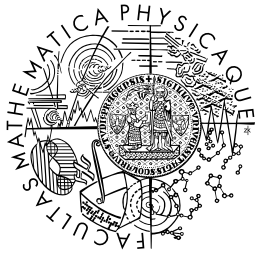


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Zuzana Šimečková

Entity Relationship Extraction

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: RNDr. Milan Straka, Ph.D.
Study programme: Computer Science
Study branch: IUI

Prague 2020

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank Profinit EU s.r.o. that they allowed me to create this thesis as part of their Big Data department, and Ing. Marek Sušický, the head of the department, for supervision. I would also like to thank my supervisor RNDr. Milan Straka, Ph.D. for his guidance and expertise.

Title: Entity Relationship Extraction

Author: Bc. Zuzana Šimečková

Institute: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Milan Straka, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Relationship extraction is the task of extracting semantic relationships between entities from a text. We create a Czech Relationship Extraction Dataset (CERED) using distant supervision on Wikidata and Czech Wikipedia. We detail the methodology we used and the pitfalls we encountered. Then we use CERED to fine-tune a neural network model for relationship extraction. We base our model on BERT – a linguistic model pre-trained on extensive unlabeled data. We demonstrate that our model performs well on existing English relationship datasets (Semeval 2010 Task 8, TACRED) and report the results we achieved on CERED.

Keywords: entities, named entities, entity relationship, entity relationship extraction, Czech, BERT

Contents

Introduction

Relationship extraction is the task of extracting semantic relationships from a text. It is closely connected to named entity recognition, the task of tagging entities in text with their corresponding type, and entity linking, the task of disambiguating named entities to a knowledge base. If all these task are used together, we can, for example, construct a knowledge database automatically from text.

For English, multiple attempts were made to solve or at least advance in relationship extraction, varying both in the exact formulation of the task and in used technologies.

In this thesis, we focus on relationship extraction in the Czech language. Our goal is to construct a neural network model that extracts relationships from sentences with labelled subject and object entities.

In modern machine learning, the quality of datasets plays a key role. In the first part of this thesis, we address the absence of a Czech dataset for relationship extraction. We generate our dataset by aligning Wikidata[1] with Czech Wikipedia.[2] This type of aligning is referred to as distant supervision.

Given the absence of a dataset, we also deal with an absence of a baseline our model could be compared to. We evaluate the performance of the proposed model architecture on some well known English datasets to show that it is comparable to the state-of-the-art results.

Thesis Organization

This thesis is split into two parts. Before we dive into the first part, we provide background information that is relevant to this thesis, such as more details on relationship extraction, related terminology and further motivation. We also briefly introduce the Czech language.

The first part focuses on datasets. First, we present some existing supervised and unsupervised datasets. Second, we propose methodology for generating a Czech relationship extraction dataset via distant supervision. Using the methodology we elaborate on the implementation process and on the generated dataset - the Czech Relationship Extraction Dataset (CERED).

In the second part, we overview some concepts, that are used in modern natural language processing models. Then we describe the architecture of our model and compare its performance with reported results on popular English relationship extraction datasets. We also report our results on CERED.

[1] https://www.wikidata.org/wiki/
[2] https://cs.wikipedia.org/wiki/

1. Background

This thesis is split into two parts that focus on different aspects of relationship extraction in the Czech language. In this chapter, we provide a glossary of some NLP and relation extraction terms, we elaborate on the task itself, and we include an introduction to the Czech language for non-Czech readers.

1.1 Terminology

Terminology in NLP subtasks is often non-standardized or not exact. We attempt to introduce the most important concepts for our work as exactly as possible while respecting the terms that seem to be already established.

Relation in this context is an abstraction of a semantic relation, for example, a father relation. A relation has a type (father), it is binary (between a son and the father) and oriented (the father and the son are not interchangeable), and describes the relationship between a subject (the son) and an object (the father). We will use the term relation as an equivalent for its type and the term relationship for an instance of the relation.

Subject and **Object**. The subject is the first argument of a relation, and the object is the second. In the sentence *"Albus Severus is Harry Potter's son."* a relation of type SON is captured, the subject is Harry and the object is Albus Severus. The reasoning for this choice of direction is as follows: suppose we are gathering information about Harry, then we would probably have both the information that his son is Albus Severus and that his father is James. So we are gathering information about the subject (Harry Potter), even though in most sentences Harry is likely to be the grammatical object: *"James is Harry's father."* We will use the notation RELATION(subject, object): SON(Harry Potter, Albus Severus Potter).

Both the subject and the object can generally be any word or sequence of words that represent concepts that can form relations. In some cases, subjects, objects, or both are limited to entities or named entities.

Named entity is a real-world object, such as a person, location, organization, and products, that can be denoted with a proper name. Named entities can be viewed as instances (e.g., New York City is an instance of a city) of some concept – **class**. Sometimes, numeric data is considered in this category as well (for example, by Named Entity Recognition tools). An **entity** is a named entity whose proper name is unknown or unimportant but still is an instance. (The word book can represent an abstract concept - a class - as well as an entity.)

Relation inventory is the set of relations that are considered valid for a given dataset or model.

Positive relation mention is a sentence that captures a relationship: a relation together with a tagged subject and object. We will omit the word *positive* unless we want to emphasize the fact.

Negative mention is close to a (positive) relation mention in the sense that it is a sentence with tagged subject and object, but the relation type is one of the following types:

- OTHER - human annotator would classify a relation, that is not in the inventory.
- NO RELATION - in this case, human annotators should feel an absence of a relationship between the subject and the object.

NO RELATION comes with difficulties. Since there is no semantic relationship between subject and object, it makes it harder to choose subject-object pairs. It is probably desirable to have subject-object pairs that could be related in a different sentence.

1.2 Relationship Extraction

The relationship extraction task concentrates on the prediction of relationships. In the typical setup, the goal is to predict a relation type based on a sentence with two tagged entities. Variations of this exist, for example, in real-life applications, the goal might be to output a set of relationships based on a longer piece of text (and therefore the extractor would have several sentences mentioning the same entities).

Since the input for a relationship extraction model should contain tagged entities, a pipeline of an entity tagging tool and relationship extraction model would be necessary to perform relationship extraction on real data.

We are aware of two sources of motivation for relationship extraction. First, it can be an alternative to a summarization tool. Therefore it could be used in situations when people are required to read long texts in a short amount of time. In the second application, the extractor could convert texts into structured data and therefore build a graph of entities and their relations. For both applications, an entity linking tool would be beneficial (such a tool disambiguates named entities to a knowledge base) to eliminate the confusion of similarly named entities.

Since many applications are likely to benefit from entity disambiguation, a pipeline of a named entity recognizer, an entity linker and finally a relationship extraction would potentially be useful.

1.3 Czech language

One of the objectives of this thesis is to work with the Czech language. Therefore we find it useful to make some notes on Czech (for non-Czech speaking readers). Czech is a Slavic language with rich morphology and relatively free word order. Most of the Czech morphology can be treated with a morphological analyzer. Still, it might be useful to have a better understanding of the language we are working with.

1.3.1 Inflexion

In Czech, nouns, adjectives, pronouns and numerals are declined. The declination expresses (not necessarily unambiguously) one of seven cases and a number (singular or plural). Any inflected word in Czech has grammatical gender. For words that have natural gender, those two genders nearly always align: “žena” (woman)

singular				
	masculine animate	masculine inanimate	feminine	neuter
nominative	**příčný**		příčná	příčné
genitive	příčného		příčné	příčného
dative	příčnému		příčné	příčnému
accusative	příčného	**příčný**	příčnou	příčné
vocative	**příčný**		příčná	příčné
locative	příčném		příčné	příčném
instrumental	příčným		příčnou	příčným
plural				
	masculine animate	masculine inanimate	feminine	neuter
nominative	příční	příčné		příčná
genitive	příčných			
dative	příčným			
accusative	příčné			příčná
vocative	příční	příčné		příčná
locative	příčných			
instrumental	příčnými			

(a) Declension of “příčný” (diagonal)

	singular	plural
nominative	**ulice**	**ulice**
genitive	**ulice**	ulic
dative	ulici	ulicím
accusative	ulici	**ulice**
vocative	**ulice**	**ulice**
locative	ulici	ulicích
instrumental	ulicí	ulicemi

(b) Declension of “ulice” (street)

Figure 1.1: Examples of czech declension, taken from Wiktionary.

is feminine and “muž” (man) is masculine. The inflexion of each declinable word follows a pattern. This all means that a single word can have relatively many different forms.

Verbs are conjugated, the conjugation expresses person, number, tense, voice, mode and others. Verbs follow one of 14 patterns. An average Czech either finds the theory about Czech verbs and tenses confusing or is even unaware of the existence of the verb patterns. That likely contributes to common use of incorrect forms of verbs even in the official language.

An important aspect of declension for us is agreement. In English, subject and verb agree (limited just to the third person). In Czech, subject and verb also agree, but there also needs to be an agreement in noun phrases.

We include an example to help grasp these concepts to readers who do not speak any inflexive language. In Figure 1.1 words “příčný” (diagonal) and “ulice” (street) are declined, the noun phrase “Příčná ulice” is the Czech equivalent of

the Diagon Alley. The lexeme sizes are 11 and 7, so combinatorically, the noun phrase could have 77 forms. As we explained, in Czech, there is an agreement in noun phrases, and only 9 different forms of the noun phrase "Příčná ulice" are valid.

1.3.2 Word Order

Unlike in English, the sentence structure is relatively relaxed in Czech. The basic sentence structure is of SVO (subject verb object) type, but even when the word order is entirely different, the sentence might still be understandable thanks to the inflected forms of words. At the same time, the word order is not arbitrary; some orderings of words do not form a valid sentence. Those that are valid can carry a considerably different message (with a different emotion or a different emphasis).

