^D$ are output vectors, k denotes the length of input and output sequences, L denotes attention head dimensions, $\mathbf{Q} \in \mathbb{R}^{L \times D}$ is query, $\mathbf{K} \in \mathbb{R}^{L \times D}$ is key, and $\mathbf{V} \in \mathbb{R}^{L \times D}$ is value matrix. The idea of the entity-aware self-attention mechanism is to differentiate the attention between word and entity tokens. Therefore four query matrices are introduced for each combination of word and entity tokens: $\mathbf{Q}, \mathbf{Q}_{w2e}, \mathbf{Q}_{e2w}, \mathbf{Q}_{e2e} \in \mathbb{R}^{L \times D}$. This mechanism helps model to better capture relationships between entities [58]. Note that in LUKE the entity-aware self-attention is used only for fine-tuning training tasks.

Pre-training

The model pre-training follows BERT Masked Language Model (MLM) task and adds a new pre-training task that leverages entities. The authors downloaded a Wikipedia dump from December 2018 and created the pre-training dataset by splitting the content of each page into sequences of ≤ 512 words. Entities annotations are generated from Wikipedia

hyperlinks. Similarly to the original MLM task, entities (respectively their word span for multi-word entities) are masked by special `[MASK]` token in the new pre-training task. This way the model learns representations of whole entity word span. As in BERT, 15% of all words and entities are randomly masked. The pre-training loss is a sum of MLM loss and cross-entropy loss for the new entity pre-training task. Model is optimized by the AdamW optimizer. Both model parameters and word embeddings from RoBERTa were used for initialization of LUKE model to reduce overall training time.

Named entity recognition experiments

LUKE NER experiments are conducted on CoNLL-2003 dataset [53]. The model is trained as follows: First, for each sentence, all possible spans are enumerated as named entity candidates. These spans are then classified as one of entity types or non-entity type. Second, sentence is input into model. The sentence has `[MASK]` entity tokens in the places corresponding to all enumerated spans. Lastly, the word representations of first and last words in each span, and the entity representation corresponding to the span are concatenated to form span representation. This span representation is used to classification of the span, by feeding it into a linear classifier. Cross-entropy loss is used for model training. Spans consisting of ≥ 16 words were excluded in the training to save computing power. Also, such long entity names should be very rare. At inference time, firstly all non-entity type spans are ignored. Greedily selection of spans from the remaining spans is employed. To prevent overlapping spans selection, a span is selected based on probability of its predicted entity type, only if it does not overlap with already selected spans.

2.3.3 RobeCzech

RobeCzech – Czech RoBERTa [51] is the current Czech state-of-the-art NER model. Results (see table 2.2) are reported on CNEC 1.1 and CNEC 2.0 [47] datasets, as well as on their CoNLL versions [29]. While CNEC 1.1 and CNEC 2.0 contain nested (embedded, overlapping) entities, their CoNLL versions contain only flat (non-nested) entities. Note that results are evaluated on fine-grained classification, where 42 named entity types are predicted [52][51].The paper also presents results for morphological analysis and lemmatization, dependency parsing, semantic parsing and Sentiment Analysis tasks, where for the first two tasks a feature-based approach is adopted, and for the latter two a fine-tuning approach is adopted.

Pre-training

RobeCzech represents a Czech version of RoBERTa [31] model. The results on all tasks demonstrates the superiority of RoBERTa model, where RobeCzech has better results than the other Czech BERT-based [22] models, such as Czert [49] and Slavic BERT [17]. The model is trained on a large corpus of various sources with total size 4 917M tokens. The training batch size is set to 8 192. Each batch consists of contiguously sampled sentences. Maximum length of each sample is 512 tokens. Adam optimizer is used, the number of optimization steps is 91 075 and the resulting training time is approximately 3 months on 8 QUADRO P5000 GPUs [51]. The total number of model parameters is 125M.

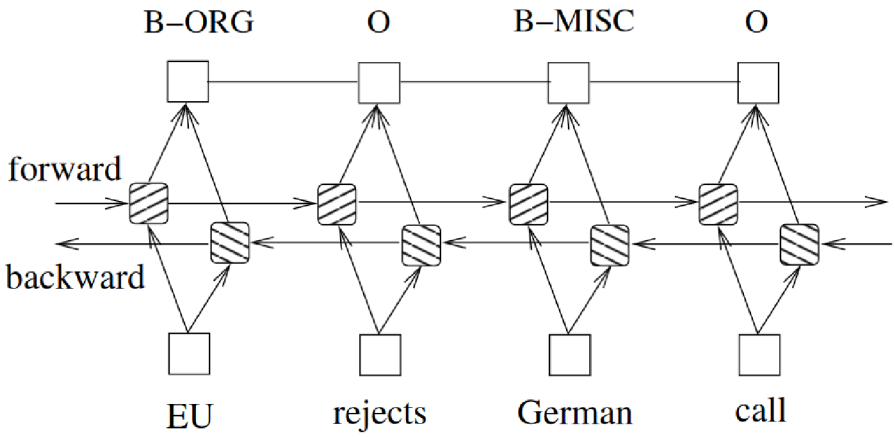


Figure 2.7: A Bi-LSTM-CRF model architecture [25].

Named entity recognition experiments

The authors adopted their previous work NameTag 2 [52], where two models are presented: LSTM-CRF and seq2seq. LSTM-CRF is used for flat NER, seq2seq for nested NER. Both are feature-based approaches. Architecture of the LSTM-CRF consists of bidirectional long short-term memory network (Bi-LSTM, see 2.7) as context encoder and conditional random fields (CRF) layer as tag decoder. The sequence-to-sequence (seq2seq) model uses Bi-LSTM as context encoder and LSTM as tag decoder. The models are trained on FastText [19] word embeddings, end-to-end word embeddings, which include lemmas, input forms and part-of-speech tags; and bidirectional gated recurrent unit made character-level word embeddings [52]. In RobeCzech, NER models use mentioned embeddings and RobeCzech contextualized word embeddings as additional inputs. RobeCzech CWRs are not fine-tuned during training [51]. Adam optimizer is used for training; batch size is set to 8. The authors apply 0.5 dropout and for word dropout they replace 20% of words by the unknown token. The word dropout should force the model to more rely on context [52]. At the time of releasing, NameTag 2 reported state-of-the-art results for English on CoNLL-2003 dataset [52].

CNEC 1.1	CNEC 2.0	CoNLL CNEC 1.1	CoNLL CNEC 2.0
87.82	85.51	87.47	87.49

Table 2.2: RobeCzech reported F-score for Czech NER. Nested entities use seq2seq model, Flat (CoNLL) entities use LSTM-CRF model [51].

2.4 Metrics and evaluation

NER systems are evaluated by three metrics: Precision, Recall and F-score. The ground-truth data are created by human annotator [30]. There is an exact match and a relaxed match.

2.4.1 Exact match

Named entity is considered as correctly recognized, when NER system both detects its span and classifies its true type [30]. Common classification terms are specified as follows:

- **True Positive (TP)** - entity is returned by the NER system and is in the ground truth.
- **True Negative (TN)** - entity is not returned by the NER system and is not in the ground truth.
- **False Positive (FP)** - entity is returned by the NER system and is not in the ground truth.
- **False Negative (FN)** - entity is not returned by the NER system and is in the ground truth.

Precision is defined as the ratio of correctly recognized entities in the NER system results.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

Recall is defined as the ratio of correctly recognized entities out of all correct entities.

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

F-score or F-measure is the harmonic mean of precision and recall.

$$F - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.8}$$

F-score is sometimes denoted as F1. Some works also states the micro-averaged and macro-averaged F-score. Micro-averaged F-score is calculated from all individual true positives, false positives and false negatives. Macro-averaged F-score is calculated as average of each entity type F-score [30].

2.4.2 Relaxed match

Some works report relaxed match results. The definition of this term varies among works. For example in BSNLP-2019 [40], work that deals with NER on documents, an entity is considered as correctly recognized if at least one annotation of named mention of this entity is returned by the NER system. In CNEC [46], relaxed match is defined as correct detection of a named entity span, or as correct detection of a named entity span and correct classification of super-type (first level type).

Chapter 3

Datasets

This chapter contains description of datasets for Named Entity Recognition (NER), their comparison and discusses problems with historical texts and datasets properties. For better understanding of this chapter, please refer to section 2.3 describing Named Entity Recognition. A suitable datasets are essential for Named Entity Recognition (NER). They are needed for model training as well as evaluation. NER dataset or corpus is a collection of real-world documents annotated by entity types. Words in a dataset are tagged by entity type or by special tag coding no-entity, in some datasets words that are not a part of named entity are not tagged. Some datasets also contain additional information for other tasks, e.g. part-of-speech (POS). An important aspect is that entities are contained within real-world text, so NER algorithms can learn from their contexts.

Entity types and their number are defined by the objective of the dataset and differ from dataset to dataset. Most common entity types are: person, location, organization and object. There may be two-level hierarchy of entity types to achieve fine-grained results, for example a subtype of person is a surname. Some datasets, e.g. CoNLL-2003 [53], also differentiate between the word at the beginning of and the rest of words of a multi-word named entity. NER datasets are created either by collecting documents and manually annotating them or by generating from sources like Wikipedia. First way representative work is [46], second is [58]. The objective of the dataset also defines the text source from which dataset was created. According to list in [30], there are datasets created from news articles, Wikipedia articles, social media user comments such as tweets and YouTube comments. Medical text datasets are also listed as representative of domain specific datasets. Naturally, most datasets are in English language as it is most used language today, fewer datasets are in other world languages like German and Spanish. For Czech language a limited number of dataset exist, as fewer authors are interested in Czech NER, most of them are from Czechia or Central European countries.

In general, a suitable NER dataset for given task has these properties: text source should as close as possible to target domain in terms of language, style, field and vocabulary; dataset should contain minimal number of miss-annotated named entities and should be large; so a various forms of named entities placed in a various context can be recognized. Therefore larger dataset contributes to better results.

3.1 Historical content

As this thesis deals with historical newspapers, chronicles and registry books, the dataset to solve this task needs to be adjusted to their content. Historical newspapers can be just considered as news with historical content. The style of chronicles and registry books is presumed to be more formal with fewer phrases and more time and location expressions. Historical Czech texts differ from contemporary Czech, especially in vocabulary, word forms, spelling and word order [27]. Regarding vocabulary, there are archaic words, or words of things that does not exist at present time, e.g. "telegraf" – *telegraph*. Some names, that existed in past, changed, notable location names, and refer to its present day equivalent. Another issue is archaic personal names. In the case of spelling, there are differences that today would be considered a spelling mistake (e.g. "výtězně", in contemporary Czech "vítězně" – *triumphantly*) [27]. In the case of word forms, different verb and noun forms can be found, for example "zástupcové" – nowadays "zástupci", in English *representatives* [27]. Also different phrases are used. Different word order is also a problem. While the contemporary news uses adjectives as premodifiers, in historical news adjectives are used as postmodifiers. Notable example is "německá říše" and "říše německá" (*German Empire*) in historical news [27]. Given the historical realities, a less literary language can be expected. Dialects were more used than in contemporary Czech. Slang may also appear, for example in company chronicles.

3.2 CoNLL-2003

CoNLL-2003 [53] is considered as standard English dataset for comparing results of NER systems. The dataset consist of English and German news articles, the English part are an annotated Reuters news, the German part are an annotated news from German newspaper Frankfurter Rundshau. Both news collections are from 1990s. Each language data is divided in standard training, development and test parts; plus there is a large file of unannotated data [53]. As stated above, CoNLL-2003 introduced tags for first word and the rest of words of multi-words named entity. Named entities are non-overlapping and non-recursive, the latter means that multi-word named entities do not contain another named entities, e.g. in named entity "United states of America", "America" is not tagged as a named entity. Example of annotations is shown in Figure 3.1. There are four types of named entity types: persons, locations, organizations and miscellaneous. The dataset contains total 10 059 persons, 10 645 locations, 9 323 organizations and 5 062 miscellaneous entities in 301 418 tokens for English and 5 369 persons, 6 579 locations, 4 441 organizations and 3 968 miscellaneous entities in 301 418 tokens for German [53]. The dataset is highly influential and many NER datasets follows its structure.

3.3 Czech NER datasets

3.3.1 Czech named entity corpus

Czech named entity corpus (CNEC) from 2007 is the first known Czech dataset for NER [46]. The dataset was extended, the last version CNEC 2.0 is from 2014 [47]. CNEC 1.0 dataset contains total 2 000 sentences randomly selected from the Czech National Corpus [15]. Czech National Corpus is significant source of Czech corpuses for development of natural language processing applications. In CNEC 2.0, there is total 8 993 sentences, 35 220

U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

Table 3.1: Example of CoNLL annotations in sentence "*U.N. official Ekeus heads for Baghdad.*". [53]

	CNEC 2.0	CHNEC	BSNLP-2019	SumeCzech-NER
Size	8 993 sentences	8 251 words	-	1 000 000 articles
Named entities	35 220	4 017	9 877	30M
Named entity types	46	5	5	7
Annotations	manual	manual	manual	automatic
Year	2014	2020	2019	2021

Table 3.2: Overview of Czech NER datasets.

named entities classified into 46 entity types. In comparison with previous version, the named entity hierarchy was modified [47]. The named entity hierarchy classifies the entity types in two-levels, first level for a coarse-grained classification of total 8 entity types, which then expands into total 46 entity types, as can be seen in Figure 3.1. The dataset contains recursive named entities. CNEC represents a large Czech NER dataset of general style, suitable as basis for further work. This dataset was transformed to CoNLL format to enable comparison with other languages [29].

3.3.2 Czech historical named entity corpus

Czech named entity corpus (CHNEC) from 2020 is the first known historical Czech dataset for NER [27]. Its so far first version CHNEC 1.0 is made of annotated historical newspaper *Posel od Čerchova* from the second half of 19th century. The original newspaper were scanned and digitized by optical character recognition. The importance of this dataset is that the source is historical texts, close to the input data in this work in a fashion described in the subsection 3.1. As the name of the CHNEC dataset suggests, the words in dataset are in Czech, but there are also some limited number of words from German, French and Latin. In the CHNEC 1.0, there are 8 251 words, 4 017 named entities and 5 named entity types: personal names, institutions, geographical names, time expressions and artifact names / objects.

3.3.3 BSNLP 2019

BSNLP 2019 dataset was published a part of shared task at the 7th workshop on Balto-Slavic Natural Language Processing (BSNLP) [40]. The dataset consist of converted Web pages, mainly news articles, divided into 4 sets of documents, each centered around one topic: *Asia Bibi*, *Brexit*, *Ryanair* and *Nord stream 2*. As the task was multilingual, it covers four languages: Bulgarian, Czech, Polish and Russian. There are 5 named entity types: person, location, organization, product, and event. Total number of named entities

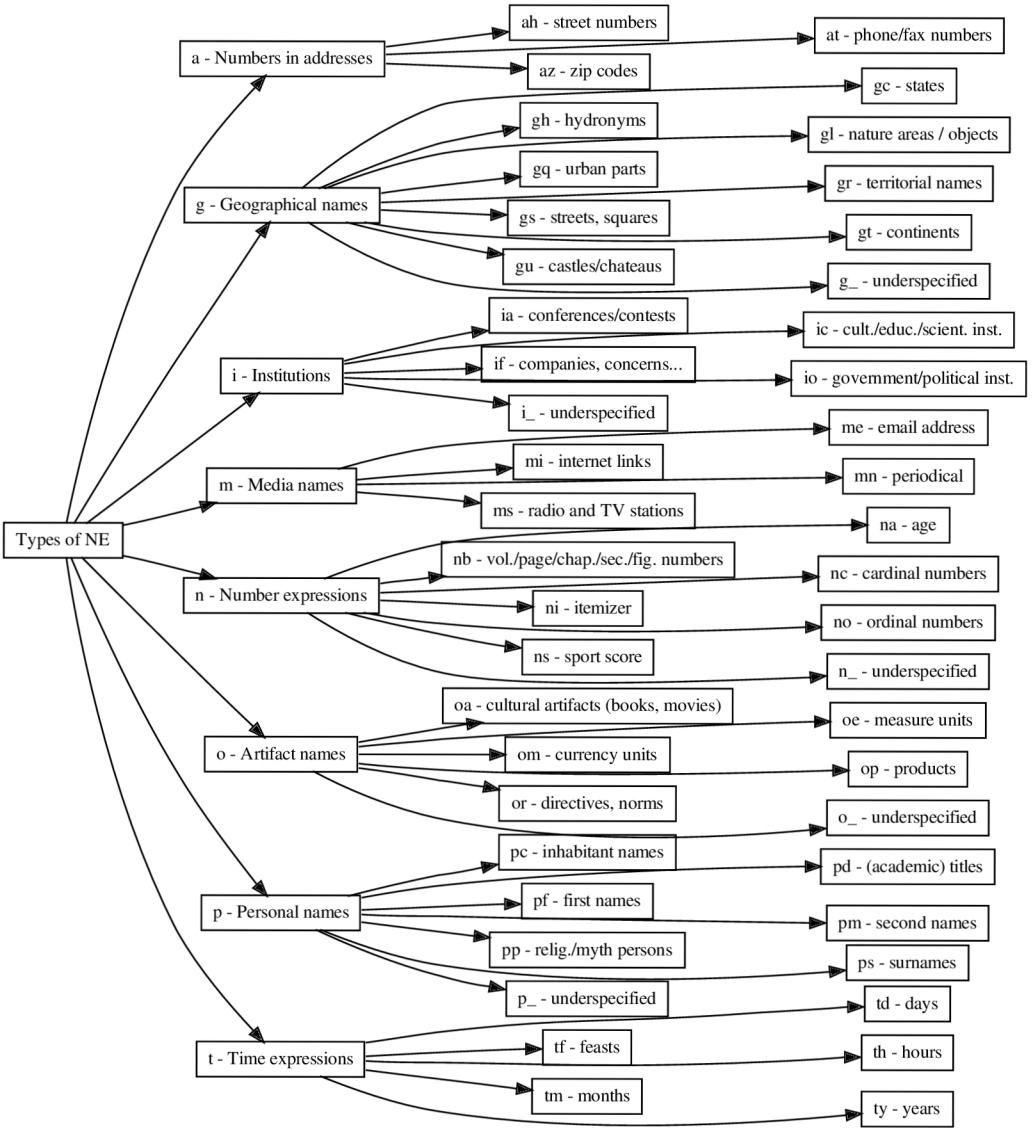


Figure 3.1: CNEC 2.0 classification hierarchy [47].

for Czech is 9 877. This dataset represents contemporary Czech news texts, so the benefit is the news style. However, the topics covered are far from historical news content, so usability of this dataset for this thesis is a question for implementation.

3.3.4 SumeCzech-NER

SumeCzech-NER is a recent (2021) large Czech NER dataset [34], constructed automatically from SumeCzech 1.0 [50]. SumeCzech 1.0 is a collection of 1 000 000 Czech news articles gathered from 5 Czech news websites: novinky.cz, lidovky.cz, denik.cz, idnes.cz and ceskenoviny.cz. SumeCzech-NER has total 7 named entity types: numbers in addresses, geographical names, institutions, media names, artifact names, personal names, and time expressions [34]. The dataset consists of NER annotations in IOB format for SumeCzech

1.0 dataset. This is a very large dataset a contemporary Czech news articles, and could be very beneficial for NER system.

Chapter 4

Named entity recognition system design

This chapter presents a proposed NER system for Czech historical documents. Based on the information in the previous chapters, a NER system for information extraction and necessary data to train such system is discussed.

First, challenges related to NER system are discussed. Second, a current state of systems for historical Czech information extraction is elaborated. Next, an input data for the NER system are described. Next, used datasets are discussed and new datasets are proposed. Last, the NER system architecture and training procedure are proposed.

4.1 Challenges

A suitable datasets are essential for many NLP tasks including Named Entity Recognition. While some datasets can be annotated automatically, many NLP tasks require manually annotated datasets in order to perform well. This also applies for NER. Manual creation of NER dataset is very time-consuming work and therefore limits the results of NER systems. In case of automatically annotated datasets, usually external annotations (e.g. Wikipedia) are used. Otherwise another NER system for named entity annotation is required, but this approach brings in annotation errors of used NER system. Example of such dataset is SumeCzech-NER [34]. However, there is still a benefit from a greater amount of data.

The digitization process of historical documents is not perfect and OCR errors appear. OCR errors include wrongly recognized character and not detected words. These errors will negatively affect the NER system; nevertheless OCR is not a part of this thesis.

Recent language models are relatively large, with the number of parameters usually exceeding 100M. To get the best possible results, state-of-the-art models are trained on very large datasets, e.g. BERT was trained on datasets that totaled 3 300M words and 16 GB of uncompressed text data; RoBERTa was trained on datasets with size of 160 GB [22][31]. These results are not reproducible for this thesis, as such the training would require many weeks or a high number state-of-the-art deep learning GPUs to reduce the training time. Therefore, a pre-trained models and distributed representations are needed. The training time though will still be a challenge; the only way to reduce the training time is to employ a powerful GPU training setup. In particular, original LUKE pre-training took 30 days on a server with 16 NVIDIA Tesla V100 GPUs [58]. The pre-training dataset consisted of whole English Wikipedia dump, which has approximately 3.5 billion words.

However, the Czech Wikipedia is much smaller [14] and only small subset will be necessary, so the training should not that tedious.

4.2 Current state

With the digitization of historical documents comes the need to analyze the document's content. To the best of my knowledge, no complex system or tool for Czech historical document information extraction exists. The task is relatively new and attracts attention of researchers in recent years, as previous research was focused on contemporary Czech documents. With the advancement of information extraction thanks to deep learning, solving the problem should become more achievable. There are only minor efforts to tackle this problem yet, notably CHNEC [27] and [26]. Both works present models for Named Entity Recognition. In CHNEC [27], the best Bi-LSTM model using static fastText achieved 73% F-score on CHNEC dataset. The best model in [26], a Slavic BERT [17] fine-tuned on mix of CHNEC and CNEC achieved 82% macro-averaged F-score. However, these results are not sufficient for real-world applications. The main drawback in historical Czech NER is the lack of datasets, respectively the small size of the only Czech historical dataset – CHNEC.

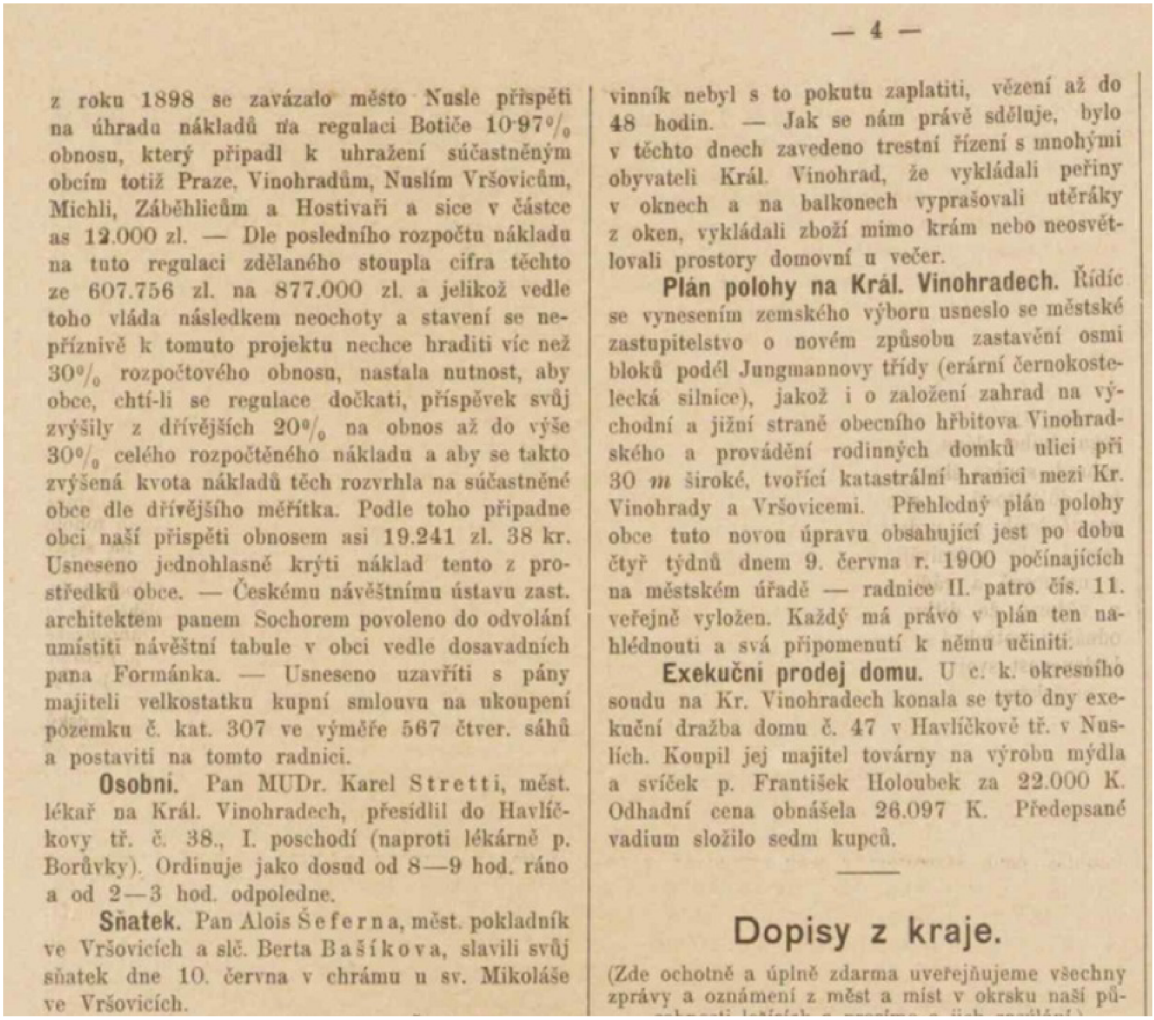
Situation for contemporary Czech information extraction is better. There are vast amounts of contemporary Czech texts available [15], and datasets for NER, such as CNEC [47], BSNLP 2019 [40] and SumeCzech-NER [34]. Several works for NER were published, e.g. [51], [52], [17], [49]. During research, two relevant Czech NLP research groups were found: *Institute of Formal and Applied Linguistics* [5] at Charles University and *NLP group* [9] at University of West Bohemia. The two research groups produced most Czech related works.

The goal of this thesis is to push the results forward and develop a NER system for Czech historical documents. The input data of this system are described in next section.

4.3 Input data

This thesis aims to apply Named Entity Recognition to Czech historical documents of three types: newspapers 4.1, chronicles 4.2 and registry books 4.3. The text style can be classified as journalistic (newspapers, chronicles), communication (newspapers, chronicles), and administrative (registry books). The style of chronicles and registry books is presumed to be more formal with fewer phrases and more time and location expressions. Chronicles contain data about events. Sources of chronicles may be municipalities, parishes, companies, schools, associations, and others. Registry books state data about birth, death, marriage such as involved people, relatives, locations; and people profession. Newspapers are printed texts, while chronicles and registry books are mostly handwritten. Handwritten text is problematic due to lower quality of OCR conversion. The handwriting quality differs a lot; some texts are easily readable, while others may be unreadable even for human.

Input data will be downloaded from public sources, such as archives and libraries. For example: *Státní oblastní archiv v Třeboni* [2], *Státní oblastní archiv v Plzni* [11] and *Zemský archiv v Opavě* [3]. A representative dataset of Czech historical documents will be created. Documents will be digitized by PERO-OCR [28] application, a part of PERO project. Figure 4.4 shows a screenshot of PERO-OCR application with chronicle example and its converted text.



— 4 —

z roku 1898 se zavázalo město Nusle přispěti na úhradu nákladů na regulaci Botiče 10·97% obnosu, který připadl k uhražení súčastněným obcím totiž Praze, Vinohradům, Nuslím Vršovicům, Michli, Záběhlicům a Hostivaři a sice v částce as 12.000 zl. — Dle posledního rozpočtu nákladu na tuto regulaci zdělaného stoupla cifra těchto ze 607.756 zl. na 877.000 zl. a jelikož vedle toho vláda následkem neochoty a stavení se nepříznivě k tomuto projektu nechce hraditi víc než 30% rozpočtového obnosu, nastala nutnost, aby obce, chtí-li se regulace dočkati, příspěvek svůj zvýšily z dřívějších 20% na obnos až do výše 30% celého rozpočtěného nákladu a aby se takto zvýšená kvota nákladů těch rozvrhla na súčastněné obce dle dřívějšího měřítka. Podle toho připadne obci naší přispěti obnosem asi 19.241 zl. 38 kr. Usneseno jednohlasně krýti náklad tento z prostředků obce. — Českému návěštnímu ústavu zast. architektem panem Sochorem povoleno do odvolání umístiti návěštní tabule v obci vedle dosavadních pana Formánka. — Usneseno uzavříti s pány majiteli velkostatku kupní smlouvu na ukoupení pozemku č. kat. 307 ve výměře 567 čtver. sáhů a postaviti na tomto radnici.

Osobní. Pan MUDr. Karel Stretti, měst. lékař na Král. Vinohradech, přesídlil do Havlíčkovy tř. č. 38., I. poschodí (naproti lékárně p. Borůvky). Ordinuje jako dosud od 8—9 hod. ráno a od 2—3 hod. odpoledne.

Sňatek. Pan Alois Šeferna, měst. pokladník ve Vršovicích a slč. Berta Bašíkova, slavili svůj sňatek dne 10. června v chrámu u sv. Mikoláše ve Vršovicích.

vinník nebyl s to pokutu zaplatiti, vězení až do 48 hodin. — Jak se nám právě sděluje, bylo v těchto dnech zavedeno trestní řízení s mnohými obyvateli Král. Vinohrad, že vykládali peřiny v oknech a na balkonech vyprašovali utěráky z oken, vykládali zboží mimo krám nebo neosvětlovali prostory domovní u večer.