The position of attributes tends to be mostly fixed. Some attributes are prepositive, some postpositive, but most of them stay at the same position (within the noun phrase) unless some enumeration is used. If we return to the "Příčná ulice" example, we are not able to recollect a sentence, where the reversed order is used.

Part I

Datasets

2. Existing Datasets

In this chapter, we overview three well-known datasets related to Entity Relationship Extraction. We start with supervised datasets (SEMEVAL 2010 task 8 and TACRED), then we focus on distant supervision.

2.1 SEMEVAL 2010 Task 8 Dataset

The SemEval-2010 Task 8 dataset (S10T8) was introduced in SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals [Hendrickx et al., 2010]. We summarize how S10T8 was created and present additional information from the article, so that we can compare it later with other datasets.

The authors started by choosing an inventory of semantic relations. They aimed for such a set of relations that is exhaustive (enables the description of relations between any pair of nominals) and mutually exclusive (given the context and the pair of nominals, only one relation should be selectable). Chosen relations with descriptions and examples are listed in Table 2.1.

They decided to accept as relation arguments any noun phrases with common-noun heads, not just for example named entities mentioning: "Named entities are a specific category of nominal expressions best dealt with using techniques which do not apply to common nouns." They restricted noun phrases to single words with the exception to lexicalized terms (such as science fiction).

The annotation process consisted of three rounds. In the first round, authors manually collected around 1 200 sentences for each relation through pattern-based Web search (with at least a hundred patterns per relation). This way, they obtained around 1 200 sentences for each relation. In the second round, each sentence was annotated by two independent annotators. In the third round, disagreements were resolved, and the dataset was finalized. Every sentence was classified either as a true relation mention or was a near-miss. The near-miss sentences were classified as OTHER, or were removed.

The relations inventory constains nine positive relations and one positive. The authors decided to include the directionality into the relation a therefore the inventory size is $9 \cdot 2 + 1$ in total.

The dataset contains 10 717 relation mentions. For the original competition, teams were given three training dataset of sizes 1 000 (TD1), 2 000 (TD2), 4 000 (TD3), and 8 000 (TD4). Since there was a notable gain TD3 →TD4, the authors concluded that even larger dataset might be helpful to increase the performance of models. On the topic, the creators have written:

> .. that is so much easier said than done: it took the organizers well in excess of 1 000 person-hours to pin down the problem, hone the guidelines and relation definitions, construct sufficient amounts of trustworthy training data, and run the task.

Table 2.1: S10T8 summary. List of relations, their official descriptions, a random relation mention and both the relative and the absolute count of mentions when the directionality is ignored.

CAUSE-EFFECT	12.4%
An event or object leads to an effect.	(1 331)
Example: *The burst has been caused by water hammer pressure.*	
INSTRUMENT-AGENCY	6.2%
An agent uses an instrument.	(660)
Example: *The author of a keygen uses a disassembler to look at the raw assembly code.*	
PRODUCT-PRODUCER	8.8%
A producer causes a product to exist.	(948)
Example: *The factory's products have included flower pots, Finnish rooster-whistles, pans, trays, tea pots, ash trays and air moisturisers.*	
CONTENT-CONTAINER	6.8%
An object is physically stored in a delineated area of space.	(732)
Example: *This cut blue and white striped cotton dress with red bands on the bodice was in a trunk of vintage Barbie clothing.*	
ENTITY-ORIGIN	9.1%
An entity is coming or is derived from an origin (e.g., position or material).	(974)
Example: *The avalanches originated in an extensive mass of rock that had previously been hydrothermally altered in large part to clay.*	
ENTITY-DESTINATION	10.6%
An entity is moving towards a destination.	(1 137)
Example: *This book has transported readers into ancient times.*	
COMPONENT-WHOLE	11.7%
An object is a component of a larger whole.	(1 253)
Example: *The system as described above has its greatest application in an arrayed configuration of antenna elements.*	
MEMBER-COLLECTION	8.6%
A member forms a nonfunctional part of a collection	(923)
Example: *The student association is the voice of the undergraduate student population of the State University of New York at Buffalo.*	
MESSAGE-TOPIC	8.4%
A message, written or spoken, is about a topic.	(895)
Example: *Cieply's story makes a compelling point about modern-day studio economics.*	
OTHER	17.4%
	(1 864)
Example: *The child was carefully wrapped and bound into the cradle by means of a cord.*	

2.2 TACRED dataset

The TAC Relation Extraction Dataset was introduced in [Zhang et al., 2017b]. TACRED is a supervised dataset obtained via crowdsourcing. It contains about 100 000 examples, which makes it about ten times bigger than S10T8 dataset.

The authors are relatively brief about the data collection process:

> We create TACRED based on query entities and annotated system responses in the yearly TAC KBP evaluations. ... We make use of Mechanical Turk to annotate each sentence in the source corpus that contains one of these query entities. For each sentence, we ask crowd workers to annotate both the subject and object entity spans and the relation types.

TACRED relation inventory captures 41 relations with the subject being an organization or a person; plus a negative relation. Objects are of the following types: cause of death, city, country, criminal charge, date, duration, ideology, location, misc (used for alternative name relation and NO RELATION only), nationality, number, organization, person, religion, state or province, title and URL. The choice of subjects and objects it therefore very different from the S10T8 dataset.

TACRED was designed to be highly unbalanced. 79.5% of mentions represents the NO RELATION relation. This ratio of negative relation should be closer to real-world text and supposedly should help avoid false-positive predictions. However, even if we look only at positive relations, there are vast differences in frequency: the top six relations make up half the dataset and the bottom six less than 2%. In absolute numbers, the least common ORD:DISSOLVED relation has only 33 mentions, and the median is only 286 mentions.

Table 2.2: TACRED summary. List of relations, a random example, and both the relative and the absolute count. The table is restricted to ORG:* relations.

NO_RELATION Example: " One step at a time , " said Con Edison spokesman Chris Olert in Sunday editions of The Daily News .	79.5% (84 491)
ORG:ALTERNATE_NAMES Example: The ARMM was established as a result of the peace agreement between the government and the Moro National Liberation Front -LRB- MNLF -RRB- in 1996 .	1.3% (1 359)
ORG:CITY_OF_HEADQUARTERS Example: Once completed , the cuts will leave the Irvine , California-based Option One subsidiary with about 1,400 employees .	0.5% (573)
ORG:COUNTRY_OF_HEADQUARTERS Example: The Review based its report on a new survey conducted by the International Agency for Research on Cancer in Lyon , France .	0.7% (753)
ORG:DISSOLVED Example: News Corp. sold its satellite television service DirecTV in 2008 to Liberty Media .	0.0% (33)
ORG:FOUNDED Example: New York-based Zirh was founded in 1995 and makes products using natural oils and extracts .	0.2% (166)

ORG:FOUNDED_BY Example: The Jerusalem Foundation , a charity founded by Kollek 40 years ago , said he died of natural causes Tuesday morning .	0.3% (268)
ORG:MEMBER_OF Example: Lyons and the Red Sox say they are n't aware of any other Major League Baseball team with such an arrangement .	0.2% (171)
ORG:MEMBERS Example: The NFL refused to abandon the city , and the Saints won the NFC South in 2006 , their first season with Brees and Payton .	0.3% (286)
ORG:NUMBER_OF_EMPLOYEES/MEMBERS Example: Established in September 1969 , the organization now has 57 member states worldwide .	0.1% (121)
ORG:PARENTS Example: The initial offering of AIA raised $ 178 billion for AIG , while the sale of ALICO to MetLife reaped about $ 155 billion .	0.4% (444)
ORG:POLITICAL/RELIGIOUS_AFFILIATION Example: Manila signed a peace treaty with the MNLF in 1996 , ending a decades-old separatist campaign in return for limited Muslim self-rule .	0.1% (125)
ORG:SHAREHOLDERS Example: Stop the NAACP and Al Sharpton 's National Action Network from committing this disgrace in our community .	0.1% (144)
ORG:STATEORPROVINCE_OF_HEADQUARTERS Example: Romney 's investments are in Novo Nordisk , a Danish company , and Millipore Corp. , based in Billerica , Mass. .	0.3% (350)
ORG:SUBSIDIARIES Example: Moody 's said it has growing concerns about the control wielded by Johnson at Fidelity 's parent company , FMR LLC .	0.4% (453)
ORG:TOP_MEMBERS/EMPLOYEES Example: Frank Gut , the chief financial officer , previously oversaw the Swiss brokerage operations at Bank Julius Baer in Zurich .	2.6% (2 770)
ORG:WEBSITE Example: Swiss Bankers Association : http://www.swissbanking.org	0.2% (223)

2.3 Riedel NYT dataset

The previous two datasets were obtained through a tedious human labour – human annotators went through texts and manually annotated the data. This process is slow and expensive, which explains the relatively small data volume of the datasets. In this section, we introduce a dataset presented in [Riedel et al., 2010] that was created without the need for any additional manual annotation.

This dataset was generated with the distant supervision approach. This approach is based on aligning structured data (knowledge base) with text, i.e. automatically tagging mentions of the structured data in text. In distant supervision, we usually expect that if there are two entity mentions in a sentence that are related, then the sentence expresses their relationship. The authors acknowledge that this assumption is often violated and they propose a methodology that attempts to predict whether the assumption is violated in a sentence. Using this methodology, they generated a dataset from the The New York Times Annotated Corpus [Consortium and Company, 2008] and Freebase [Bollacker et al., 2008].

Table 2.3: Random 10 relations from the Riedel NYT dataset. For each relation the name, the relative and the absolute number of mentions and a random mention is shown.