Plán polohy na Král. Vinohradech. Řídíc se vynesením zemského výboru usneslo se městské zastupitelstvo o novém způsobu zastavění osmi bloků podél Jungmannovy třídy (erární černokostelecká silnice), jakož i o založení zahrad na východní a jižní straně obecního hřbitova Vinohradského a provádění rodinných domků ulici při 30 *m* široké, tvořící katastrální hranici mezi Kr. Vinohrady a Vršovicemi. Přehledný plán polohy obce tuto novou úpravu obsahující jest po dobu čtyř týdnů dnem 9. června r. 1900 počínajících na městském úřadě — radnice II. patro čís. 11. veřejně vyložen. Každý má právo v plán ten nahlédnouti a svá připomenutí k němu učiniti.

Exekuční prodej domu. U c. k. okresního soudu na Kr. Vinohradech konala se tyto dny exekuční dražba domu č. 47 v Havlíčkově tř. v Nuslích. Koupil jej majitel továrny na výrobu mýdla a svíček p. František Holoubek za 22.000 K. Odhadní cena obnášela 26.097 K. Předepsané vadium složilo sedm kupců.

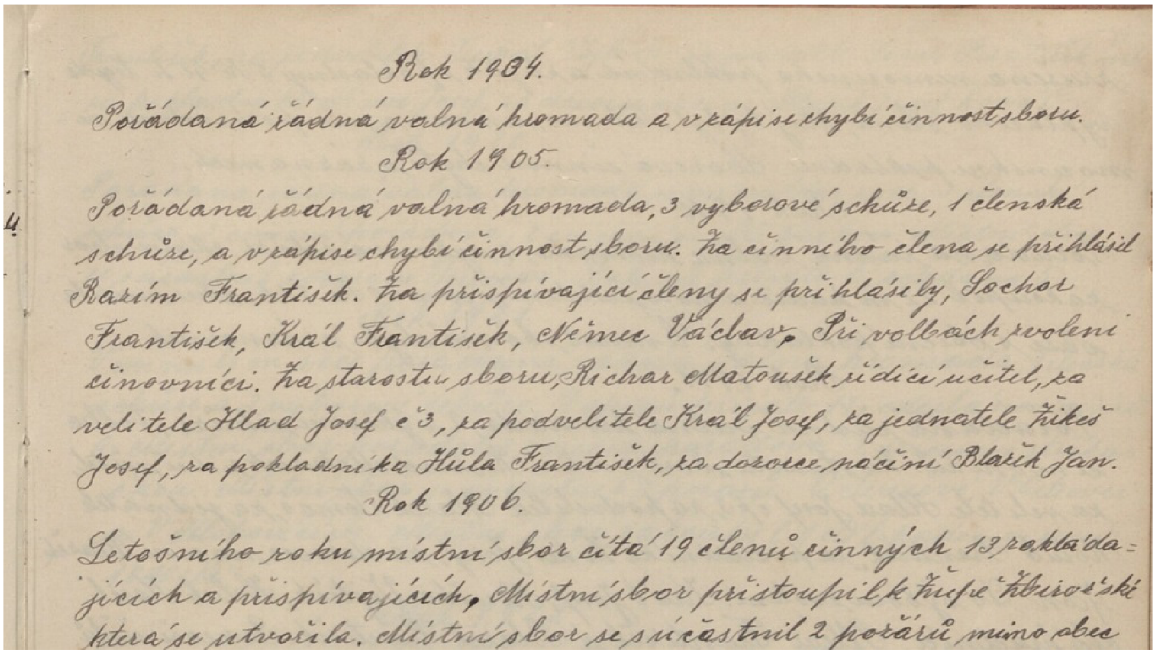
Dopisy z kraje.

(Zde ochotně a úplně zdarma uveřejňujeme všechny zprávy a oznámení z měst a míst v okrsku naší pů-

Figure 4.1: Example of newspaper – printed text [2].

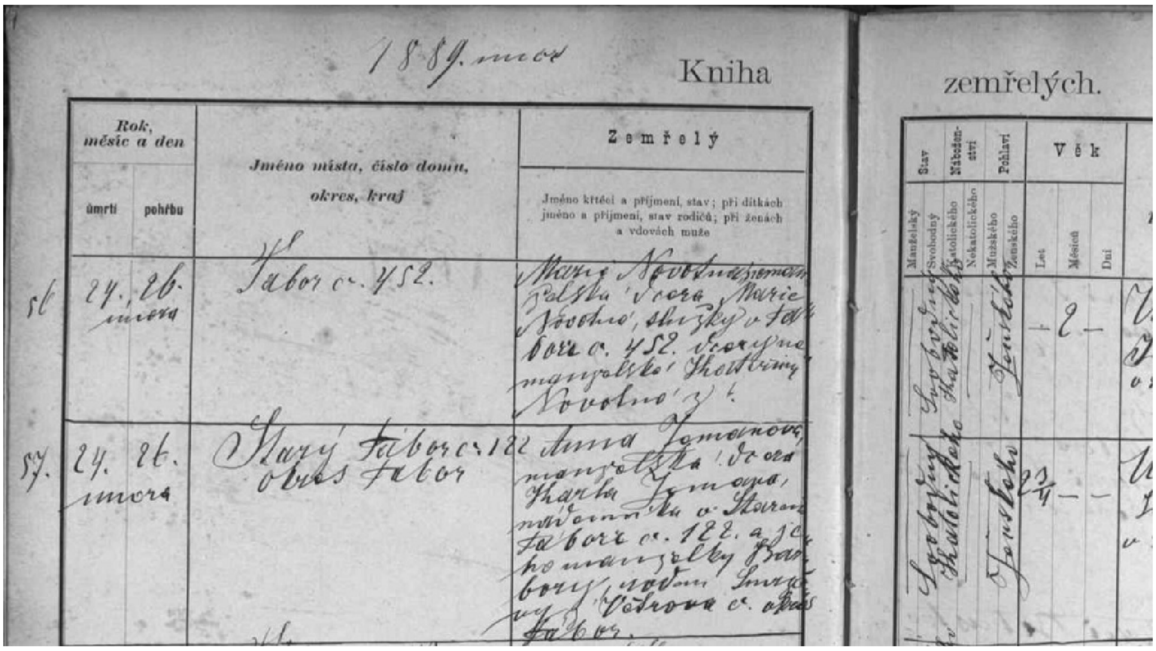
4.4 Used datasets

The baseline dataset for NER system training and evaluation will be the combination of CNEC 2.0 and CHNEC 1.0, as was used in [26]. However, it is clear that the amount of data in these datasets will not be sufficient, as state-of-the-art systems require a large NER dataset. Therefore used dataset needs to be extended and adjusted data domain in further steps. BSNLP 2019 and SumeCzech-NER datasets could be leveraged after some processing into a common format. As datasets described in chapter 3 are in different formats, have different named entity types and their size varies significantly; a suitable approach for mixing the datasets needs to be taken. The coarse-grained classification of entities is sufficient for this task goal, same as in the case of [27]. In the present Czech datasets, there is different naming of entity types and some entity types are irrelevant. New dataset will follow CHNEC entity type definition. CHNEC entity types are: *personal names, institutions, geographical names, time expressions and artifact names / objects.* The annotations format will be CoNLL, same as in CHNEC [27]. In this manner, compatibility comparability with CHNEC will be ensured. In the case of CNEC 2.0 [47], its CoNLL version [29] will be utilized.



Rok 1904.
Pořádaná řádná valná hromada a v zápise chybí činnost sboru.
Rok 1905.
Pořádaná řádná valná hromada, 3 výborové schůze, 1 členská schůze, a v zápise chybí činnost sboru. Za činného člena se přihlásil Bařím František. Za přispívající členy se přihlásily, Sochor František, Král František, Němec Václav. Při volbách zvoleni činovníci. Za starostu sboru, Richar Matoušek řídící učitel, za velitele Hlad Josef č 3, za podvelitele Král Josef, za jednatele Zikeš Josef, za pokladníka Hůla František, za dorosce náčiní Blažek Jan.
Rok 1906.
Letošního roku místní sbor čítá 19 členů činných 13 zakláda-jících a přispívajících, Místní sbor přistoupil k župě Zbirožské která se utvořila. Místní sbor se súčastnil 2 požárů mimo obec

Figure 4.2: Example of chronicle – good readability [11].



Kniha zemřelých.

Rok, měsíc a den		Jméno místa, číslo domu, okres, kraj	Zemřelý
úmrtí	pohřbu		Jméno křtící a příjmení, stav; při dítkách jméno a příjmení, stav rodičů; při ženách a vdovách muže

Stav		Náboženství		Pohlaví		Věk		
Manželský	Svobodný	Katolického	Nekatolického	Mužského	Ženského	Let	Měsíců	Dní

Figure 4.3: Example of registry book – worse readability [2].

Data will be annotated by suitable tools, such as Inforex [33] or Label Studio [54]. Figure 4.5 shows Inforex annotation tool. Annotation of data will probably be very time consuming. Manually annotated dataset will be important for system testing. With a dataset created by steps described above, good results could be achieved.

Figure 4.4: Screenshot of PERO-OCR application with a chronicle example [**?**][11].

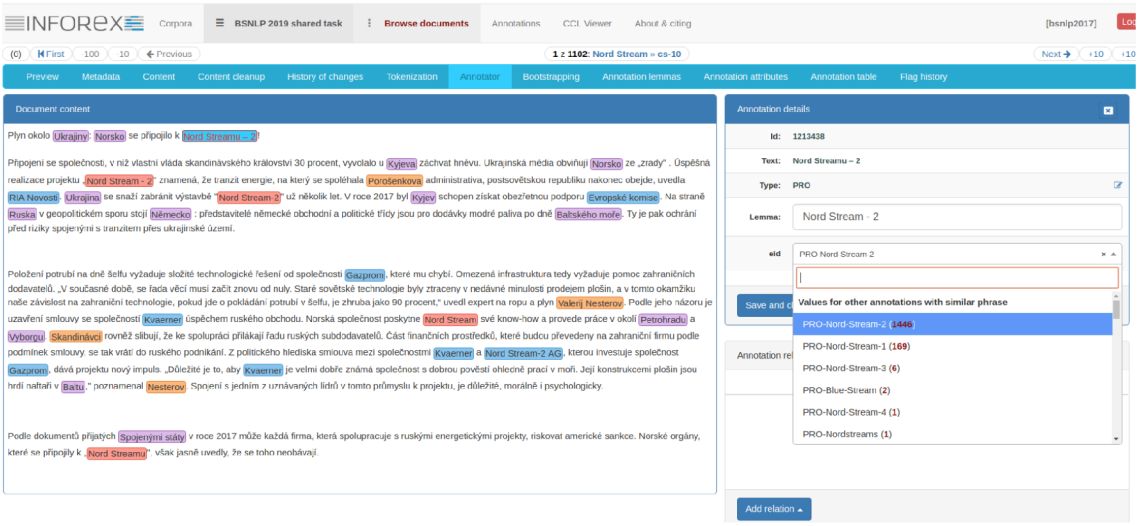


Figure 4.5: Screenshot of Inforex Web annotation tool [33] (image taken from [40]).

4.5 Model architecture and training

A neural language model will be used as the basis of the NER system. RobeCzech [51] model will be adopted together with Slavic-BERT [17] and Czert [49]. For detailed description, see 2.3.3 and 2.2.2, respectively. LUKE represents a new approach for entity-related tasks. RobeCzech, a Czech variant of RoBERTa [31], is a state-of-the-art Czech general-purpose language model. Fine-tuning approach will be primarily used for NER training.

The baseline model will be RobeCzech language model [1] fine-tuned on CNEC 2.0 and CHNEC datasets (more in previous section). Own language models based on RoBERTa architecture will be trained. To meet the performance demands and to be able to actually train the model in a reasonable time, smaller model variants will be proposed and trained. Reruns and hyper-parameter tuning will most likely not be possible. Pre-trained own models will then be fine-tuned on CNEC 2.0 and CHNEC datasets, as well as on own NER dataset. SumeCzech-NER can be added for further experiments. The model performance will be evaluated and analyzed.

Python is selected as implementation language, as it is most suitable language for machine learning. PyTorch [37], a Python machine learning framework, will be used for model training and evaluation. For data handling, tokenizers, parallelization and more Hugging Face [4] libraries will be used. Data handling will probably require scripts written in Python.

[1] Note that RobeCzech paper RobeCzech present both language model and a special LSTM-CRF model for flat NER, where RoBERTa embeddings are used as inputs.

Chapter 5

Implementation

This chapter presents details of Named Entity Recognition system implementation, which was proposed in previous chapter. An emphasis was put on possibility to experiment and further improvement of the system, along with development of PERO OCR [28]. The goal is to provide a system with best results possible, which can be improved using new datasets. Eventually, such system can be tuned for performance with techniques such as knowledge distillation [24] [44] and neural network pruning and then used en masse on large historical document databases.

Python, which is regarded as most suitable language for machine learning, was selected as an implementation language. The models are themselves trained with PyTorch [37] deep learning library, which is used by Hugging Face [4] Transformers library. Other libraries from Hugging Face platform, such as Accelerate, Datasets, Evaluate and Tokenizers are also employed. Hugging Face platform was chosen as the best tool for Natural Language Processing (NLP) currently available. It facilitates most common functions in NLP. Implemented system is in the form of Python scripts using above mentioned libraries. These scripts produce new neural network models. Results of these models are described in the next chapter.

Beside scripts, datasets are needed in order to train new models. Several datasets were used and created. New language models were also trained. First, Named Entity Recognition scripts are discussed, and then datasets and their creation are described, followed by Masked Language Model training.

5.1 Named Entity Recognition models

Experimenting with Named Entity Recognition models involves data management, training and evaluation of models. In this section, most important aspects of implementation are discussed.

5.1.1 Model training script

Following a design in previous chapter, a Named Entity Recognition model needs to be trained. For this purpose `train_ner_model.py` was created. This script fine-tunes a Hugging Face language model on NER datasets, evaluates the model on NER datasets test splits and reports the results.

Script takes 2 required arguments: `--results_csv` for a CSV file for logging the test results and `--config` for a YAML file training configuration. All training settings are

configurable, this way the training script can be used to fine-tune any Hugging Face language model on any NER dataset in the Hugging Face dataset format. Training hyperparameters are set in this configuration too. The configuration file represents one training experiment, therefore an experiment can be easily reproduced and repeated on any machine with Python, PyTorch and Hugging Face libraries installed. Experiments can be evaluated and modified with minimal effort.

The YAML configuration file allows to specify: language model, with its name, description and path; multiple NER datasets with their names, descriptions and paths; training hyperparameters such as type of learning rate scheduler, batch size, number of training epochs, learning rate, weight decay, warmup rate, etc. The YAML format provides easily readable and synoptic way to manage model training.

After loading the configuration file, a summary of experiment with basic information is logged into the `experiment_results.txt`. This enables to recognize which results belongs to the experiment, because multiple experiments are run subsequently, as will be described below. Then all necessary objects like tokenizers, data loaders, optimizer and learning rate scheduler are initialized according to specification from configuration file. Datasets in Hugging Face format are loaded. Input dataset has to have the same entity types as CHNEC 1.0 [27]. The CHNEC entity types are: *Personal names*, *Institutions*, *Geographical names*, *Time expressions* and *Artifact names / Objects*. Each dataset has a train, validation and test split. Train and validation splits from all specified datasets are tokenized by model tokenizer and concatenated. In order to fine-tune a language model on NER task, a classification head is put on top of language model. This head consist of 1 linear layer with softmax. After that model is trained in a training loop for specified number of epochs and evaluated on validation split after each epoch. Training can be run on multiple GPUs, thanks to parallel data processing. Hugging Face accelerate library is used for parallelization.

Evaluation is performed on true labels, e.g. softmax outputs of the last classification layer are converted into IOB format, where each token in a sequence has a label. A *seqeval* metric is employed for evaluation. Seqeval is a common metric for NER task. It calculates F1, accuracy, precision and recall. During training, loss, learning rate and metric values are logged into TensorBoard logger. These logs can be viewed in TensorBoard tool. When training is finished, trained model is evaluated on every specified dataset's test split. Results are then logged into text log, as well as CSV log. If multiple experiments are run together, with `--results_csv` argument one CSV file can be used repeatedly. In this case, the results CSV file is created only once and every following experiment only appends its results at the end of file. Script skeleton was taken over from Hugging Face Course Chapter 7 *Token classification* [12].

5.1.2 Cloud computation

Because training of transformers language models requires great computing power, computing on external server was necessary. The MetaCentrum[7], a virtual organization providing cloud computing services, was selected for its availability and free access for Czech academia. The MetaCentrum uses PBS Pro scheduling system. Computational tasks are launched in a form of the so-called job, an entity consisting of computational resources and a sequence of commands written in a UNIX shell script. A script for running the MetaCentrum model training job was implemented. It clones this thesis git repository, downloads datasets and models and prepares software environment. All required libraries are installed. Then it

runs the `train_ner_model.py` script. One, several or all configurations can be run. Script runs the Python training script and then saves experiment results to home storage. If more than one experiment is run, these two steps are repeated in a loop.

5.1.3 Dataset preparation

Training script `train_ner_model.py` accepts datasets in Hugging Face format. All data processing is handled by Hugging Face Datasets library, which provides a lot of functionality. Datasets internally use Apache Arrow library for data storage, which enables fast processing of large amounts of data, without loading the data into memory due to memory-mapping. One dataset training example is a dictionary with 3 keys: `id`, `tokens` and `ner_tags`. Tokens is a list of words and `ner_tags` is a list of labels. Labels are represented as integers. These integer labels are mapped to IOB labels during evaluation. A training example is shown in Figure 5.1.

```
{'id':  '4138',
'tokens':  ['Přednášel', 'Frant', '.', 'Pruský', 'z', 'Olomouce', '.'],
'ner_tags':  [0, 1, 2, 2, 0, 5, 0]}
```

Figure 5.1: One training example from PONER dataset.

The used CNEC, CHNEC, SumeCzech-NER and PERO OCR NER (PONER) NER datasets, are in different format and need to be prepared. This includes dataset loading scripts for each dataset, transform data into above mentioned format and saving them into Hugging Face datasets format, so the datasets are ready for training. CNEC, CHNEC and PONER datasets are saved in CoNLL format. Annotated texts are split into linguistic sentences (or just block of text in some cases). Each sentence is split into words. Every word is on separate line, together with its annotation labels separated by space. For example in CNEC, the labels are *lemmas*, *morphological tags* and *ner tags*. This way one dataset can be used for different NLP tasks. Sentences are separated by an empty line. Loading script read the CoNLL format files into Hugging Face dataset format. Transformation of data consists of removing unnecessary data columns and changing NER tags entity types to the same as in CHNEC. Same steps need to be performed for every newly added NER dataset.

5.2 Datasets

Datasets are integral part of any machine learning effort. Here, detailed description of created datasets is presented. For each dataset, its purpose, creation process, data description and other details can be found bellow. Additionally, preparation of SumeCzech-NER is also described.

5.2.1 PERO OCR NER

New Named Entity Recognition dataset was created. Since the resulting NER system is intended to operate on historical documents processed by PERO OCR [28] system, a dataset created from historical and OCR-sourced data is needed. For this reason, the new dataset is named PERO OCR NER (PONER).

Source data

Document pages from chronicles are the source data, because chronicles are rich in named entities placed in various contexts. PONER consists of 400 document pages, from which 250 are from rural and 150 from urban chronicles. Out of the 250 rural chronicles, 180 are from *central Moravia*, especially from *Šumperk* and *Přerov regions*. The rest 70 are from *České Budějovice region*. The urban chronicles are from *Přerov* city chronicle. This data distribution is important in order to include text from various environments, dialects (e.g. not only central Moravian dialect), places and language styles. Scans of document page were first processed by PERO OCR, and the resulting text transcriptions were annotated. The quality of OCR transcriptions is very high, errors occur very rarely. The document pages are mostly from first half of 20th century, less from post-war period. The oldest document page is from year 1771 describing last widespread famine in Czech kingdom, the most recent document page is from year 1993 describing introduction of telephone connection. Rural chronicles often describes village inhabitants, agricultural works like harvests, prices of agricultural commodities, weather and cultural events. Unlike rural chronicles, urban chronicles describe industrial information, city development, local politics and much more cultural events. Nationwide or even international events (e.g. *Munich Agreement*) occur in both types. Pages from post-war period are ideologically affected, collectivization of Czech countryside is a major theme.

Dataset creation

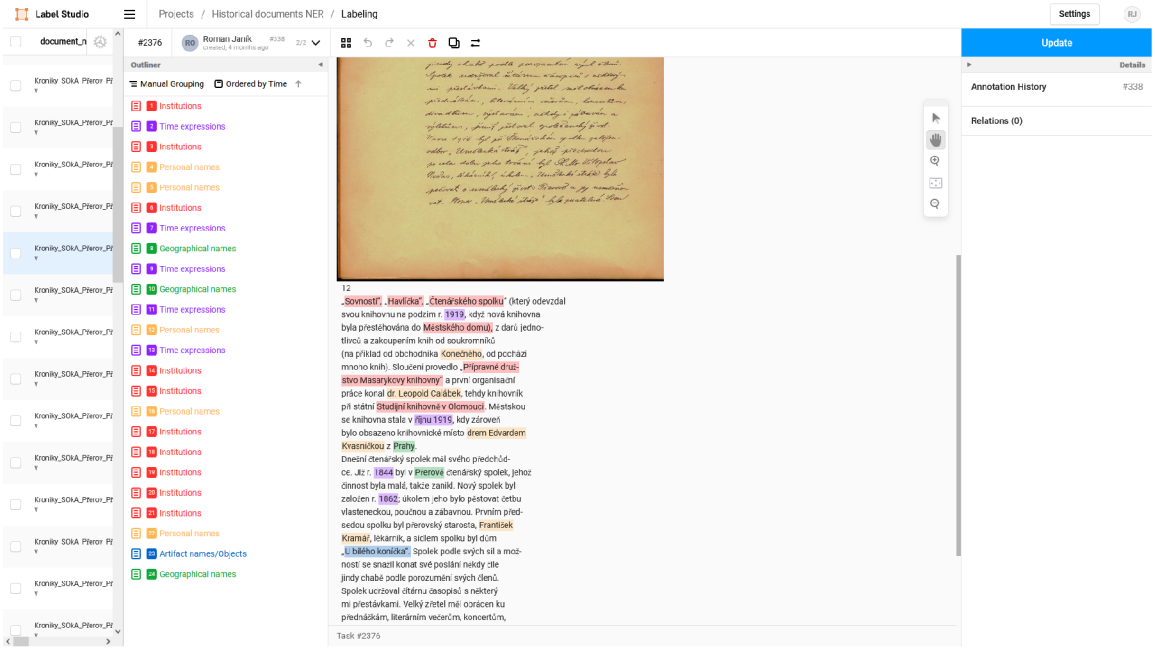


Figure 5.2: An annotation task in Label Studio application.

Data were annotated in Label Studio [54] tool. Label Studio is a web server application, which is run locally. Data are annotated in a provided webpage. The tool is organized in a project manner, project has its settings defining a labeling interface. Labeling interface sets GUI of annotation webpage, controls and annotation schema (here marking

text span by its class). An annotation item is called "Task", here one task is annotation of one document page. Tasks are defined in a JSON task file containing paths/urls to annotated files and other task information. Document page text transcription files are imported together with their respective JPEG images in order to provide a way to check the original data in case of OCR errors. JSON task file creation is implemented in script `create_label_studio_import_file.py`. A task webpage is shown in Figure 5.2.

The manual annotation is a lengthy process. To accelerate the annotation, tasks were pre-annotated by a trained NER model. The best model from first round of experiments was used (see Subsection 6.2.1). Pre-annotations can be considered as high-quality. Comparison of pre-annotations and final manual annotation is shown in Figure 5.3. The model often marked a phantom named entity in front of real one containing only newline character. Longer named entities were usually also marked wrongly. Some named entities are not marked by model at all, notably groups of people as *Personal names* (e.g. nationalities "*Němci*"). Nevertheless the annotation took several weeks of effort.

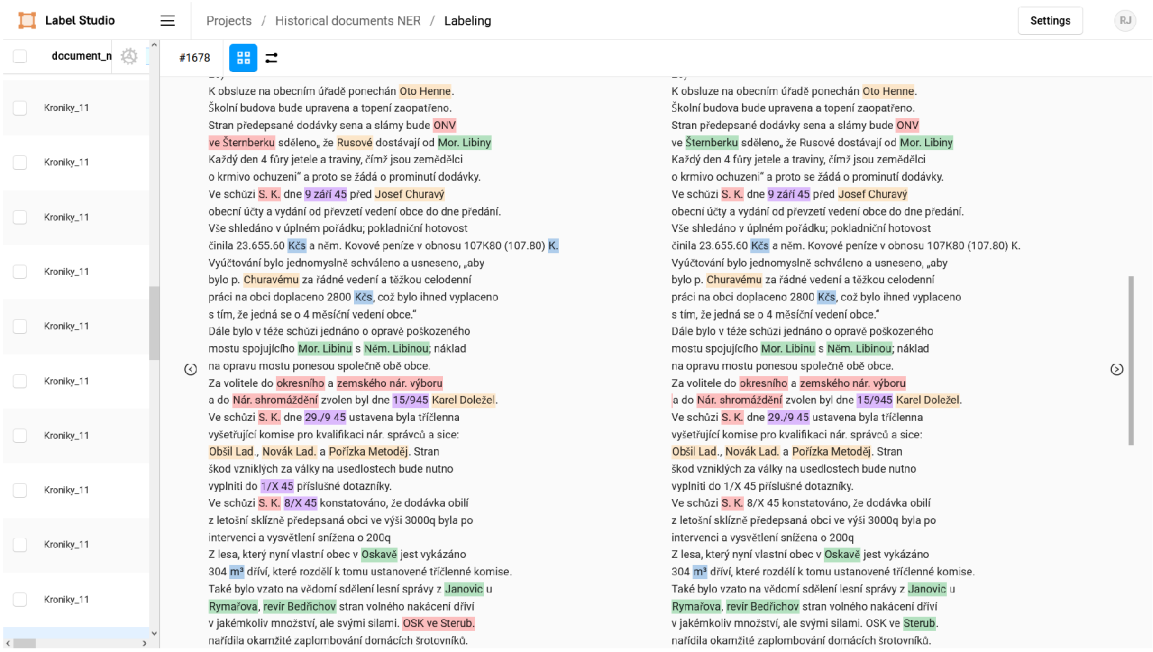


Figure 5.3: Comparison of pre-annotation by NER model (on the right) and final manual annotation (on the left).

As mentioned in Section 4.4, PONER dataset uses the same named entity types/classes as CHNEC. Named entities found in page text were marked with respect to annotation manuals for CNEC [47] and CHNEC [47] datasets that can be found in dataset's archives. However, these manuals have several conflicts listed below:

1. CNEC has animal names included in *Personal names*.
2. CNEC has a subclass "personal names of unspecified type / unclassifiable in other types" in *Personal names* with example "Slované". CHNEC has a subclass "designation of collectives" in *Intitutions* with examples "benediktini, husité, republikané, atd.".

3. CNEC have castles and palaces names in subclass "municipalities, castles and palaces" in *Geographical names*. CHNEC has a subclass "specific building objects" in *Artifact names / Objects* with examples "věž u svatých, kostel sv. Bartoloměje, zámek Kozel, klášter benediktinský u Davle, hrad Domažlický, atd.".

4. CNEC has a subclass "lectures, conferences, competitions, . . . " with example "Stanley Cup" in *Institutions*. CHNEC is missing such subclass.

5. CNEC have subclasses "measurement units (written is shortcuts)", "names of unspecified types / unclassifiable in other types", "regulations, standards,..., their collections" and "names of chemicals, chemical formulas" in *Artifact names / Objects*. CHNEC is missing such subclasses.

6. CHNEC have subclasses "names of historical events" with examples "bitvě na Bílé hoře, Pražská defenestrace , bitva u Slavkova, atd." and "names of official recurring events" with examples "Mezinárodní filmový festival Karlovy Vary, Všesokolský slet, atd." in *Time expressions*. CNEC is missing such subclasses.

7. CHNEC has a subclass "books, magazines, editions etc. printed matter" in *Artifact names / Objects*. CNEC has media class.

8. CHNEC has a subclass "currency names" in *Artifact names / Objects*. Both currency abbreviations with examples "Kč, zl, zl. r. č." and full names with examples "zlatých, tolar, krejcarů" are marked. In CNEC, only currency abbreviations are marked.

These conflicts were solved the following way: Since the media class / entity type is not used in the dataset or during training of NER models, CHNEC method is selected. Also for conflicts 6 and 8 CHNEC method is selected. For all other conflicts, CNEC method is selected.

Several other problems also occurred. Measurement units are sometimes put together with their values by OCR algorithm (e.g. "25ha", "4q"). However, Label Studio allows only marking of whole words. These entities were not marked. The "whole word only" policy also caused marking of special characters such as dots and commas at the end of entity, even though it is not a part of it (e.g. "Jan Novák."). In cases, where the decision if a text span is a named entity or not and what type it belongs, CNEC and CHNEC dataset were searched. Resulting annotation by Label Studio is a list of annotations consisting of: start character index, end character index and entity type. Annotations can be exported from Label Studio application in JSON file. Some of above mentioned shortcomings were filtered out. However, some could not be easily filtered and required writing a set of rules. Still, several thousand problematic named entities / text spans needed to be looked at manually. This lengthy process took several days. Basic filtering is implemented in script `remove_start_whitespace.py`, semi-automatic adjustment is implemented in `adjust_annotation_end.py`.