/LOCATION/LOCATION/CONTAINS Example: *But John Traugott , 68 , a hospital chaplain in Rockaway Park , Queens hinted at some of Chinatown 's problems .*	8.6% (58 625)
/PEOPLE/PERSON/NATIONALITY Example: *We were unable to reach agreement , " Foreign Minister Frank-Walter Steinmeier of Germany announced tersely to reporters .*	1.5% (10 464)
/PEOPLE/PERSON/PLACE_LIVED Example: *Camane -LRB- Emily Blunt -RRB- runs around Rome putting up signs declaring that Octavius is Julius Caesar 's rightful heir .*	1.2% (8 275)
/BUSINESS/COMPANY/FOUNDERS Example: *Nick Grouf , president and chief executive at Spot Runner in Los Angeles , is scheduled to announce the investments today .*	0.1% (999)
/PEOPLE/DECEASED_PERSON/PLACE_OF_DEATH Example: *Marie Antoinette finally did arrive in Paris , at Christian Dior , where the designer John Galliano proclaimed her his platinum muse .*	0.3% (2 190)
/BUSINESS/PERSON/COMPANY Example: *Christopher Bailey , with his light-handed take on Burberry 's heritage , could add some military backbone and useful outerwear .*	0.9% (6 455)
/LOCATION/US_COUNTY/COUNTY_SEAT Example: *Mr. Perhacs was taken to Jersey City Medical Center , where he died , said Edward J. De Fazio , the Hudson County prosecutor .*	0.0% (125)
/BUSINESS/COMPANY/PLACE_FOUNDED Example: *Next month in Paris , Ms. Tilbury will direct makeup at the spring fashion shows of Lanvin , Chloé and Alexander McQueen .*	0.1% (537)
/PEOPLE/PERSON/PLACE_OF_BIRTH Example: *Preston Robert Tisch was born in the Bensonhurst section of Brooklyn on April 29 , 1926 , to parents who came from Russia .*	0.5% (3 603)
/FILM/FILM/FEATURED_FILM_LOCATIONS Example: *Half Nelson , " a new independent film about an idealistic young Brooklyn teacher , takes this claim at face value . "*	0.0% (19)

We will refer to this dataset as Riedel NYT.

Relation inventory for the usual training part of the dataset contains 58 relations. The best represented is the NA relation with over 80%. The representation of relations varies between the train and test set. For example, two relations are present only in the test set.

3. CERED

In the previous chapter, we concluded that creating relationship extraction dataset manually is expensive. We introduced Riedel NYT dataset that shows, that automatic generation of relationship extraction datasets is possible. In this chapter we describe our process of generating **Cz**ech **R**elationship **E**xtraction **D**ataset (CERED). We will discuss various decisions that we made during this process and their impacts.

3.1 Overview

The objective is to use distant supervision to create a relationship extraction dataset for the Czech language. This section is a brief summary for easier orientation in this chapter. Each of these paragraphs is a teaser for one section of this chapter.

First we research available knowledge bases and Czech text corpora to determine which ones will best suit our purpose. We chose Wikimedia projects Wikidata and Czech Wikipedia.

Next we analyze how we will find mentions of Wikidata relations in Czech Wikipedia. We sketch out dataflow diagrams and we discuss all the different complex aspects of this task.

We continue by choosing technologies that we use. Aware of the volume and other characteristics of chosen data, we choose Python as the main programming language, Spark as a way to speed up the computations and MorphoDita to deal with the specifics of the Czech language.

The questions and options that rose from the analysis get at least partially answered and decided during implementation. We tested different configurations and went through the data to determine which works the best. As a result, we generated CERED0.

The generated dataset has a huge relation inventory and other properties that would be impractical for training. We describe variations of the generated dataset CERED1-4, which were designed to be more practical.

3.2 Data Sources

To be able to perform distant supervision we need to find suitable data - Czech text corpus and a knowledge base (Figure 3.1). In the first subsection, we will explain the requirements and constraints we have on such data and present our options. In the next two subsections, we will provide more information on the selected ones.

3.2.1 Constraints and Requirements

The main constraint is quite straightforward, there has to be a nontrivial shared set of entities and relations mentioned in the text and stored in the knowledge base. We expect fact-based text to be more suitable then fiction literature. Therefore we prefer encyclopedic or journalistic genre. One option is to focus on some

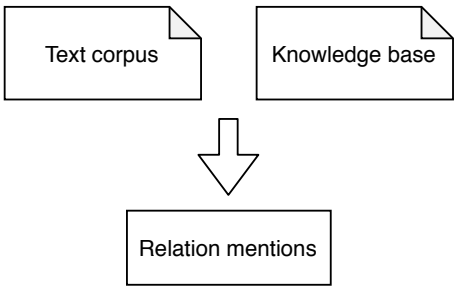


Figure 3.1: Distant supervision diagram

subset of Czech National Corpus,[1] for example SYN2013PUB, SYN2009PUB, and SYN2009PUB are corpora of written journalism. The other option is to lean in the direction of encyclopedic text with Czech Wikipedia.

To the best of our knowledge, our options for knowledge base are limited to Wikidata or Google Knowledge graph.[2]

We decided to use Czech Wikipedia and Wikidata, mostly because the intersection of information expressed in text data and in structured data seems promising because they are build on each other. Another advantage is the multilingualism of Wikimedia projects, and therefore the transferability of this work is higher. Moreover, the Wikimedia projects are downloadable, which makes them easier to work with.

3.2.2 Czech Wikipedia

Wikipedia is a multilingual online encyclopedia created and maintained as an open collaboration project by a community of volunteers as defined in Wikipedia contributors [2020]. From our point of view, Wikipedia is a corpus of text with tagged topics of articles and some entity mentions. Czech Wikipedia contains approximately 440 000 articles and ranks top 30 across all the different language editions of Wikipedia.[3]

A dump of Czech Wikipedia is about 1,6GB and 770MB when compressed.

3.2.3 Wikidata

Wikidata is a knowledge base which acts as a central storage of the structured data of Wikimedia projects. Just like Wikipedia, this project is freely available and edited by users (and bots). It provides the option to query the database online (for small enough queries), but it is also possible to download the database in standard formats.

The database focuses on **items**, which represent objects, entities, concepts, etc. The first data collected in Wikidata were links to the multilingual versions of Wikipedia articles on the same topic – on the same Wikidata item. Each item is assigned an identifier, prefix Q and a unique number, referred to as **QID**. A label together with a description of an item should serve as a human readable identifier. Labels, descriptions and optional aliases are language dependent.

[1] `https://www.korpus.cz/`

[2] `https://developers.google.com/knowledge-graph`

[3] As of March 2020 according to `https://en.wikipedia.org/wiki/List_of_Wikipedias`

Properties, another big concept of Wikidata, can be thought of as categories of items (*mother P25* implies a category of all mothers) or as relations between items (*Ron Weasley Q173998* has a *mother P25 Molly Weasley Q3255012*). Each property has its **PID**, an identifier consisting of a prefix P and a unique number, and a data type for a value it can be paired with (such as an item, string, url, number or media file).

Information about any item is recorded in statements. Statement is a key-value pair of a property and a value of prescribed data type. For example, for *Ron Weasley Q173998* there are six statements about his siblings:

- *sibling P3373*: *Ginny Weasley Q187923*,
- *sibling P3373*: *Fred Weasley Q13359612*,
- *sibling P3373*: *George Weasley Q13359613* and so on.

Wikidata project contains over 80 000 000 items, which raises requirements on technological resources required to work efficiently with such data. JSON dump of Wikidata takes 110GB of disk space or 37GB if bzip2 compressed.

3.3 Analysis

The process of CERED creation is mostly an attempt to execute the first two parts of the pipeline we mention in Section 1.2. To the best of our knowledge, there is no suitable entity linking tool for Czech. There are tools for named entity recognition that we could theoretically use to our advantage, if we decided to focus on named entities only.

Therefore, we need to find a way to get to similar results as the first stages of the pipeline. We do not expect that our CERED generator will be as powerful as the respective dedicated tools would be. We do not try to create general entity recognition and linking tools – on the contrary, we exploit all suitable extra information that chosen Wikimedia projects provide.

There are several aspects that we need to think through. We introduce them in the following list for better orientation, and then elaborate every one in more detail in dedicated subsection.

- **Dataflow** – we chose Wikidata and Czech Wikipedia, but we did not discuss how to connect them and what exactly should be the outcome so that we can proceed to locating mentions.
- **Entity Matching** – suppose we collected a piece of text together with a set of entities that could be mentioned in the text. The process of entity matching attempts to mark words in the text that mention an entity.
- **Wikilink Mentions** – Wikilinks are (mostly) human labeled entity mentions. Utilizing them is the closest we can get to a supervised dataset without actually supervising the dataset.
- **Relation Matching** – after entities are matched in the Wikipedia texts, relationships are extracted from the Wikidata dump and we use distant supervision assumption to locate relation mentions in the texts.

- **Relation Inventory** – we generated CERED, but the relation inventory is overly diverse. Moreover, the dataset is extremely unbalanced – the number of mentions per relation varies – and we lack negative relation.
- **Result Evaluation** – every time we generate CERED during development, we need to evaluate its quality. We propose methods for this evaluation.

3.3.1 Dataflow

We start the whole process of creating our dataset with two files. The first file is a Czech Wikipedia dump. It is a collection of articles where each article has its title, id, and text. And the other file is a Wikidata dump.

The simplest way of processing those files would be to process them separately and thus obtaining sentences on one side and relationships (a relation type with two items) on the other. This approach comes with a clear disadvantage. We would lose any additional information about the sentences that could be potentially useful (for example article title might be helpful to determine which items are mentioned in the sentence).