In order to be comparable to CNEC and CHNEC datasets, Label Studio JSON export file needed to be converted to CoNLL format. This conversion is implemented in `create_my_dataset_conll.py` script. In CoNLL format, text is split into sentences separated by and empty line and sentences are split into words, each on a separate line (see Subsection 5.1.3). Natural language toolkit (NLTK)[18] library was utilized for the task of sentence and word tokenization (the task of text splitting into sentences or words). NLTK provides a statistical model *Punkt*, supporting several languages including Czech. Although

the tokenization results are relatively good, Punkt model failed to tokenize more complicated sentences correctly. Most frequent errors originated in separators (dot, comma) inside of real sentence. This problem needed to be solved by manual correction of sentences in the CoNLL file. This process took several days. Since the dataset contains corrected sentences, it can be used for training a new model for sentence tokenization based on Transformers architecture.

The final dataset consist of 1.268 kB, 9,310 sentences and 14,639 named entities (numbers are taken from split files). Average number of named entities per document page is 36.5975. The dataset is split into three sub-sets: 45% for training, 5% for validation and 50% for testing. The testing split is 50%, because an extensive test evaluation is needed. Split ratio can be changed and dataset split again. Split statistics are described in Table 5.1. The distribution of named entity types is shown in Table 5.2.

Split	Sentences	Entities
train	4,189	6,641
validation	465	707
test	4,655	7,291
Total	9,310	14,639

Table 5.1: PONER split statistics.

Named entity type	Tag	Entities
Personal names	p	4,009
Institutions	i	2,901
Geographical names	g	2,964
Time expressions	t	2,720
Artifact names/Objects	o	2,045
Total		14,639

Table 5.2: PONER named entity type distribution.

Page crop tool

In the phase of collecting historical documents for the creation of PONER dataset a problematic documents were found. Since the real-life documents are scanned and then processed by PERO OCR, all text in the images is gathered. Unfortunately, some archives supply the scanned historical documents with additional header and footer, containing information like archive name, document name, date and copyright. This text is also in the output transcription of the OCR algorithm. Because historical data are not easily available, documents had to be reprocessed by PERO OCR without the header and footer. The possibility of defining a static rectangular region and the crop the images proved to be false, since images vary in resolution dramatically even though they come from the same historical document. An example of historical document with header and footer is shown in Figure 5.4.

To solve this problem, a machine learning approach was used. A semantic segmentation neural network model was trained. For training the model a suitable dataset was created. The Page segmentation dataset consist of 50 images of historical document pages with header and footer annotated in Label Studio label-studio tool. The number of images

is sufficient, since the dataset was used to fine-tune a small variant of SegFormer [57], a pretrained image model nvidia_mit-b0 [8]. The model consists of a hierarchical Transformer encoder and decoder head is attached during fine-tuning. It has only 3.7M parameters. The model was trained for 100 epochs with learning rate 6.e-5 and batch size of 16 examples. Additional data augmentation was applied: brightness, contrast, saturation and hue jitter. The resulting model achieved 0.9822 mean Intersection over Union (IoU), while for actual page class the model achieved 0.9960 IoU.

Page crop tool is implemented in script `page_crop.py`. Tool takes 2 arguments for specification of source and output directories. It iterates over all JPEG images in source directory, each image is processed by nvidia_mit-b0 model to get a segmentation mask. Mask is used for crop rectangle calculation. Image is then cropped and saved to output directory.

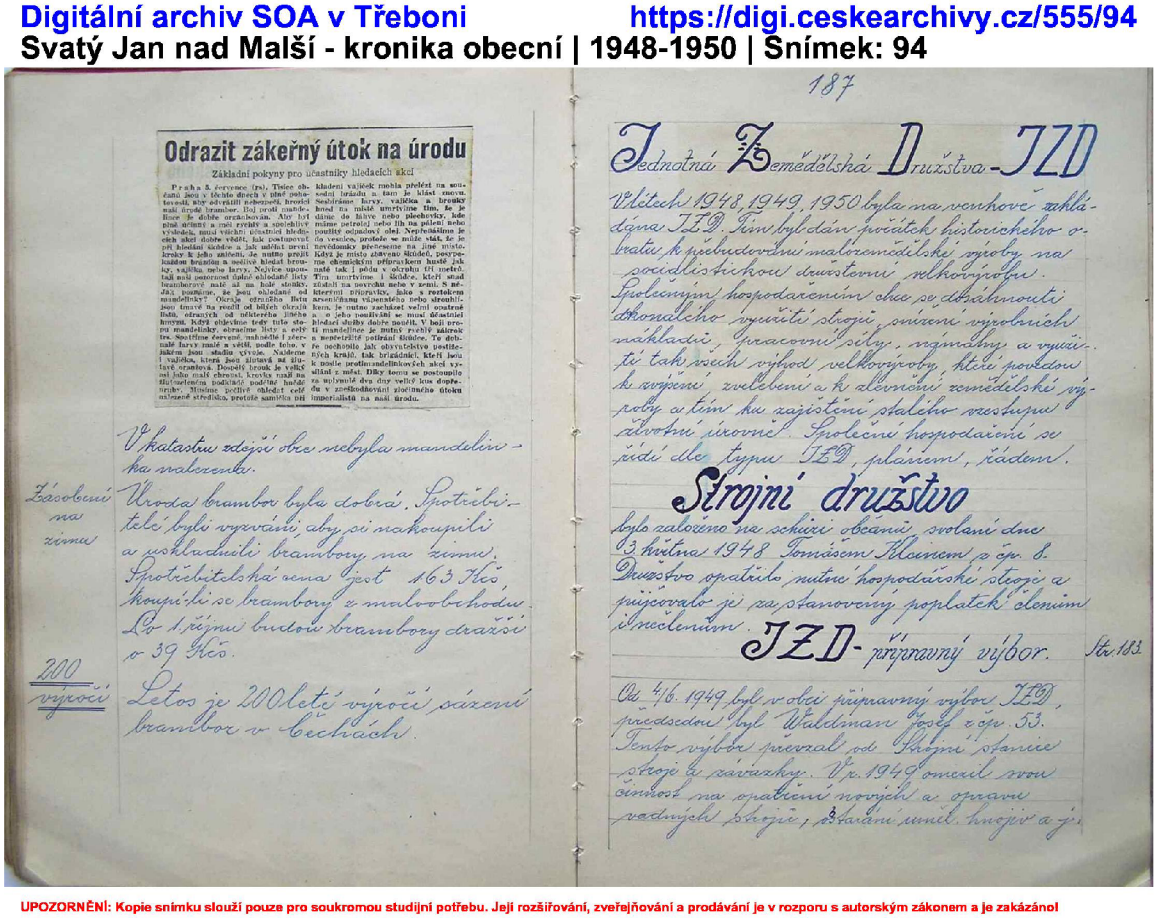


Figure 5.4: Example of historical document page with header and footer [2].

5.2.2 Masked language datasets

For training a language model from scratch a suitable textual dataset is needed. The data needs to be close to target domain, here historical documents like chronicles, newspapers, magazines, books and registry books. Also, same language across data needs to be ensured.

Three datasets were created: textual OCR-sourced PERO OCR Books and PERO OCR Periodicals, and their tokenized combination PERO OCR MLM.

PERO OCR Books and PERO OCR Periodicals

As source data, books and periodicals (newspapers, magazines) from the first half of 20th century were used. Data were processed by PERO OCR [28] algorithm. PERO OCR outputs transcriptions in XML format and actual text is then extracted from the XML files by script `extract_transcriptions_from_page_xml.py` from PERO OCR project. XML transcriptions are structured by OCR regions. OCR regions are regions containing text, they are detected in the first step of OCR pipeline. Usually they overlap with linguistic paragraphs in document pages. In these OCR regions, text is processed line by line. Every line object in XML transcription has a confidence of OCR algorithm. To prevent the model from learning OCR-errors, the extraction script was edited to extract only regions with average line confidence above 65%. This threshold was selected empirically. Document pages with $\leq$ 8 lines were filtered out, since they mostly contain nonsense characters. Periodicals data were almost completely in Czech language, some parts of other languages were removed manually. Books data contained a lot of various languages beside Czech. Since the number of books documents was close to 22,000, an automatic way of filtering these documents was necessary. Extraction script was edited to use a language classification model on first 200 characters of OCR region's text. To accelerate the processing, only first 5 regions were considered. If ≤ 2 regions were classified as Czech out of the first 5 regions, the document page was not extracted. As a language classification model, a fine-tuned XLM-RoBERTa [6] model was chosen. XLM-RoBERTa [21] itself is a multilingual version of RoBERTa. Filtered document pages were subsequently converted to Hugging face text dataset format.

Dataset statistics are displayed in Table 5.3. An important value is the total number of words. This dataset is relatively small, in order to be able to train a language model in a reasonable time. For example, RobeCzech [51] was trained on 4,917M words – a 17.9 times more.

Dataset	Number of words	Size in MB
PERO OCR Books	131.89M	846,50
PERO OCR Periodicals	142.65M	932.91
PERO OCR Books + Periodicals	274.54M	1,779.41
*datasets for RobeCzech training	4,917M	-

Table 5.3: Sizes of PERO OCR Books and Periodicals text datasets.

PERO OCR MLM

Language model training requires vast amounts of text data. Text data need to be converted to a numerical representation – in a process of tokenization. This can be done at the beginning of a training job, but since this process requires some time and is repeated every training, it is reasonable to perform it only once in advance. PERO OCR MLM consist of combined text datasets PERO OCR Books and PERO OCR Periodicals. The datasets were concatenated, tokenized and split by chunks of predefined length. The chunk length equals to maximal input length of a language model. It was set to 512 tokens, same as in

RoBERTa and RobeCzech. Resulting dataset was split into train and test splits in ratio 0.995:0.005. This ratio was chosen according to other works.

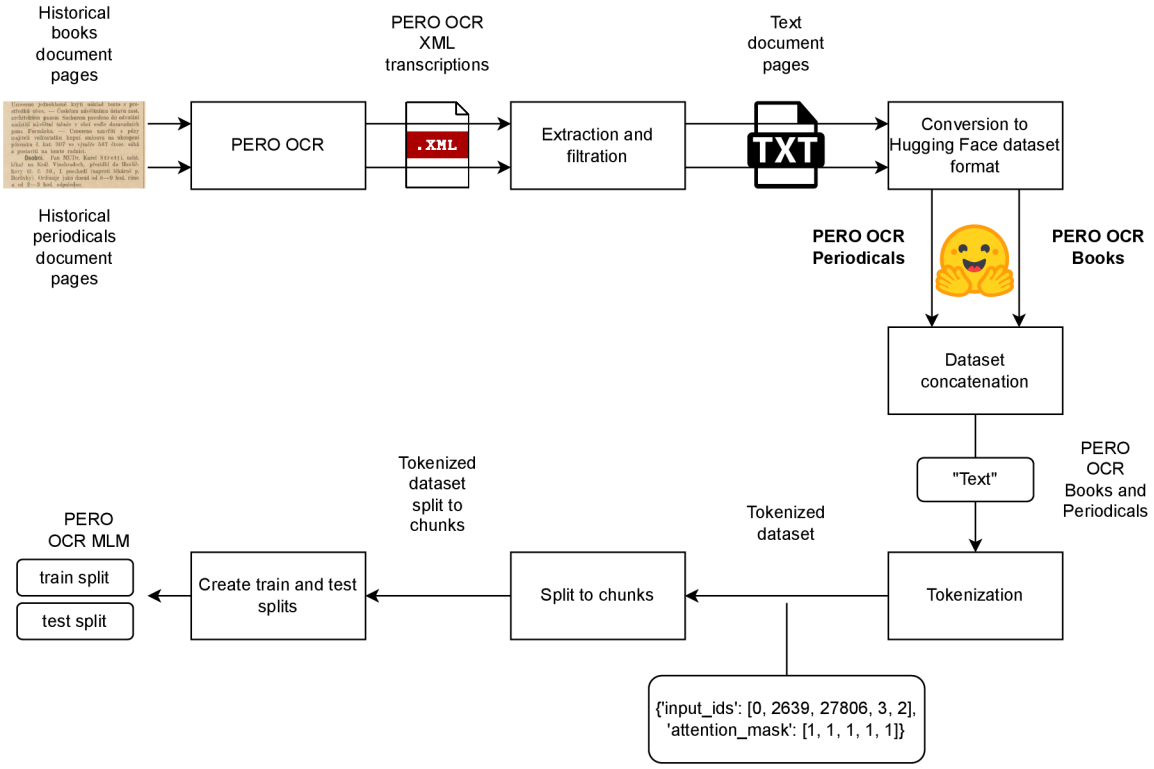


Figure 5.5: PERO OCR MLM creation steps.

In order to create more experiments for Masked Language Modeling (Section 6.1), 4 variants of the dataset defined by tokenizer vocabulary size were created. For this purpose 3 new tokenizers were trained on train splits of PERO OCR Books and PERO OCR Periodicals datasets. For comparison, the fourth tokenizer was taken from RobeCzech model. Tokenizer training is implemented in `train_tokenizer.py`. The tokenizers are byte-level [41] Byte-Pair Encoding (BPE) [45] type. Tokenizers Hugging Face library was utilized. The size of tokenizer vocabulary directly affects the parameter size of language model. Model with bigger (up to some point) vocabulary tend to achieve better language understanding and thus better results on downstream tasks. The tokenizers were trained with following vocabulary sizes: 52k, 26k and 12k. The original tokenizer from RobeCzech has 52k vocabulary. The final dataset is in Hugging Face format and prepared for language model training. The creation steps of PERO OCR MLM are depicted in Figure 5.5. Datasets statistics are summarized in Table 5.4. As can be seen in the table, with the increase of vocabulary size, the total number of examples decreases, as tokenizer with bigger vocabulary splits the text data into smaller subwords. This rule does not apply to old original tokenizer from RobeCzech model, since this tokenizer was trained on different text dataset.

Dataset variant	Vocabulary size	Total examples	Train split	Val split
new_tokenizer_12k_dts	12,000	1,090,924	1,085,469	5,455
new_tokenizer_26k_dts	26,000	994,275	989,303	4,972
new_tokenizer_dts	52,000	925,753	921,124	4,629
old_tokenizer_dts	51,961	1,021,431	1,016,323	5,108

Table 5.4: Sizes of variants of PERO OCR MLM dataset.

5.3 Masked language models

Masked language model training is implemented in `train_ml_model.py`. This script allows training a Hugging Face language model from scratch on a Masked Language Model task. The model learns to predict masked tokens in an input sequence. By this way a language representation is learned. The script takes a required argument `--config` for a YAML file training configuration. Training settings are configurable, similarly to Named Entity Recognition training script (Subsection 5.1.1). The configuration file represents one training experiment, allowing for easy reproduction, editing and repeating of an experiment. In the configuration file, a tokenizer (Subsection 5.2.2), a model, datasets and training hyperparameters are specified. Any RoBERTa model can be specified by a model configuration JSON file, allowing setting properties like hidden size, number of attention heads, number of hidden layers, maximum position embeddings and vocabulary size. With small changes in the script, other model architectures could be accepted as well. For a selected tokenizer out of 4 prepared tokenizers, related dataset is used. The datasets are loaded and ready for training. Training hyperparameters are configurable: type of learning rate scheduler, batch size, number of training epochs, learning rate, weight decay, warmup rate, etc.