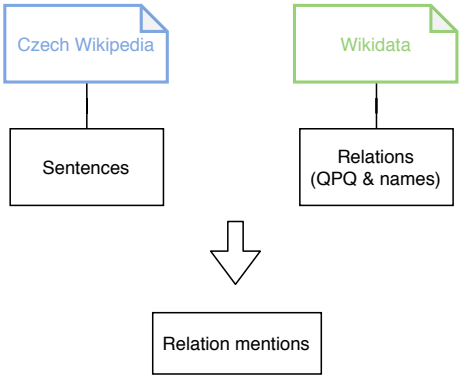
To prevent the loss of information, we could precompute additional data for each article and attach it to each sentence (article title, all Wikilinks in the article, etc.), risking a massive increase in required capacity to work with such data. On a similar note, we could process Wikidata to store item names (labels and aliases) for each relationship, worsening the situation even further.

Popisky u obrázku informed and uninformed

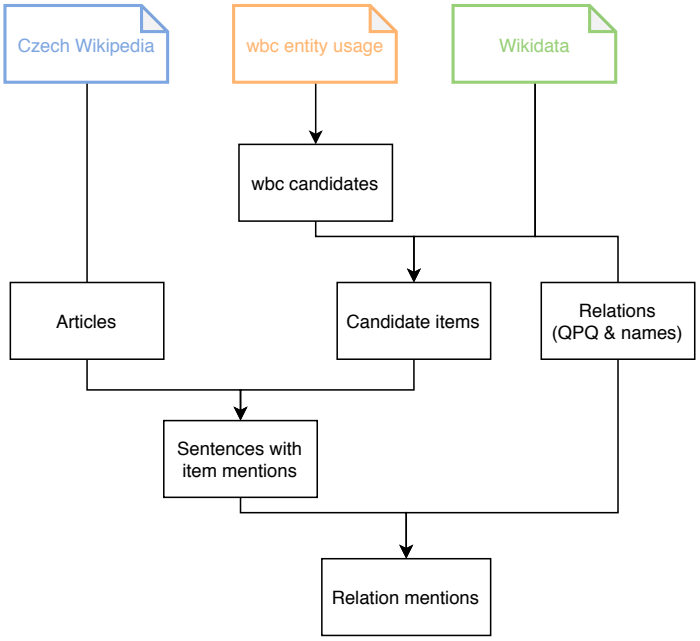
We decided to update the dataflow to address those issues. We preprocess the Wikidata dump to contain only the data we use. We refer to this processed version of Wikidata as **custom Wikidata**. An item is kept only if it has a Czech name and we significantly reduce its statements: we keep the title of its Czech Wikipedia article and create a list of (QID,PID,QID) triples, **QPQ**, representing statements that contain information about relations between this item and other items. This way, we have all the necessary information – article title to be able to connect an article to an item, names for each item to be able to find mentions of items, and finally QPQ triples to connect relations and sentences. Moreover, custom Wikidata size is closer to traditional RAM size. Therefore we could for example load item names into memory, which comes in handy during the implementation.

One approach to finding item mentions in text could be called uninformed (Figure 3.2a). We could assume that any item can be mentioned in any sentence. This approach has two issues: the computation would likely take quite some time, but mainly we expect a huge amount of ambiguous mentions. An example of this ambiguity, which we see as problematic, might be children named after their parents. In this case, not only that the entities might get confused, but also if we then assign the relation, we might easily confuse a sentence mentioning a spouse relation for a parent relation, which, unfortunately, is extremely challenging to solve.

On the other side, we can use the extra information that Wikimedia projects provide and opt for a more informed approach. A diagram of this approach is captured in Figure 3.2b. The topic of most Czech Wikipedia articles is a Wikidata item, therefore this item is nearly certainly mentioned in the article. Some Wikidata statements were based on relevant articles and thus it seems



(a) Uninformed approach of locating mention relations in text



(b) Informed approach of locating mention relations in text

plausible to expect that items related to the main item of an article are mentioned in it. We, therefore, decided to look only for a tiny subset of all Wikidata items in each article, the so called **candidate items**, which consist of the article item and all times connected to it by a statement.

Czech Wikipedia maintains a `wbc_entity_usage` table, which contains information about which article uses which item. If we use this table, we are able to obtain a list of items, that should be mentioned in an article, let us call this list a **wbc candidates**. A wbc candidate is at the same time a candidate item.

We might consider adding even a second level of relatives (items related to items that are related to the main item), but the branching factor might be relatively high and cause unwanted ambiguity. Consider an instance item like a specific country, all countries would be second level relatives and thus candidate items. Since countries tend to be of a certain type (kingdom, republic, state etc.), there might be simply the type or some other more general name amongst their names (*United States of America Q30* are also known as America or United States) and more countries might share this name.

So far we mostly discussed the advantages of the proposed informed approach, mainly a hope for higher precision, specifically higher precision for item mentions. We should elaborate on some disadvantages as well. We are not trying perform fully do entity linking. In the end we will only use item mentions, if the following condition holds: there are two entity mentions in one sentence and there exists a QPQ that connects them. It is questionable whether we need an informed approach to increase relation mention precision. The improbability that this condition is fulfilled for false-positive item mentions might in fact be sufficient.

One more way to locate item mentions is through **Wikilinks**. A Wikilink links a page to another page within the same-language Wikipedia. First additional information this brings is simply the item mention (if the linked page or article has its main item). We can also consider the textual part of the link to be another name for the linked item. The quality and suitability of this name are to be examined and if we find these names useful, they can be added to the item names we use.

3.3.2 Entity Matching

We have text on one side, gathered candidate items on the other side and our goal is to find occurrences of these items in the text. We call this process **entity matching** and each found occurrence is an **entity mention**.

No matter how the matching is done, it seems always beneficial to start the process with some text preprocessing. Quite a lot of changes need to happen even if some seem like little details. We separate this preprocessing into a wiki specific part, lexical analysis, and the last part is devoted to lexical analysis on standalone noun phrases.

When we eventually proceed to entity matching, there is a wide spectrum of complexity we might aim for. Bearing in mind that we are no trying to create a strong sophisticated tool for entity recognition and linking., we describe some of those levels of complexity and choose the right method for our use case.

Wikipedia parsing

Wikipedia parsing starts with an article in Wikitext and produces human-readable plain text – a **clean text**. Note that we should keep track of positions of Wikilinks from the Wikitext in the clean text.

Wikipedia is written in Wikitext (Wiki markup, Wikicode). This markup provides all usual functionalities such as determining the layout or fonts, and enables commonly-used concepts like lists, links, media file insertion, or tables, and some more wiki specific concepts like infoboxes.

We plan on using one or even a combination of existing Wikitext parsers, since each of them provides different functions.[4] Therefore the parsing itself is not too troublesome.

One problem that needs to be addressed is what should we consider a valid text. For example, it is not clear how to work with tables. From one point of view, if we convert a table into an unstructured text, it will not be a regular text in terms of sentence structure. From a different point of view, an unstructured text obtained by converting a table still contains information that human readers will likely decode. Moreover, tables and other structured data tend to contain a lot of information. This will likely cause problems, because we want to concentrate on sentence-like data. For example, pairs of persons and countries might be matched from sport results tables. This kind of data might significantly damage the quality of CERED.

The elimination of all Wikipedia content that is too structured or generally not enough sentence-like, but at the same keeping as much as possible, is addressed later in Subsection 3.5.2. That way we can see the consequences of the eliminated and kept content.

Lexical Analysis

Keeping text (such as an article) in long sequences of characters is not the best format for our purpose. Instead we need to parse those sequences into smaller units, such as words and sentences. A tool that addresses such tasks is usually called a **tokenizer** and a **token** is a term generalizing the term word. Often words in text are the same as tokens, but "aren't" in English and "mohu-li" in Czech are likely considered one word but multiple tokens. On the contrary, "M*A*S*H" or "email@email.com" might not be considered a word, but should be considered exactly one token each. Naive tokenizer might just simply split on non-alphanumeric characters,but for better performance, a more sophisticated tool is needed.

Since we work with Czech, which is a language with rich morphology, we might want to know non-inflexed form of a token, moreover we might want to assign some unique idintifier to such form in case it has some homographs (words that are spelled the same way). Such am identifier is called a **lemma** and a collection of tokens that share the same lemma is called a **lexeme**. We outsource handling text to a Czech tokenizer and tagger called MorphoDiTa [Straková et al., 2014], that achieves state-of-the-art results for the Czech language. Using such a tool, we can convert clean text into sentences made up of tokens and we even obtain the lemma and lexeme of each token.

[4] `https://www.mediawiki.org/wiki/Alternative_parsers`

Lexical Analysis on Names

Tokenizers (and lemmatizers) are usually trained to perform well on sentences and might be inaccurate on noun phrases when they stand alone, as entity names do. If we were determined to tokenize them, a simple trick like constructing a sentence with the name in it and tokenizing this sentence can partially solve this problem. Such a sentence that would be grammatically correct and not semantically terrible could be something like "This is /name/", but realistically, this sentence was quite likely not at all common in the training process of MorphoDiTa. Some foreign words can be erroneous as well and keeping their original form might be the only easy way around it.

Matching methods

Entity matching can be performed with various degrees of sophistication. We provide a short overview of four such degrees. The degrees 1 and 2 do not require any knowledge of the language they work with, while 3 and 4 are language dependent. For simplicity, we assume that we are only looking for mentions in one sentence at a time, unless written otherwise. We also include an example of such matchings to demonstrate how successful we expect them to be.

1 String equality. This method is the easiest method of entity matching. It is based on a simple substring check, which is later extended with additional functionality. In more detail, for each entity, we have multiple name variants and for each of those names, we check whether the name is a subsequence of the sentence tokens.

We also need to consider the letter cases. Named entities should have fixed letter cases and no additional processing is needed in most cases. In other cases, an established name for a named entity might be written with the lowercased first letter – *Weasley family Q716534* has Czech names "Weasleyovi" (the Weasleys) and also "rodina Weasleyových" (Weasley family). If we consider entities that are commonly written with lower case, the sentence now needs to be preprocessed so that for example the first letter is not capital. Moreover, there is no guarantee that common names will be lowercased in Wikidata. To conclude, nearly nothing can be assumed about the case of letters, and therefore one of the following solutions needs to be implemented: we can convert everything to one chosen letter case or we need to perform some more sophisticated attempts at predicting, which words can have more versions in terms of letter cases.

Another problem that we encounter is how to properly handle spaces. We list some troublesome examples and accept the fact that not everything can be done perfectly. J. K. Rowling has J.K.Rownling as one of Wikidata names, confirming that both versions might appear in written text, but not all entities with similar name type have all space-variants listed in names. We, therefore, assume that spacing around the "-" character might vary.

It is also not clear if word order in entity names is fixed (or at least almost always fixed). Even a simple reversion in name will affect the performance of this method (J. K. Rowling and Rowling, J. K.).

The greatest weakness of this method is its inability to recognize entities if their name is inflected. To emphasize how many words are not equal to their lemma in Czech text, we colored them in a sample text presented in Figure 3.3.

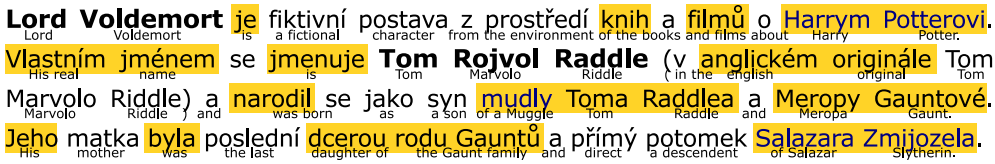


Figure 3.3: First paragraph from Czech Wikipedia page about *Lord Voldemort Q176132*, also known as "Tom Rojvol Raddle" (Tom Marvolo Riddle), "Voldemort" and "Pán zla". Highlighted words are not in their base forms.

We elaborated on Czech language in Section 1.3, but just for simplicity – in English the verb "to be" has many different forms (am, are, were, was, would and so on), and in Czech all nouns and verbs behave like this, quite often with many more forms.

To illustrate the effect of the described method, Figure 3.4 displays the resulting entity mentions on a sample text.

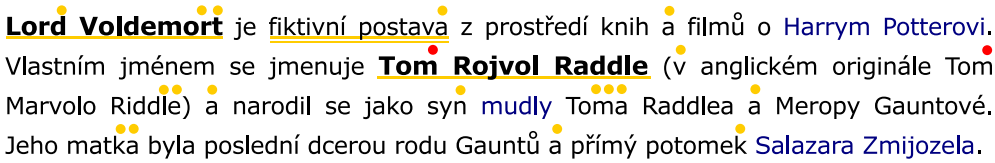


Figure 3.4: Results of entity matching using string equality on the first paragraph from Czech Wikipedia page about *Lord Voldemort Q176132*, also known as "Tom Rojvol Raddle" (Tom Marvolo Riddle), "Voldemort" and "Pán zla". Dots represent found mentions on a single word, red stands for five found mentions, lines represent found mentions of noun phrases (five Wikidata entities match "Tom", and one Wikidata entity matches the whole "Tom Rojvol Raddle").

2 String similarity (approximate string matching). String similarity is still based on simple string manipulation, no vocabulary or other language knowledge is necessary. The goal is to find entity mention, even if its name is a little altered in the sentence. This alternation can include all of the issues listed for the previous method – cases of letters, spacing, word order, and word forms, but even better, it might help in cases that we did not anticipate.

There are many metrics describing string similarity. Some can cope better with word order issues, some with word forms, some with spacing. We do not test all of them for our usecase, but still find it useful to mention them, since in other than the Czech language, some might work well.

First category of string similarity metric is based on **edit distance**. Levenshtein distance is the minimum number of edits (additions, deletions, and substitutions of a character) to get from one string to the other. As a metric the ratio of Levenshtein distance and of the sum of the lengths of the strings can be used. This metric, unsurprisingly, handles well the mentions that are close in the amount of edits needed, so mentions with different inflection, spacing, or letter casing will likely be considered a match.

Damerau–Levenshtein distance is very similar to the previous, but a transposition of two adjacent characters is also considered an edit. We might argue that some Czech words tend to transpose the last characters in different word

forms and thus this metric could work better for those forms, but there might be a higher risk of false positives.

Another category is **based on tokens**. The tokenization converts both the sentence string and the name string into a sequence of tokens. If we consider those sequences to be sets (S, N respectively), metrics that work with sets can be utilized. For example, Jaccard index (intersection over union) is computed as $|S \cap N|/|S \cup N|$. Any other set similarity measure can be used.

Set based metrics ignore the order of tokens and therefore could solve issues with mentions in which the word order is not the same as in the name. On the other side, an increase in false positives is to be expected, and some additional post-processing is needed to determine which token in the sentence should be considered a mention, if the token was used in the sentence multiple times.

3 Morphological analysis. Moving on from methods that are mostly unaware of the language they work with, we will finally use the **morphological analysis** we mentioned earlier.

With lemmas of both the sentence and the name, we can use any metric from the previous subsection on string similarity (joining the lemmas using a space if the metric expects only two input strings).

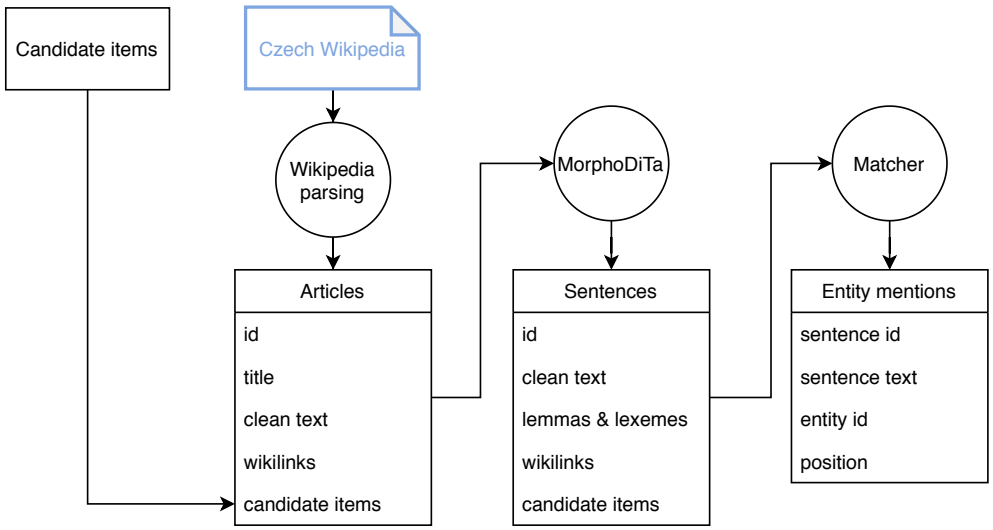


Figure 3.5: Matching with morphological analysis

If we decide to keep the entity names in their original form (tokenization on them can be error prone, as we already explained), we can try to use the correct form of the tokens in the sentence. For each word in the sentence, we use its lexeme to determine if (part of) a entity name matches the word.

Detailed diagram of matching entities using morphological analysis is shown in Figure 3.5 and the obtained results on the sample text are captured in Figure 3.6.

4 Advanced concepts. Proper entity matching (either in named entity recognition or entity linking) might be expected to recognize entity mention even if the entity is not mentioned explicitly by its name. Pronouns should be assigned an entity they represent (if they do) and other nouns as well. In languages like Czech, where the subject of a sentence is often omitted, the entity mention is even less obvious, but still present. Since the topic of this thesis is not entity

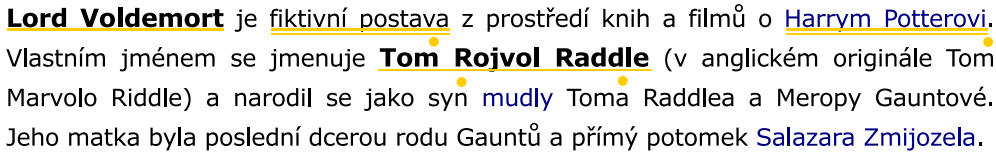


Figure 3.6: Results of entity matching using morphological analysis and informed approach on the first paragraph from Czech Wikipedia page about *Lord Voldemort Q176132*, also known as "Tom Rojvol Raddle" (Tom Marvolo Riddle), "Voldemort" and "Pán zla". Dots represent found mentions on a single word, lines represent found mentions of noun phrases (one candidate entity matches "Tom", and one one candidate entity matches the whole "Tom Rojvol Raddle").

matching, we will not debate techniques to achieve this level of matching neither will we implement them.

obrázky jsou špatně, zase

Conclusion

After looking at the results, we decided to use a simple metric based on morphological analysis – a detailed description of the matching method CERED was generated with is described in Subsection 3.5.3. We still find it useful to keep this summarization of different string metrics as part of this thesis.

3.3.3 Wikilink Mentions

As we already mentioned, from our point of view, Wikilinks are entity mentions created by Wikipedia editors. The text part of the link can in theory be anything, providing us with some more advanced examples of entity linking that our matching methods cannot perform.

Since this data is not fully supervised and the word supervised is overused (semi-supervised, distant-supervised, etc.) we decided to call it **silver**, because they are not of the optimal quality that is usually called gold, but they are the best we can get.

3.3.4 Relation Matching

If a sentence contains two entity mentions that are related, chances are that the sentence in fact does express their relationship and thus is a relation mention. This concept is called **distant supervision assumption** and can be also formulated in the following way: If two entities participate in a relation, all sentences that mention these two entities express that relation. This assumption is commonly used, even though it is clearly not correct, because it is easy to use.

To evaluate, how often this assumption is violated, is labour-intensive, but luckily is has already been performed. In Riedel et al. [2010], the distant supervision assumption is compared to the **express-at-least-once assumption**, which states that if two entities participate in a relation, at least one sentence that mentions these two entities might express that relation.