All training logs are logged into `experiment_results.txt` file. Short summary is logged with basic information like experiment name, model, start time and training hyperparameters. After that, necessary objects are loaded, for example tokenizer, dataset, model and data collator. The dataset is ready for training; it is already tokenized and split to chunks of 512 tokens. An example is shown in Figure 5.6. In Masked Language Model task, tokens are masked, therefore labels are taken from the original text. Following RoBERTa [31] training, tokens are masked dynamically – every time an example is fed to the model. A 15% of tokens are masked, the same masking strategy as in RoBERTa is employed. The token masking is performed by the data collator object. The training is performed for a specified number of epochs. The metric for model evaluation is the Perplexity metric. Perplexity is computed as exponentiation of evaluation loss. Additional logging of train loss, learning rate, evaluation loss and perplexity to TensorBoard is performed during training. In order to prevent a gradient explosion, a gradient norm clipping is used with value of 8.0.

```
{'input_ids':  [885, 14, 4211, 440, 539, 489, 2059, 15, 142, 951, 5184,
12764, 413, 196, 440, 16224, 24965, 7611, 142, 85, 28463, 7375, ...,
33758],
'attention_mask':  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, ..., 1],
'labels':  [885, 14, 4211, 440, 539, 489, 2059, 15, 142, 951, 5184, 12764,
413, 196, 440, 16224, 24965, 7611, 142, 85, 28463, 7375, ..., 33758] }
```

Figure 5.6: One training example from PERO OCR MLM dataset.

Since the language model training is much more time and resource expensive than NER model training, more performance optimizations were implemented. The training script supports distributed setup, so multiple GPUs can be used. Mixed precision [35] training is also employed. To further accelerate the training process, gradient accumulation is used. Gradient accumulation is a technique, where optimizer performs a step after several training steps, instead of every training step. This effectively magnifies training batch size. Increasing batch size improves perplexity, as was shown in RoBERTa. Also skipping the optimization step saves computation time. In the script, gradient accumulation steps are calculated from fixed effective batch size, as shown in Equation 5.1. The effective batch size was set to 4,096. With a minor code change, it can be also configurable. Additional functionality of finding executable batch size is used. This functionality starts the training code with batch size set in configuration file. If the used GPUs run out of memory, another try with halved batch size is performed. Because of that, a lengthy process of finding runnable batch size is resolved; no manual training script reruns are needed. Script skeleton was taken over from Hugging Face Course Chapter 7 *Training a causal language model from scratch* [13].

$$gradient\ accumulation\ steps = \frac{effective\ batch\ size}{batch\ size \cdot number\ of\ GPU\ devices} \tag{5.1}$$

Cloud computation

For training, the MetaCentrum [7] was utilized. Shell script `train_mlm_model.sh` for training the language model on the MetaCentrum was implemented. It clones this thesis git repository, downloads datasets and tokenizers and prepares software environment. All required libraries are installed and then the `train_ml_model.py` script is started. On the MetaCentrum, jobs using GPUs are put into job scheduler queue `gpu@meta-pbs.metacentrum.cz`. This queue has maximum runtime of 24 hours, which is insufficient for language model training even with all above mentioned performance optimizations. Another option is to use queue `gpu_long@meta-pbs.metacentrum.cz` with maximum runtime 336 hours. Unfortunately, this queue has very long waiting times, so different solution was needed. Instead, using the `gpu@meta-pbs.metacentrum.cz` queue with modified training script was chosen. The `train_ml_model.py` script was modified to be able to save training before MetaCentrum job times out into a training state checkpoint and restore the training from this checkpoint on the next training job. The train state saving and loading use functions from Accelerator class from Accelerate library. All necessary objects are saved in a train state: the model, optimizer, learning rate scheduler, random number generators, etc. Additionally, current epoch, training steps and batch size are saved to a YAML file. Upon restoring the train state, last epochs is used in a training loop and the data loader object skips the number batches set by last training step. This ensures that the model starts to train from the same state when it was stopped. Besides before job timeout, script also saved train state during evaluation, which is performed every $200 \times$ *gradient accumulation steps*. The train state restoration is controlled by the script `--from_state` argument.

Chapter 6

Experiments

This chapter describes experiments performed in order to create a NER system suitable for identifying and correctly classifying named entities in Czech historical documents. Experimenting with various models, datasets and training hyperparameters also brings insights for further development. Since model training is highly configurable, new models, datasets a hyperparameters can be used in future experiments. The experiments were evaluated with its relevant metrics. The NER models time performance was also indicatively evaluated.

All model training was performed on MetaCentrum [7] cloud service, due to requirement of great computation power. The MetaCentrum provides several computation clusters (a set of computation nodes) with GPUs. Computation cluster *galdor* with 4 NVIDIA A40 GPUs was selected as the most powerful and available GPU cluster. The NVIDIA A40 GPU [10] offers a memory of 45,634 MiB, up to 74.8 TFLOPS with tensor 32bit (float) and memory bandwidth 696 GB/s.

First, experiments with masked language models are described, followed by experiments with Named Entity Recognition models.

6.1 Masked Language Modeling

Masked Language models were trained on PERO OCR MLM dataset (Subsection 5.2.2). In general, neural network models achieve better results with increasing number of their parameters. At the same time, the training and the inference time also increase. With few exceptions, published models are focused on the best results possible. However, the proposed NER system is intended to be used on big databases of historical documents, thus the computation costs and corresponding training and inference time need to be considered. For example RoBERTa BASE [31] model has 125M, RoBERTa LARGE 355M and RobeCzech [51] 125M parameters. The authors of these big language models use either huge number of GPUs (RoBERTa LARGE was trained with 1024 NVIDIA V100 GPUs for 1 day) or train for a very long time (RobeCzech was trained with 8 NVIDIA QUADRO P5000 GPUs for 3 months). Besides that, these big models are trained on a much bigger datasets than PERO OCR MLM (while PERO OCR MLM has 1,779.41MB, RoBERTa was trained on a combination of Books corpus and Wikipedia datasets totaling 160GB of uncompressed text data). To meet the performance demands and to be able to actually train the model in a reasonable time, smaller model variants were proposed and trained. The sizes of models were inspired by paper: *Well-Read Students Learn Better: On the Importance of Pre-training Compact Models* [55]. Model types are listed in Table 6.1. There are

5 types of models: Big, Medium, Small, Mini and Tiny. The main parameter affecting the resulting size of a model is the number of hidden layers (and also the quality of a model). Second most important parameter is the vocabulary size. Last important parameter is hidden size. The intermediate size and number of attention heads were fixed to the hidden size parameter. The intermediate size is calculated as $hidden\ size * 4$ and the number of attention heads is calculated as $hidden\ size\ /\ 64$.

Model type	**Big**	**Medium**	**Small**	**Mini**	**Tiny**
num_hidden_layers	8	6	4	4	8
hidden_size	512	256	512	256	128
intermediate_size	2048	1024	2048	1024	512
num_heads	8	4	8	4	2
# par_new_t_12k	31.9M	8.0M	19.3M	6.4M	3.2M
# par_new_t_26k	39.1M	11.6M	26.5M	10.0M	5.0M
# par_new_t_52k	52.4M	18.3M	39.8M	16.7M	8.4M
# par_old_t_52k	52.4M	18.3M	39.8M	16.7M	8.4M

Table 6.1: Statistics of trained RoBERTa language models.

Training hyperparameters setting was inspired by RoBERTa, RobeCzech and recently published paper dealing with pretraining Masked Language models: *Trained on 100 million words and still in shape BERT meets British National Corpus* [43]. In the last paper, the authors explore the effects of pretraining a Masked Language model on modestly-sized and representative dataset. All used hyperparameters are listed in Table 6.2. The number of epochs was set to 100, because of training time reasons. As is shown below, the resulting perplexity of models is relatively good. Initial batch size was set to 128. Most of the training experiments were not able to train on this value and following the finding executable batch size technique (described in Section 5.3), the batch size was halved. The largest model types were even trained on batch size of 32. Nevertheless the effective batch size was set to 4,096; so batch size halving prolonged the training time. AdamW optimizer was used with initial learning rate 1.e-3, β_1 0.9, β_2 0.98 and weight decay 0.1. Cosine learning rate scheduler was employed with warmup ratio (percentage of warmup steps) of 0.04.

The model training was performed on 4 NVIDIA A40 GPUs. This is the maximum number of GPUs on 1 computation node. More nodes could be requested, but the queue `gpu@meta-pbs.metacentrum.cz` waiting times would be too long. With 4 GPUs on 1 node, the waiting times were usually several days. The training took 1 – 3 days to complete. For this reasons, no extensive hyperparameter search was performed, only 1 experiment for each model type and dataset (vocabulary size). Total 20 experiments were conducted. Unfortunately, the experiments with dataset created by RobeCzech original tokenizer (`old_tokenizer_dts`) failed. This error was caused by omitting necessary file `config.json` in tokenizer's directory on MetaCentrum. Because of that, the model was trained with empty vocabulary. The training script did not report any problem during training. This fact was found out during subsequent NER training. Running experiments again was not possible due to time limitation. Therefore only 15 correct language models were obtained. In total, the language model training took several weeks.

The experiment results are presented in Table 6.3. As can be seen in the table, with increasing model size the quality of model also increases. The naming of "Medium" model might be misleading, because "Small" model has more parameters. Within each model type, perplexity is increasing with bigger vocabulary. An explanation for phenomena is probably

Hyperparameter	Big	Medium	Small	Mini	Tiny
Number of layers	8	6	4	4	8
Hidden size	512	256	512	256	128
FF intermediate size	2048	1024	2048	1024	512
Vocabulary size [k]	12, 26, 52, 52	12, 26, 52, 52	12, 26, 52, 52	12, 26, 52, 52	12, 26, 52, 52
Attention heads	8	4	8	4	2
Dropout	0.1	0.1	0.1	0.1	0.1
Number of epochs	100	100	100	100	100
Training steps [k]	26.5, 24.1, 22.5, 24.8	26.5, 24.1, 22.5, 24.8	26.5, 24.1, 22.5, 24.8	26.5, 24.1, 22.5, 24.8	26.5, 24.1, 22.5, 24.8
Batch size	4k	4k	4k	4k	4k
Warmup ratio	0.04	0.04	0.04	0.04	0.04
Learning rate	1.e-3	1.e-3	1.e-3	1.e-3	1.e-3
Learning rate decay	cosine	cosine	cosine	cosine	cosine
Weight decay	0.1	0.1	0.1	0.1	0.1
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
AdamW ϵ	1e-8	1e-8	1e-8	1e-8	1e-8
AdamW β_1	0.9	0.9	0.9	0.9	0.9
AdamW β_2	0.98	0.98	0.98	0.98	0.98
Gradient clipping norm	8.0	8.0	8.0	8.0	8.0

Table 6.2: Pretraining hyperparameters. Hyperparameters that differ in the model types are underlined, other are identical for all model types.

Experiment name	Model type	Dataset	Perplexity	Train time [h]
m_4L_256H_new_t_12k	Mini	new_t_12k	9.6591	20
m_4L_256H_new_t_26k	Mini	new_t_26k	11.6932	32.5
m_4L_256H_new_t_52k	Mini	new_t_52k	13.3614	30.5
m_4L_512H_new_t_12k	Small	new_t_12k	5.1387	32
m_4L_512H_new_t_26k	Small	new_t_26k	6.5031	34.5
m_4L_512H_new_t_52k	Small	new_t_52k	7.4445	43.5
m_6L_256H_new_t_12k	Medium	new_t_12k	8.0355	24.5
m_6L_256H_new_t_26k	Medium	new_t_26k	10.0622	27.5
m_6L_256H_new_t_52k	Medium	new_t_52k	11.5481	34.5
m_8L_128H_new_t_12k	Tiny	new_t_12k	15.0521	21.5
m_8L_128H_new_t_26k	Tiny	new_t_26k	18.6940	24.5
m_8L_128H_new_t_52k	Tiny	new_t_52k	21.1557	28.5
m_8L_512H_new_t_12k	Big	new_t_12k	3.9487	51
m_8L_512H_new_t_26k	Big	new_t_26k	4.8481	53
m_8L_512H_new_t_52k	Big	new_t_52k	5.6538	63

Table 6.3: Masked Language Modeling experiment results.

in undertraining of the bigger models. The training time increases with the number of model parameters and is especially high for the Big model type. This is caused by the fact, that batch size was decreased from 128 to 32 due to low GPU memory. The number of

experiment was limited by computation resources and time, more extensive experimenting is needed.

6.2 Named Entity Recognition

Experiments with Named Entity Recognition models were performed on several datasets. Czech Named Entity Corpus (CNEC) 2.0 CoNLL [47][29] dataset with flat entities, Czech Historical Named Entity Corpus (CHNEC) 1.0 [27] and PERO OCR NER (PONER) 1.0 (Subsection 5.2.1) datasets were used. Datasets were converted to CHNEC named entities format and prepared for training (as described in Subsection 5.1.3). Before training train and validation splits of all specified datasets were concatenated and the model was on this dataset. Trained model was then evaluated on separate datasets test splits.

The SumeCzech-NER [34] dataset could not be used, as the description of the dataset preparation was no longer reproducible. This dataset consist of only NER annotations in IOB format for SumeCzech 1.0 [50] dataset. The SumeCzech 1.0 dataset was downloaded from Common Crawl [1] after fixing the provided download script. Unfortunately, the SumeCzech-NER preparation depends on sentence and word tokenization from NLTK [18] Punkt model and the present Punkt model version is different from the one used by the authors. The quality of annotations is also very limited; the annotations were produced by a model with 78.45 F1-score.

For experiment evaluation, the F1-score metric (Section 2.4) was used. A named entity is considered correct only if both the span and the type of the named entity are correct. All model training used 1 NVIDIA A40 GPU. The training times were in range of 10 to 60 minutes, so the use of multiple GPUs was not necessary. First experiments with known Czech language models are described, followed by experiments on own Masked Language models.

6.2.1 Experiments with known Czech models

Experiments were conducted with following Czech language models: RobeCzech [51], Slavic-BERT [17] and Czert-B [49]. Several experiment sets were created, as is listed below. All experiment results are averaged over 3 runs. The hyperparameters batch size, learning rate decay were fixed to values of 16 for batch size and "linear" for learning rate decay. AdamW β_1 was set to 0.9 and β_2 to 0.999.

RobeCzech base experiments

The RobeCzech model was trained on CNEC and CHNEC datasets. This initial experiment set was aimed to gain insights which hyperparameters most influence the model results and influenced the following experiments. Hyperparameter search was done on: number of epochs (3, 5, 10), learning rate (1.e-5, 2.e-5, 3.e-5), weight decay (0.0, 0.01) and warmup ratio (0.0, 0.06). The results are listed in Tables 6.4 6.5. Hyperparameters are encoded in experiments names. It is clear, that the experiments with lowest learning rate 1.e-5 and lowest number of epochs tend to have worse results. Naturally, learning rate and number of epochs are bigger factors than weight decay and warmups. Experiments with same learning rate and number of epochs, but different in weight decay and warmups have similar results. The differences among models with same learning rate and number epochs are very low, which clearly suggests that weight decay and warmup minimally influence the results.

In the case of F1 measured on CNEC 2.0 test split (average), first 8 of first 10 results are scored by models trained with number epochs set to 10. In all 3 runs, first 8 experiments reach above 88 F1. Best results are achieved by models with learning rate 3.e-5 and number epochs 10. This gives a possibility to achieve even better results with increasing both hyperparameters. For CNEC 2.0 test F1, there is a low decrease between results in descending order, whereas for CHNEC 1.0 test F1 the decrease is higher and there is bigger variance between the best and worst result. This is probably caused by the small size of dataset. The CHNEC 1.0 test split has only 391 examples. The largest difference between the best and worst results is in Run3 - more than 7% F1. Overall, models with bigger learning rate and num of epochs achieve better results, but there is greater difference in order than for CNEC 2.0. This can be also explained by the small size of dataset. The average best model `linear_lr_3e5_10_epochs_wd_wr` for CNEC 2.0 achieves 88.85% F1, which is greater than RobeCzech model by 1.36%. The average best model `linear_lr_3e5_10_epochs_wr` for CHNEC 1.0 achieves 87.19% F1, which is greater than CHNEC 1.0 model by 5.19%. New state-of-the-art results were obtained.

Exp name	**CNEC 2.0 test**
linear_lr_3e5_10_epochs_wd_wr	88.8462
linear_lr_3e5_10_epochs	88.7701
linear_lr_3e5_10_epochs_wd	88.7018
linear_lr_2e5_10_epochs	88.6883
linear_lr_3e5_10_epochs_wr	88.6361
linear_lr_2e5_10_epochs_wd	88.4940
linear_lr_2e5_10_epochs_wd_wr	88.4382
linear_lr_2e5_10_epochs_wr	88.3092
linear_lr_3e5_5_epochs_wd	88.0942
linear_lr_3e5_5_epochs	88.0337
linear_lr_3e5_5_epochs_wd_wr	87.9874
linear_lr_3e5_5_epochs_wr	87.8325
linear_lr_2e5_5_epochs_wd_wr	87.7829
linear_lr_1e5_10_epochs_wd	87.5831
baseline_linear_lr_2e5_5_epochs	87.5783
linear_lr_1e5_10_epochs_wr	87.5293
linear_lr_1e5_10_epochs	87.3479
linear_lr_1e5_10_epochs_wd_wr	87.2911
linear_lr_2e5_5_epochs_wr	87.1676
linear_lr_2e5_5_epochs_wd	87.1213
linear_lr_2e5_3_epochs_wd_wr	86.3061
linear_lr_2e5_3_epochs_wr	86.2600
linear_lr_2e5_3_epochs_wd	86.2049
linear_lr_2e5_3_epochs	86.1972
linear_lr_1e5_5_epochs_wr	86.1010
linear_lr_1e5_5_epochs	85.7330
linear_lr_1e5_5_epochs_wd	85.6914
linear_lr_1e5_5_epochs_wd_wr	85.5893

Table 6.4: RobeCzech base experiments results on CNEC test split. Experiments are listed in descending order.

Exp name	CHNEC 1.0 test
linear_lr_3e5_10_epochs_wr	87.1938
linear_lr_3e5_10_epochs_wd_wr	86.3387
linear_lr_2e5_10_epochs	86.1782
linear_lr_3e5_10_epochs	86.1438
linear_lr_3e5_5_epochs_wr	85.9637
linear_lr_2e5_10_epochs_wd	85.7523
linear_lr_3e5_5_epochs_wd_wr	85.6452
linear_lr_3e5_5_epochs	85.6016
linear_lr_3e5_10_epochs_wd	85.5275
linear_lr_2e5_5_epochs_wd	85.3444
linear_lr_2e5_10_epochs_wd_wr	85.2069
linear_lr_2e5_10_epochs_wr	85.0981
linear_lr_3e5_5_epochs_wd	84.8112
linear_lr_1e5_10_epochs_wd_wr	84.7795
linear_lr_1e5_10_epochs_wd	84.7032
linear_lr_2e5_5_epochs_wd_wr	84.6745
linear_lr_2e5_5_epochs_wr	84.3771
linear_lr_1e5_10_epochs_wr	84.2870
baseline_linear_lr_2e5_5_epochs	84.2685
linear_lr_2e5_3_epochs_wd_wr	84.1756
linear_lr_1e5_10_epochs	84.0974
linear_lr_2e5_3_epochs_wd	83.6815
linear_lr_1e5_5_epochs_wr	83.3998
linear_lr_2e5_3_epochs	83.3521
linear_lr_1e5_5_epochs_wd_wr	82.9886
linear_lr_2e5_3_epochs_wr	82.9206
linear_lr_1e5_5_epochs	82.8305
linear_lr_1e5_5_epochs_wd	81.9030

Table 6.5: RobeCzech base experiments results on CHNEC test split. Experiments are listed in descending order.

RobeCzech more epochs

The results of base set of experiments suggested, that with training on more epochs, better results can be achieved. Since the learning rate 3.e-5 proved to perform the best, only this value is included in the experiments. The results are shown in Tables 6.6 6.7. Except for the best validation result 0.8713 F1, all results are very similar among each dataset. In comparison with base experiments, no benefit from training on more epochs is seen. The results are comparable, in case of the best base experiment `linear_lr_3e5_10_epochs_wd_wr` even very slightly better. CHNEC results show greater difference between the best and worst result than CNEC.

Slavic-BERT experiments

To enable comparison with RobeCzech, a different Czech language model was trained. Slavic-BERT is a multilingual BERT model, pretrained on 4 languages: Czech, Bulgarian, Polish and Russian. Its architecture follows BERT BASE with 177M parameters

Exp name	CNEC 2.0 test
linear_lr_3e5_20_epochs_wd	88.6677
linear_lr_3e5_15_epochs_wd_wr	88.6376
linear_lr_3e5_20_epochs_wr	88.5532
linear_lr_3e5_15_epochs_wr	88.5527
linear_lr_3e5_20_epochs	88.4945
linear_lr_3e5_20_epochs_wd_wr	88.3507
linear_lr_3e5_15_epochs_wd	88.2413
linear_lr_3e5_15_epochs	88.1368

Table 6.6: RobeCzech experiments with more epochs results on CNEC test split. Experiments are listed in descending order.

Exp name	CHNEC 1.0 test
linear_lr_3e5_20_epochs_wd	87.4118
linear_lr_3e5_15_epochs_wd_wr	86.8378
linear_lr_3e5_20_epochs_wr	86.6167
linear_lr_3e5_20_epochs	86.5554
linear_lr_3e5_15_epochs_wr	86.4139
linear_lr_3e5_15_epochs_wd	86.1648
linear_lr_3e5_20_epochs_wd_wr	85.7825
linear_lr_3e5_15_epochs	85.2996

Table 6.7: RobeCzech experiments with more epochs results on CHNEC test split. Experiments are listed in descending order.

(RobeCzech has 125M parameters). Hyperparameter search was done on: number of epochs (10, 15, 20), learning rate (2.e-5, 3.e-5), weight decay (0.0, 0.01) and warmup ratio (0.0, 0.06). The results are listed in Tables 6.8 6.9. As can be seen in the tables, results are very similar among each dataset. Experiments with more epochs and learning rate 3.e-5 tend to have better results. In comparison with base experiments, the results are worse, as was expected, since RobeCzech is based on more advanced architecture and pretrained on different data.

Czert experiments

Another Czech language model Czert was also added. Czert is Czech version of BERT model, trained on only Czech corpora. The Czert authors report training on more than 340K senteces on training corpora Czech national corpus, Czech Wikipedia and Crawled of Czech news, in total 37GB of uncompressed text data. The number of model parameters is 110M. The Czert paper report 86.27 F1 on CNEC dataset with Czert-B. The model variant used in experiments is Czert-B cased, that follows BERT BASE architecture. Same hyperparameter search as for Slavic-BERT was performed. The results are listed in Tables 6.10 6.11. Lower F1 results than with Slavic-BERT and RobeCzech are achieved. Czert-B model has the lowest number of parameters and is learned on less data than RobeCzech. There is greater variance in CHNEC result than in Slavic-BERT and RobeCzech. This can be also explained by lesser number of model parameters and by the fact that CHNEC contains only third of the examples as CNEC. CNEC results show signs, that warmup rate and warmup rate with weight decay actually hurt NER training. Another interesting fact is that greater number

Exp name	CNEC 2.0 test
slavic_bert_linear_lr_3e5_15_epochs_wd_wr	86.6065
slavic_bert_linear_lr_3e5_20_epochs_wd_wr	86.5576
slavic_bert_linear_lr_3e5_10_epochs_wd	86.4728
slavic_bert_linear_lr_2e5_10_epochs_wr	86.3629
slavic_bert_linear_lr_3e5_15_epochs_wd	86.3152
slavic_bert_linear_lr_3e5_15_epochs	86.3070
slavic_bert_linear_lr_3e5_20_epochs	86.2699
slavic_bert_linear_lr_2e5_10_epochs_wd	86.2441
slavic_bert_linear_lr_2e5_10_epochs_wd_wr	86.1816
slavic_bert_linear_lr_3e5_20_epochs_wr	86.1651
slavic_bert_linear_lr_3e5_15_epochs_wr	86.1164
slavic_bert_linear_lr_3e5_10_epochs_wd_wr	86.0540
slavic_bert_linear_lr_3e5_10_epochs	86.0259
slavic_bert_linear_lr_3e5_20_epochs_wd	85.8328
slavic_bert_linear_lr_2e5_10_epochs	85.8225
slavic_bert_linear_lr_3e5_10_epochs_wr	85.4577

Table 6.8: Slavic-BERT experiments results on CNEC test split. Experiments are listed in descending order.

Exp name	CHNEC 1.0 test
slavic_bert_linear_lr_3e5_15_epochs_wr	85.6803
slavic_bert_linear_lr_3e5_20_epochs	85.6146
slavic_bert_linear_lr_3e5_10_epochs_wd	85.3077
slavic_bert_linear_lr_2e5_10_epochs_wd	85.2621
slavic_bert_linear_lr_2e5_10_epochs_wr	85.1259
slavic_bert_linear_lr_3e5_15_epochs	85.1191
slavic_bert_linear_lr_3e5_15_epochs_wd	85.1190
slavic_bert_linear_lr_2e5_10_epochs	85.0053
slavic_bert_linear_lr_3e5_10_epochs_wr	84.8835
slavic_bert_linear_lr_3e5_20_epochs_wd	84.8225
slavic_bert_linear_lr_3e5_20_epochs_wd_wr	84.8202
slavic_bert_linear_lr_3e5_10_epochs_wd_wr	84.7845
slavic_bert_linear_lr_3e5_10_epochs	84.6580
slavic_bert_linear_lr_3e5_15_epochs_wd_wr	84.2058
slavic_bert_linear_lr_2e5_10_epochs_wd_wr	84.0861
slavic_bert_linear_lr_3e5_20_epochs_wr	84.0767

Table 6.9: Slavic-BERT experiments results on CHNEC test split. Experiments are listed in descending order.

of epochs does not bring extra benefits, especially with CHNEC. Remarkable finding is that original results presented in the paper were not matched and showed relatively big difference of 1.17% F1.

Exp name	CNEC 2.0 test
czertB_linear_lr_3e5_20_epochs_wd	85.1049
czertB_linear_lr_3e5_15_epochs	85.0896
czertB_linear_lr_3e5_15_epochs_wr	84.9028
czertB_linear_lr_3e5_20_epochs	84.8831
czertB_linear_lr_3e5_10_epochs_wd_wr	84.8399
czertB_linear_lr_2e5_10_epochs	84.7451
czertB_linear_lr_3e5_10_epochs_wd	84.7285
czertB_linear_lr_3e5_15_epochs_wd	84.6423
czertB_linear_lr_3e5_20_epochs_wr	84.5822
czertB_linear_lr_3e5_10_epochs_wr	84.5666
czertB_linear_lr_2e5_10_epochs_wd	84.4971
czertB_linear_lr_3e5_10_epochs	84.4397
czertB_linear_lr_2e5_10_epochs_wr	84.3985
czertB_linear_lr_3e5_20_epochs_wd_wr	84.2631
czertB_linear_lr_3e5_15_epochs_wd_wr	84.0965
czertB_linear_lr_2e5_10_epochs_wd_wr	84.0010

Table 6.10: Czert-B experiments results on CNEC test split. Experiments are listed in descending order.

Exp name	CHNEC 1.0 test
czertB_linear_lr_2e5_10_epochs	84.0112
czertB_linear_lr_2e5_10_epochs_wr	83.8381
czertB_linear_lr_3e5_10_epochs_wd_wr	83.7556
czertB_linear_lr_3e5_15_epochs_wr	83.5913
czertB_linear_lr_3e5_20_epochs	83.5603
czertB_linear_lr_2e5_10_epochs_wd_wr	83.5307
czertB_linear_lr_3e5_15_epochs	83.4023
czertB_linear_lr_3e5_10_epochs_wr	83.1894
czertB_linear_lr_3e5_10_epochs	83.1503
czertB_linear_lr_3e5_20_epochs_wr	83.1412
czertB_linear_lr_3e5_15_epochs_wd	83.0510
czertB_linear_lr_3e5_15_epochs_wd_wr	83.0046
czertB_linear_lr_3e5_10_epochs_wd	82.8246
czertB_linear_lr_2e5_10_epochs_wd	82.7206
czertB_linear_lr_3e5_20_epochs_wd	82.6186
czertB_linear_lr_3e5_20_epochs_wd_wr	82.4114

Table 6.11: Czert-B experiments results on CHNEC test split. Experiments are listed in descending order.

RobeCzech PONER experiments

Since the PONER dataset was finished long after experiments above, experiment set including PONER dataset is presented additionally here. Weight decay and warmup rate proved to be insignificant and were omitted in experiments with RobeCzech on PONER dataset. Learning rate of 3.e-5 also showed the best results, so the hyperparameter search was limited to number of epochs. Again all datasets were concatenated and used together

during training and the model was evaluated separately on each dataset. The results are listed in Table 6.12. PONER test split includes 50% of all dataset examples, in order to show the model quality on historical documents. The PONER result can be judged as very good, since the dataset contains more complicated and longer named entities. The CNEC and CHNEC results are not affected by additional PONER dataset.