The authors sampled 600 relation mentions from two corpora, both created by distant supervision on Freebase (knowledge base commonly used before Wikidata

took over) and two text corpora - Wikipedia articles and the New York Times corpus. These 600 samples represented three different relation types (NATIONALITY, PLACE OF BIRTH, and CONTAINS) and were sampled so that there were 100 samples of each type in each corpus. We include their results in Table 3.1. They concluded that the distant supervision assumption holds more often in Wikipedia because Wikipedia is a very specific type of text corpora, where articles are centered around entities. We believe that the reasoning can be extended with the fact, that Freebase contained information from Wikipedia infoboxes, and those infoboxes were created based on the textual information.

	NYT	Wikipedia
NATIONALITY	38%	20%
PLACE OF BIRTH	35%	20%
CONTAINS	20%	10%

Table 3.1: Percentage of times a related pair of entities is mentioned in the same sentence, but where the sentence does not express the corresponding relation. Taken from Riedel et al. [2010].

For the authors the results signalized that a more sophisticated tool is needed, instead of relying on the distant supervision assumption. We acknowledge that such a tool is needed, but at the same time, we believe that in our case, where we create CERED based on Wikipedia and Wikidata, the precision they estimated is sufficient. We also assume, that Wikidata project is more suitable for this task than was Freebase.

We want to mention, that we build CERED to easily fit into the modern deep learning models and to be as simple as possible. Therefore, the main unit of text we use is a sentence, which is intuitive, but it has one downfall – the resulting relationship extractor cannot detect relations expressed over sentence boundaries.

3.3.5 Relation Inventory

In Chapter 2 three examples of relationship extraction datasets were introduced. The creators of those datasets claim, that in the creation process, they first decided on the relation inventory (relation types). Creating the relation inventory seems to be the straightforward and rational approach, and we wanted to create such inventory before actually implementing the CERED generator. However, we stumbled upon the following issues.

Huge Wikidata Relation Inventory

Wikidata relation inventory (properties in Wikidata terminology) is an order of magnitude larger compared to the traditional relationship extraction datasets and handpicking our inventory is overwhelming. We even considered reducing the size of this inventory by creating our own relations that would combine Wikidata relations (parent would be a combination of mother and father relations).

Given the excessive number of Wikidata relation types, we decided to considerably reduce them. However, to not omit the best-represented relations, we constructed the relation inventory only after the creation of CERED.

Absence of a Negative Relation

Knowledge bases, in general, do not contain negative relations (relations that could be easily mapped to the NO RELATION or OTHER RELATION), but for relationship extraction, negative mentions are essential. If we generate mentions using all properties, we can later decide which relationships will be in the inventory and the rest of them relabel to OTHER RELATION. If we were to assign all tuples of entity mentions that share a sentence and are not related as NO RELATION, we could increase the noise in CERED, because not all relationships are in Wikidata and therefore some of the NO RELATION mentions could in fact be positive mentions. The ratio of negative and positive mentions in the two bigger datasets we introduced in Chapter 2 were approximately 80%.

While curating the inventory, we should keep in mind, that we are not just choosing the relations, but also their representations, and we need to attempt to fulfil the three following requirements to the best of our abilities:

- Each relation needs to be represented enough.
- The more balanced relation representation the better.
- There should be enough negative mentions and their negativity should be assured.

3.3.6 Result Evaluation

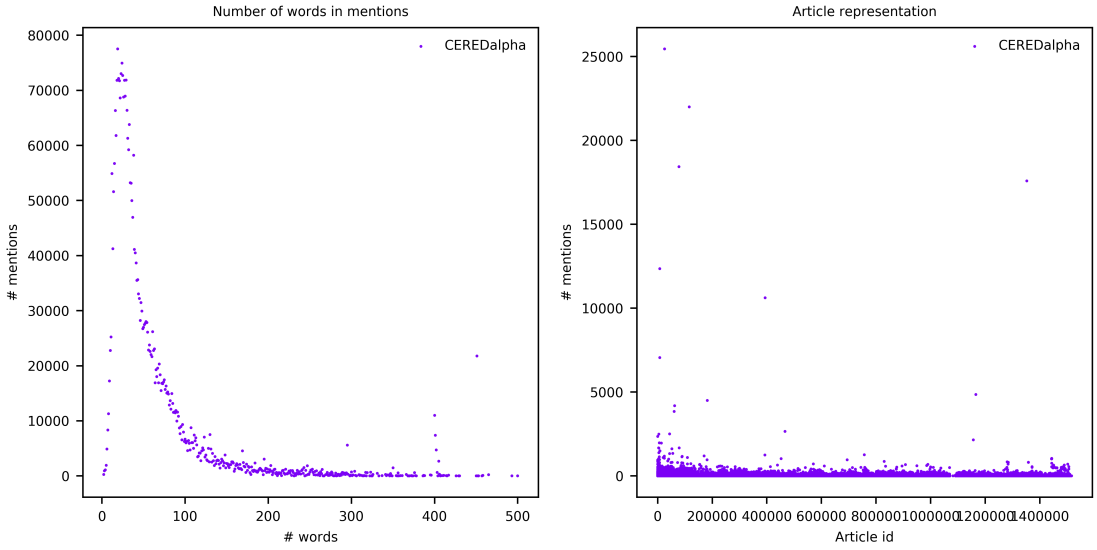
The most challenging aspect of working with Czech Wikipedia and Wikidata is their size and diversity. To the best of our knowledge there is no strictly followed guideline, when it comes to editing either the articles or item information. Just converting Wikipedia dump to clean text is challenging, due to user defined templates and other constructs we might be unaware of. On the other side, names in Wikidata can be too general (like someone's first name) and create false entity mentions.

Any change we make in the generator affects the generated dataset and analysing the consequences is difficult. We used several methods to measure the quality of the implemented generator.

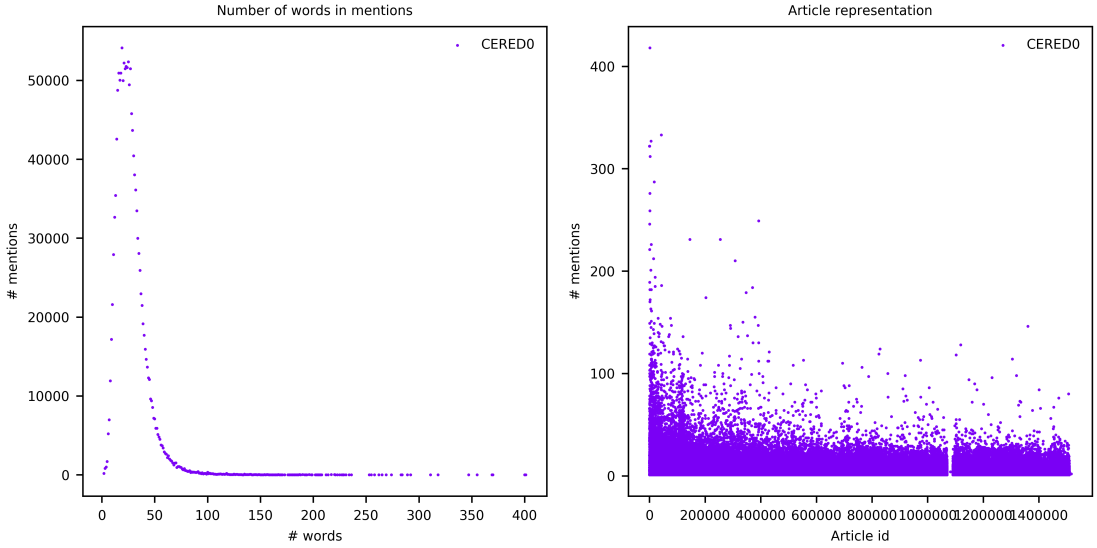
The simplest characteristic of the dataset is its size. Even though we want to maximize the size of dataset, we want to keep the precision high (eliminate false positives).

We can find false positives by finding anomalies in the dataset. We can look at different graphs characterizing the dataset, identify unexpected peeks and investigate the mentions causing them. An example of such graph is the distribution (histogram) of the number of words in a sentence and article representation is shown in Figure 3.7. Other useful characteristics include the length of a sentence in characters, relation distribution, article distribution, the number of sentence within its article, and so on.

Such graphs are better at capturing false positives than false negatives. To evaluate false negatives, it is necessary to compare the articles with the corresponding mentions in the dataset. To that account, we created a tool, which can visualize the whole article and both the entity and relation mentions.



(a) CEREDalpha – alpha version of the dataset before we removed Wikitext templates



(b) CERED0 – the final version of the generated dataset

Figure 3.7: Comparison of the histogram of the number of words (left column) and article id (right column) in two different versions of the dataset. In the left column, we can see that the curve is smoother and also less skewed to the right in CERED0. In the right column, we can see that in CEREDalpha several articles contributed over 5000 mentions, in contrast with the maximum contribution of approximately 400 mentions in CERED0. Considering that CEREDalpha is approximately two times bigger than CERED0, such smoothing of the distribution is significant.

Originally, we wanted to manually label entity mentions and relation mentions on some articles and use them to automatically evaluate the generator. However, we encountered many challenges when we tried to create such manual dataset, the main three are:

- the selected articles should represent Wikipedia (for example, articles about people are very different from articles about sport matches);
- to label entity relations, we need gold entity linking, which we must create ourself – that is challenging both because of the vast number of Wikidata items and because it is difficult to draw the line of what we expect to match (in the Figure 3.3 example, the "prostředí knih a filmů o Harrym Potterovi" clearly mentions the *Harry Potter universe Q5410773* and Voldemort is a ommited subject of the second sentence);
- it is time consuming to look for relations in Wikidata, due to the large number of relations.