Exp name	CNEC 2.0 test	CHNEC 1.0 test	PONER 1.0 test
linear_lr_3e5_10_epochs_poner	0.8861	0.8742	0.8695
linear_lr_3e5_15_epochs_poner	0.8854	0.8758	0.8690
linear_lr_3e5_20_epochs_poner	0.8833	0.8716	0.8705

Table 6.12: RobeCzech experiments results on all test splits including PONER.

Experiments conclusion

Extensive experiments were conducted with various models and datasets. RobeCzech confirmed its qualities among other BERT-like models. CNEC dataset results were never matched by CHNEC, which was caused by size difference. A limit of the number of epochs was found to be 10 or 15 epochs for all three models. Weight decay and warmup rate proved to be insignificant for NER training and should be used for tasks with much bigger datasets like Masked Language Modeling. Results on the PONER dataset can be regarded as very good. Most importantly, new state-of-the-art results were achieved, outperforming previous best results by 1.36% on CNEC and by 5.19% on CHNEC datasets.

6.2.2 Experiments with own Masked language models

The Masked Language models described in Section x were used for NER training. All 15 language models were used. Linear learning rate decay was used and AdamW β_1 was set to 0.9 and β_2 to 0.999. Batch size was increased to 32, as no memory problems occurred. With insights gained during experiments with known models, learning rate was fixed to the value 3.e-5, weight decay and warmup rate hyperparameters were omitted. Due to high number of experiments, the experiment configurations were generated by a script. Three different dataset settings are used: CNEC + CHNEC, CNEC + CHNEC + PONER and PONER only. To find suitable number of epochs, a search was performed. With the smallest (Tiny model, 8 hidden layers, 128 hidden size and vocabulary 12k) and the largest (Big model, 8 hidden layers, 512 hidden size and vocabulary 52k) models, gradually several values were tested until the experiment result stopped to benefit for extra epochs. Values of 20, 25, 30, 35, 40, 45, 50 and 55 were tried. 40 and 50 epochs were then chosen for experiments. For each language model, configurations with 3 dataset settings and 2 different numbers of epochs were created, in total 90 experiment configurations. All experiment results are averaged over 3 runs.

The models achieve worse results than those in RobeCzech experiments, as was expected. Also, the size of vocabulary was an important factor. Training for more epochs is necessary, because underlying language model was trained on less data than RobeCzech and thus achieves worse language understanding. The models trained for 50 epochs clearly shows better results. With the CNEC + CHNEC dataset setting, the best model achieved 79.9663 F1 on CNEC and 86.6073 F1 on CHNEC. With the CNEC + CHNEC + PONER

dataset setting, the best model achieved 81.0050 F1 on CNEC, 87.0109 F1 on CHNEC and 84.0611 F1 on PONER. The CHNEC results are comparable to those in RobeCzech experiments. There is a benefit from more data from PONER. With the CNEC + CHNEC dataset setting, the best model achieved 83.2326 F1 on PONER.

Chapter 7

Conclusion

The goal of this thesis was to design, implement and experiment with a system for recognition of named entities in historical documents text. I studied theoretical backgrounds of Natural language processing, information extraction, language models, neural networks and Transformers, together with Named entity recognition (NER) task and relevant datasets.

The theoretical backgrounds were used to design a NER system. This system is based on masked language models trained on several NER datasets, notably Czech Named Entity Corpus 2.0 and Czech Historical Named Entity Corpus 1.0. Creation of own datasets was proposed. The system is intended to be included in PERO OCR project pipeline. PERO OCR is an application for document digitalization. The NER system inputs will be OCR-processed historical documents in text format. The outputs are recognized named entities, their text spans and named entity types. To be able to compare with previous works, a CHNEC named entity schema was adopted. In order to meet computation requirements, smaller language model architectures were proposed.

Several own datasets were created. PERO OCR NER 1.0 is a new NER dataset created from historical Czech chronicles. The dataset is comparable in size to Czech Named Entity Corpus 2.0. This dataset can be used in future works and can be published. PERO OCR Books and PERO OCR Periodicals are new text datasets created from books OCR transcriptions and periodicals (newspapers, magazines) OCR transcriptions. These datasets can be used to train a Masked Language model. Implemented training process is highly configurable, allowing simple way to create, modify and reproduce experiments.

Extensive experiments were conducted. First, Masked Language models were trained on own text datasets PERO OCR Books and PERO OCR Periodicals. Second, experiments with several known Czech language models were performed and then own Masked Language models were used for NER training. Experiments were evaluated on Czech Named Entity Corpus 2.0, Czech Historical Named Entity Corpus 1.0 and PERO OCR NER 1.0 NER datasets. New state-of-the-art results were achieved, outperforming previous best results by 1.36% on Czech Named Entity Corpus 2.0 and by 5.19% on Czech Historical Named Entity Corpus 1.0 datasets.

For future development, larger NER datasets would benefit the NER system performance. More text data for Masked Language model training would also be beneficial. Multilingual models instead of only Czech model are an option.

Bibliography

[1] *Common Crawl* [online]. Common Crawl Foundation [cit. 2023-07-26]. Available at: https://commoncrawl.org/.

[2] *Digitální archiv* [online]. Státní oblastní archiv v Třeboni [cit. 2022-01-24]. Available at: https://digi.ceskearchivy.cz/Uvod.

[3] *Digitální archiv Zemského archivu v Opavě* [online]. Zemský archiv v Opavě [cit. 2022-01-24]. Available at: https://www.archives.cz/web/digitalni_archiv/.

[4] *Hugging Face* [online]. Hugging Face, Inc. [cit. 2023-07-23]. Available at: https://huggingface.co/.

[5] *Institute of Formal and Applied Linguistics* [online]. Faculty of Mathematics and Physics, Charles University, Czech Republic [cit. 2022-01-22]. Available at: https://ufal.mff.cuni.cz/.

[6] *Language-detection-fine-tuned-on-xlm-roberta-base* [online]. Hugging Face, Inc. [cit. 2023-07-26]. Available at: https://huggingface.co/ivanlau/language-detection-fine-tuned-on-xlm-roberta-base.

[7] *MetaCentrum* [online]. MetaCentrum [cit. 2023-07-23]. Available at: https://metavo.metacentrum.cz/.

[8] *Mit-b0* [online]. Hugging Face, Inc. [cit. 2023-07-26]. Available at: https://huggingface.co/nvidia/mit-b0/.

[9] *NLP group* [online]. Department of Computer Science and Engineering, University of West Bohemia, Czech Republic [cit. 2022-01-22]. Available at: https://nlp.kiv.zcu.cz/.

[10] *NVIDIA A40* [online]. Nvidia Corporation [cit. 2023-07-26]. Available at: https://images.nvidia.com/content/Solutions/data-center/a40/nvidia-a40-datasheet.pdf.

[11] *Porta fontium* [online]. Státní oblastní archiv v Plzni [cit. 2022-01-24]. Available at: https://www.portafontium.eu/.

[12] *Token classification* [online]. Hugging Face, Inc. [cit. 2023-07-26]. Available at: https://huggingface.co/course/en/chapter7/2?fw=pt.

[13] *Training a causal language model from scratch* [online]. Hugging Face, Inc. [cit. 2023-07-26]. Available at: https://huggingface.co/course/en/chapter7/6?fw=pt.

[14] *WikiStats – List of Wikipedias* [online]. [cit. 2022-01-13]. Available at: https://wikistats.wmcloud.org/display.php?t=wp.

[15] *Český národní korpus* [online]. Ústav Českého národního korpusu [cit. 2022-01-13]. Available at: https://www.korpus.cz/.

[16] Alec Radford, T. S. and Sutskever, I. Improving Language Understanding by Generative Pre-Training. In:. 2018.

[17] Arkhipov, M., Trofimova, M., Kuratov, Y. and Sorokin, A. Tuning Multilingual Transformers for Language-Specific Named Entity Recognition. In: *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*. Florence, Italy: Association for Computational Linguistics, August 2019, p. 89–93. DOI: 10.18653/v1/W19-3712. Available at: https://aclanthology.org/W19-3712.

[18] Bird, S., Klein, E. and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. „ O'Reilly Media, Inc.“, 2009.

[19] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*. Cambridge, MA: MIT Press. 2017, vol. 5, p. 135–146. DOI: 10.1162/tacl_a_00051. Available at: https://aclanthology.org/Q17-1010.

[20] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. et al. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*. february 2011, vol. 12, p. 2493–2537.

[21] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G. et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020.

[22] Devlin, J., Chang, M., Lee, K. and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. 2018, abs/1810.04805. Available at: http://arxiv.org/abs/1810.04805.

[23] Gupta, S. *Named Entity Recognition: Applications and Use Cases* [online]. February 2018 [cit. 2022-01-22]. Available at: https://towardsdatascience.com/named-entity-recognition-applications-and-use-cases-acdbf57d595e.

[24] Hinton, G., Vinyals, O. and Dean, J. *Distilling the Knowledge in a Neural Network*. 2015.

[25] Huang, Z., Xu, W. and Yu, K. Bidirectional LSTM-CRF Models for Sequence Tagging. *ArXiv*. 2015, abs/1508.01991.

[26] Hubková, H. and Kral, P. Transfer Learning for Czech Historical Named Entity Recognition. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Held Online: INCOMA Ltd., September 2021, p. 576–582. Available at: https://aclanthology.org/2021.ranlp-main.65.

[27] Hubková, H., Kral, P. and Pettersson, E. Czech Historical Named Entity Corpus v 1.0. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020,

p. 4458–4465. ISBN 979-10-95546-34-4. Available at: https://aclanthology.org/2020.lrec-1.549.

[28] Kodym, O. and Hradiš, M. *Page Layout Analysis System for Unconstrained Historic Documents*. 2021.

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

[30] Li, J., Sun, A., Han, R. and Li, C. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*. march 2020, PP, p. 1–1. DOI: 10.1109/TKDE.2020.2981314.

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

[32] Liu, Z., Lin, Y. and Sun, M. Word Representation. In: *Representation Learning for Natural Language Processing*. Singapore: Springer Singapore, 2020, p. 13–41. DOI: 10.1007/978-981-15-5573-2_2. ISBN 978-981-15-5573-2. Available at: https://doi.org/10.1007/978-981-15-5573-2_2.

[33] Marcińczuk, M. and Oleksy, M. Inforex — a Collaborative Systemfor Text Corpora Annotation and Analysis Goes Open. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., September 2019, p. 711–719. DOI: 10.26615/978-954-452-056-4_083. Available at: https://www.aclweb.org/anthology/R19-1083.

[34] Marek, P., Müller, Š., Konrád, J., Lorenc, P., Pichl, J. et al. Text Summarization of Czech News Articles Using Named Entities. *Prague Bulletin of Mathematical Linguistics*. Charles University in Prague, Karolinum Press. Apr 2021, vol. 116, no. 1, p. 5–26. DOI: 10.14712/00326585.012. ISSN 0032-6585. Available at: http://dx.doi.org/10.14712/00326585.012.

[35] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E. et al. *Mixed Precision Training*. 2018.

[36] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 3111–3119. NIPS'13.

[37] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, p. 8024–8035. Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[38] Pennington, J., Socher, R. and Manning, C. GloVe: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, October 2014, p. 1532–1543. DOI: 10.3115/v1/D14-1162. Available at: https://aclanthology.org/D14-1162.

[39] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C. et al. Deep Contextualized Word Representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, p. 2227–2237. DOI: 10.18653/v1/N18-1202. Available at: https://aclanthology.org/N18-1202.

[40] Piskorski, J., Laskova, L., Marcińczuk, M., Pivovarova, L., Přibáň, P. et al. The Second Cross-Lingual Challenge on Recognition, Normalization, Classification, and Linking of Named Entities across Slavic Languages. In: *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*. Florence, Italy: Association for Computational Linguistics, August 2019, p. 63–74. Available at: https://www.aclweb.org/anthology/W19-3709.

[41] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. et al. Language Models are Unsupervised Multitask Learners. In:. 2019. Available at: https://api.semanticscholar.org/CorpusID:160025533.

[42] Ruder, S. *NLP-progress: Named entity recognition* [online]. [cit. 2022-01-12]. Available at: http://nlpprogress.com/english/named_entity_recognition.html.

[43] Samuel, D., Kutuzov, A., Øvrelid, L. and Velldal, E. *Trained on 100 million words and still in shape: BERT meets British National Corpus*. 2023.

[44] Sanh, V., Debut, L., Chaumond, J. and Wolf, T. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020.

[45] Sennrich, R., Haddow, B. and Birch, A. Neural Machine Translation of Rare Words with Subword Units. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, p. 1715–1725. DOI: 10.18653/v1/P16-1162. Available at: https://aclanthology.org/P16-1162.

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

[47] Ševčíková, M., Žabokrtský, Z., Straková, J. and Straka, M. *Czech Named Entity Corpus 2.0*. 2014. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. Available at: http://hdl.handle.net/11858/00-097C-0000-0023-1B22-8.

[48] Sharnagat, R. *Named Entity Recognition: A Literature Survey* [online]. 2014. Available at: https://www.cfilt.iitb.ac.in/resources/surveys/rahul-ner-survey.pdf.

[49] Sido, J., Pražák, O., Přibáň, P., Pašek, J., Seják, M. et al. *Czert – Czech BERT-like Model for Language Representation*. 2021.

[50] Straka, M., Mediankin, N., Kocmi, T., Žabokrtský, Z., Hudeček, V. et al. SumeCzech: Large Czech News-Based Summarization Dataset. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. Available at: https://aclanthology.org/L18-1551.

[51] Straka, M., Náplava, J., Straková, J. and Samuel, D. RobeCzech: Czech RoBERTa, a Monolingual Contextualized Language Representation Model. In: *24th International Conference on Text, Speech and Dialogue*. Cham, Switzerland: Springer, 2021, p. 197–209. ISBN 978-3-030-83526-2.

[52] Straková, J., Straka, M. and Hajič, J. Neural Architectures for Nested NER through Linearization. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, p. 5326–5331. DOI: 10.18653/v1/P19-1527. Available at: https://aclanthology.org/P19-1527.

[53] Tjong Kim Sang, E. F. and De Meulder, F. 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*. 2003, p. 142–147. Available at: https://aclanthology.org/W03-0419.

[54] Tkachenko, M., Malyuk, M., Holmanyuk, A. and Liubimov, N. *Label Studio: Data labeling software*. 2020-2022. Open source software available from https://github.com/heartexlabs/label-studio. Available at: https://github.com/heartexlabs/label-studio.

[55] Turc, I., Chang, M.-W., Lee, K. and Toutanova, K. *Well-Read Students Learn Better: On the Importance of Pre-training Compact Models*. 2019.

[56] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. *Attention Is All You Need*. 2017.

[57] Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M. et al. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. *CoRR*. 2021, abs/2105.15203. Available at: https://arxiv.org/abs/2105.15203.

[58] Yamada, I., Asai, A., Shindo, H., Takeda, H. and Matsumoto, Y. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, November 2020, p. 6442–6454. DOI: 10.18653/v1/2020.emnlp-main.523. Available at: https://aclanthology.org/2020.emnlp-main.523.