3.4 Used Technologies

We chose Python to be our main programming language. To be able to work faster with a bigger volume of data, we wanted to use a CPU cluster, which leads to Spark. To top it, we use MorphoDiTa to work with the Czech language. We implemented a simple Streamlit application, which we used to comfortably view the results of our Spark queries.

In this section, we briefly introduce these technologies.

3.4.1 Python

Python is probably the most popular programming language in the ML community. It is a high-level language with a wide range of libraries. Libraries as NumPy, Pandas, and Spark enable fast and accesible computation. Tensorflow, scikit, and PyTorch allow users to focus mostly on data and ideas in machine learning. Less known libraries help us with Wikipedia parsing (wikitextparser, mwparserfromhell) or easy-to-create web applications (Streamlit).

3.4.2 Spark

Apache Spark framework provides a dataset manipulation API and allows executing it in a distributed way on a cluster, without actually implementing any parallelism. Therefore, Spark can boost the speed of computation as well as the available memory for the computation.

3.4.3 MorphoDiTa

Morphological Dictionary and Tagger (MorphoDiTa) [Straková et al., 2014] is an open-source tool for morphological analysis of natural language texts. It is designed to work well on inflective languages and achieves state-of-the-art results for the Czech language. MorphoDiTa provides a Python package trough which we can perform all standard operations such as tokenization and lemmatization.

3.4.4 Streamlit

Streamlit is a framework for creating simple web applications. It has minimal and practical API designed for users from the machine learning community. We use this library to create a pleasant way of viewing the generated dataset.

3.5 Implementation

In Section 3.3 we discussed different approaches we could choose to solve different aspects of the generation of the dataset. Now we build on the performed analysis and describe in detail the approaches we implemented.

3.5.1 Wikidata Preprocessing

The goal of Wikidata preprocessing is to load the Wikidata dump file and output three lists. The first list contains the QPQ triples, which are triples of identifiers representing a relationship (Q173998 P3373 Q187923, for example). The second list provides us with the names for the identifiers mentioned in the first list (Q187923 Ginny Weasley, P3373 sibling). The third list contains the mapping of entity identifier to the title of the corresponding Wikipedia article, if such exists. Those three lists serve as the source of structured data.

The first step of our preprocessing is to filter wikidata to remove entities that we will not use. We require each entity to have at least one Czech name (alias or label), otherwise, there would be no good way to find mentions of that entity later in the entity matching step. CERED creation does not require the Czech names of relations, so we will keep all of them.

Apart from the filtering, we should consider, whether we might benefit from keeping more information about a relationship (meaning the instantiation of a relation) than just the identifiers. Wikidata relationships often contain additional information specific for the relation type. For example, the *position held P39* relationship between *Cornelius Fudge Q1250951* and the *Ministry of Magic Q6017614* has following additional information attached to it: *start time 1990, end time 1996, replaces Millicent Bagnold, replaced by Rufus Scrimgeour*. In our use case, given that we plan to limit the text analysis to the syntactic level, such information is not beneficial.

The second step addresses removing duplicate relationships that differ only in the additional information tied to the relationship (CBS received many Peabody Awards, for example). Such duplicates, where the entire QPQ is the same, might require special attention in the relation matching step. We remove even relationships that differ only in the "P" part. If we kept them, in the relation matching we would either create multiple relation mentions (sentence with two tagged entities and the label of their relationship) that only differ in the relationship. However, CERED is designed to be a dataset on which it is possible to train a model for a single-label classification task. The dataset could be extended to allow multi-label relation type classification. However, it would complicate the models and evaluation, and the existing English datasets are also single-label, therefore, we decided to aim for a single-label dataset in this thesis.

After these steps, the first list contains approximately 2 million QPQ triples.

3.5.2 Wikitext Parsing

In this step we aim to parse Wikitext (Wikipedia markup language) from the Czech Wikipedia dump into a clean text with attached information about Wikilinks in the original markup.

As we already explained in Section 3.3.2, Wikitext contains a lot more than fully unstructured data. Different kinds of infoboxes, tables or lists are contained within the sentence-like text. Some of these elements are implemented using the so-called template syntax. Therefore, it would be tempting to simply remove all the text that is contained in a template. The problem is that some templates are useful. For example, we may use a template to divide the text into two columns containing valid sentences. Therefore, discarding all such data seems unnecessarily harsh.

When developing the methodology for Wikitext parsing, there was not a "gold" output to compare with. The only means of evaluation we had at the time was repeatedly going through a small set of articles and trying to discard unnecessary data. We tailored the rules for Wikitext parsing to these articles in such a way that only sentence-like parts remained.

Once we implemented the whole CERED generator and were able to see the relation mentions, we realized that the previous method of evaluation was not good enough. Therefore, we adopted a new one, as described in Subsection 3.3.6. We looked at the different histograms and investigated the abnormalities. For example, a lot of sports articles report results of a match (tournament, event) and these are often stored in custom tables that were not filtered by the rules from the previous paragraph. Moreover, these tables often contain information about the nationality of the players, resulting in a huge amount of matched entities and relations.

Based on the analysis of all the available data, we decided not to include the following content in the clean text:

- HTML tags within wikitext;
- headings;
- tables;
- lists;
- templates matching the following patterns: obsazení*, sloupce*, seznam*, příbuzenstvo*, *předkové*, *box*, *locmap*, *tabulka* (the English equivalents are cast*, columns*, lists*, relatives*, *ancestors*, *box*, *locmap*, *table*);
- wikilinks to categories and files.

One more technical issue we encountered was correctly assigning spans to Wikilinks, i.e. where the link starts and ends in the text. We can demonstrate the problem on the following sentence: uvozovky

> "The main [[story arc]] concerns Harry's struggle against [[Lord Voldemort]], a dark wizard who intends to become immortal, overthrow the wizard governing body known as the [[Ministry of Magic]] and subjugate all wizards and [[Muggle]]s (non-magical people)."

The correct span for Muggles should contain the trailing s even though it is not part of the Wikilink itself. In Czech such trailing characters are common. The set of characters that seem to end Wikilinks written in such forms are ␣,.\n.

One more thing we mentioned in the analysis about Wikitext is the possible boost of performance, if the text-part of a Wikilink was added to the set of names for the given entity. We exported such names, kept only those that were not already added to Wikidata, and read through many of them. This process is time-consuming, because one often has to actually look up the entity to know whether a given name is sensible. Even though we do not have any data about the proportion of good and bad names, the overall impression was clearly leaning towards not using such data. The two main reasons were that commonly the name was actually a class name, not instance name (like school linking to Hogwarts), and that pronouns were also linked to corresponding entities (such cases were less frequent, but would likely cause much trouble later on).

3.5.3 Entity Matching

We discussed in great detail the pros and cons of different entity matching methods, implying that the more complex the matching method, the better. We work with a single language and tools for lexical analysis are available and reliable. Therefore, implementing language-independent matching methods (string similarity, for example) is not beneficial.

We load the entity names in a slightly transformed form – we lower the case and add spaces around every dot character. We then use a lexical analyzer to split text to sentences and to obtain features from sentences (tokens, lemmas, and lexemes). An entity name (sequence of k tokens separated by a space) is matched in a sentence if (i) the entity is a candidate entity for the sentence, and (ii) there is a sequence of k consecutive tokens in the sentence, such that each token in the name is a member of the lexeme of the corresponding token in the sentence.

We intended to allow a less strict word order, but we were unable to justify such a choice. After reading several articles, we did not find any entity mentions that would be newly matched. This might imply that even though word order is relatively free in Czech, noun phrases tend to keep their word order. The other explanation is based more on the fact, that a human reader is more likely to recognize an entity mention, if it is in the standard word order. We, therefore, believe that matching entities in any word order would decrease precision more than increase recall, and do not allow it.

We considered allowing one special case. Most articles are based on one entity, therefore we expect many sentences to mention this entity. Often the entity is mentioned either by a pronoun (pronouns that express the subject are typically omitted in Czech), or by part of its full name. We already stated that we ignore pronouns but we tried to propose rules for choosing the correct substring of the entity name. The diversity of Wikidata makes such a task extremely difficult. Together with the risk that we could decrease the precision of entity matching, we decided to stick with full names only.

In the Wikitext parsing section, we prepared spans and identifiers for Wikilinks. We merge these with the ones matched by this module and post-process

them – we discard each mention, whose span is within a span of a different mention of the same entity. This removes duplicates and keeps the one that is the longest.

3.5.4 Relation Matching

So far we obtained sentences with tagged entity mentions. For each tuple of entity mentions within the same sentence, we checked if a relationship of those two entities was present in Wikidata (using the prepared QPQ list). Given the filtering in Wikidata preprocessing, we are guaranteed that there is at most one such relationship.

At this stage, we need to address incorrectly matched entities that cause the dataset to bloat. One example of such bloating that we encountered was in an article about kindergarten,[5] in the following sentence, where thousands of relation mentions were found. *"Jsou závazná pro předškolní vzdělávání v **mateřských školách**, v **mateřských školách** zřízených podle § 16 odst. 9 školského zákona, v lesních **mateřských školách** a v přípravných třídách základních škol."* Many kindergartens are named Mateřská škola (kindergarten), all of them are an instance of the abstract kindergarten entity and therefore candidate entities. If a sentence contains the term "mateřská škola" (or its form), all these entities will be matched, therefore, the relationship "Mateřská škola is a mateřská škola" will be assigned many times as well.

After investigating many other unusual cases, we decided to discard any sentence with at least 10 entity mentions in it. We also tried to experiment with different limits, but the results were unconvincing. For example, increasing the constant to 50 keeps an additional 13% of relationship mentions, but extends the set of sentences only by 1%.

3.5.5 Characteristics of the Generated Dataset

The full dataset, which was obtained by the process we described in the previous sections, contains almost one and a half million relation mentions. In the next few paragraphs, we present more detailed statistic of this dataset - CERED0.

We found at least one mention in 293 591 articles. In these articles, the average number of found mentions is slightly less than 5 and the median is nearly 7. The article with most mentions is Spojené království[6] (United Kingdom). The distribution of mentions in articles is shown vložit histogramy

There are 490 501 different sentences that contain a mentions. We set the upper bound on the number of entity mentions per sentence to 10. On average there were approximately 3 relation mentions in a sentence (that had at least one mention) and the maximum of 72 mentions per sentence was reached 34 times. The length of sentences ranges from 2 to 401 tokens, where the very short ones usually come from templates that were not removed. On the other hand, the very long ones are often caused by incorrectly written articles.[7] We tried to remove all

[5] `https://cs.wikipedia.org/w/index.php?oldid=18388144`

[6] `https://cs.wikipedia.org/w/index.php?oldid=18752891`

[7] Example of an incorrectly written article `https://cs.wikipedia.org/w/index.php?oldid=18723498`

templates to see if the range (and distribution) of the number of words improves, but we did not achieve a significant improvement.

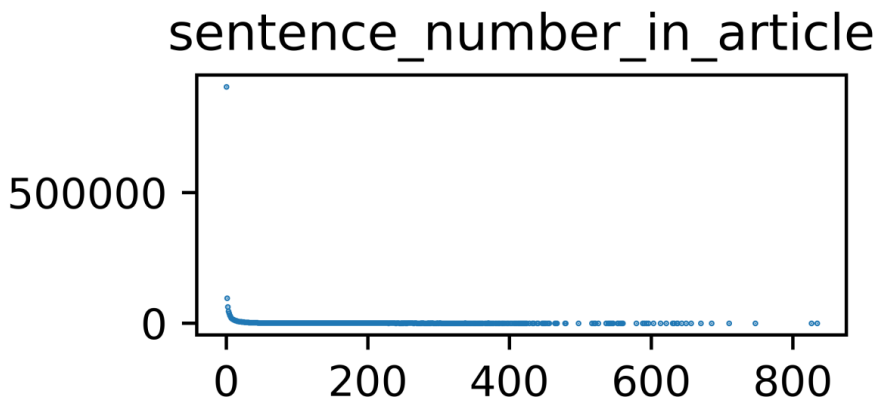


Figure 3.8: todo

Another possibility is to observe how the position of a sentence in an article influences the number of relation mentions. We expected that the first sentences in the articles would contain the highest number of relation mentions. The first sentences tend to contain Wikilinks and the use of pronouns or shorter names is limited, because each entity has to first be introduced by its full name. As we can see in Figure 3.8, our hypothesis seems to be correct, and partially applicable to several initial sentences, not just the first one. Surprisingly, 904 803 mentions come from sentences that are first in their respective articles – this constitutes over 60% of all mentions.

3.6 CERED Versions

In this section we describe how and why we decided to postprocess the generated datasets.

The full CERED is already a valid relationship classification dataset. It has nearly one and a half million mentions, but as we discussed in the previous section, some of them might be of poorer quality than others. In this section, we describe different versions of CERED with CERED0 being the biggest (least filtered) and CERED4 the smallest.

Each dataset version is split up into three disjoint sets: a train set, a dev set and a test set. Ideally, the test set would operate on a different set of entities (so that models learn to predict relationships based on sentences, not on the knowledge of entities). We believe that such a restriction is unnecessarily strong. Some entities are mentioned in many articles, which does, in fact, make them part of common knowledge (connected to the language we train on). Instead, we decided to relax the restriction to distinct articles. We sampled two sets of 10 000 articles, one for the test set and one for the dev set. In each version of CERED, the same articles are used for dev, test and train.

Table 3.2: CERED statistics.

Dataset	# mentions	Inventory size	Negative %	Test dataset
CERED0	1 462 546	692	0	25 066
CERED1	999 287	64	14.1	17 214
CERED2	374 914	64	18.8	6 530
CERED3	157 972	64	22.6	2 787
CERED4	10 766	59	22.1	188

3.6.1 CERED0

CERED0 is the raw dataset we described in the previous section. We do not artificially change the relation type, so no negative relation is present. We do not recommend this version for direct training, we mostly keep it to preserve the full information obtainable from our generator.

3.6.2 CERED1

CERED1 is close in size to CERED0. We removed long (over 100 words) and short (under 5 words) sentences. We also removed overly represented relations and changed labels of under-represented relations to OTHER, which is the negative relation in CERED1. We properly define under-represented and over-represented relations in the next paragraph, where we describe the CERED2 dataset.

3.6.3 CERED2

CERED1 has 3 potential flaws that we have not addressed yet. First, one sentence could be included multiple times in the dataset. Second, there are some relations with only a few relation mentions. Third, we did not try to handicap overly represented relatations.

We try to address these issues in CERED2. We start with CERED1 and for each sentence, if there are multiple relation mentions, we keep only the mention that was the least represented in CERED1, and discard the others. Next, we count the sizes of representations of all mentions and say that a relation is under-represented if it has less than 1 000 mentions. We take all the mentions whose relation is considered under-represented and relabel them to OTHER. This change is also applied to CERED1. After such relabeling, the OTHER relation still was not the most represented. Lastly, we decided to discard mentions of the two most represented relations: *INSTANCE OF P31* and *COUNTRY P17*.

3.6.4 CERED3

Even after the restriction on the uniqueness of sentences, half of the CERED2 sentences are the first sentences within the articles they originated from. Such sentences have a rather unique structure that is not as common outside of Wikipedia. This inspired CERED3, which is the remainder of CERED2 after removing all mentions in the "first" sentences.

3.6.5 CERED4

There are two stages in the CERED generation process that might be relatively unprecise. The entity matching stage and the relation matching stage. CERED4 keeps only the relation mentions from CERED3, in which both entities were manually labelled directly in wikitext in the form of wikilinks. The CERED4 dataset is, unfortunately, small, but has the highest potential to be precise.

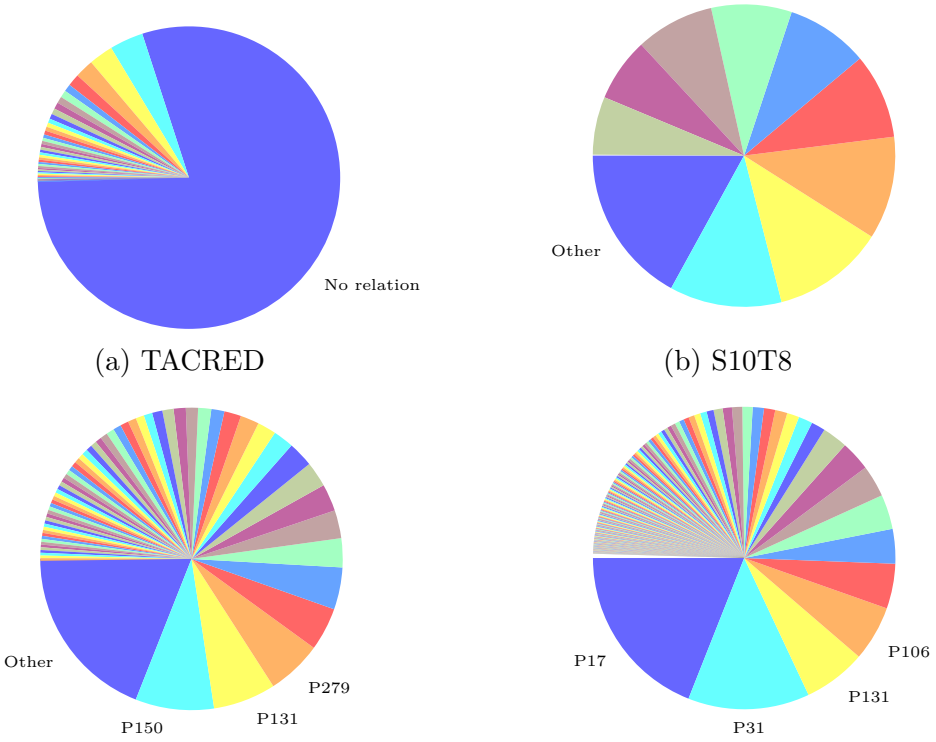
3.6.6 Other Considered Variations

When curating the CERED we considered many different criteria. For example, we chose the constant 1 000 for underrepresented relations after several experiments. The advantage of this value is that only two relations were more dominant than the new OTHER relation and at the same time, we would keep 63 positive relations. We wanted our dataset to be at least somewhat comparable to other datasets in the proportion of data and inventory size. S10T8 contains 10 717 mentions and 10 relations, TACRED contains around 100 000 mentions and 41 relations. The size of Riedel NYT depends, just like its inventory, on what data is used (and whether duplicates are removed and so on), but commonly used version contains nearly 700 000 mentions and 58 relations. Since the sizes of CERED1-3 range from approximately a million to approximately 160 000 mentions, we wanted the relation inventory size to be around 50 or 60. The distributions of relations in different datasets is captured in Figure 3.9 and detailed for CERED1-4 in Figure 3.10.

Named entities. One more labour intensive variation we wanted to create was supposed to be focused on named entities. We tried to curate a set of named entity types (such as person, place, etc.) based on Wikidata, in order to later use them as additional information during training.

We used the QPQ triples described in Subsection 3.5.1 to create an oriented graph with two types of edges. The first edge type – instance edges – connects Wikidata items Q_1 and Q_2 if there is a triple Q_1P31Q_2, where P31 is the Wikidata property *instance of P31*, and the second type – subclass edges – are added for the *subclass of P279* property. The goal of the graph is to capture the transitive nature of the two properties, i.e. if Harry Potter is an instance of a fictional human and fictional human is a subclass of a human, we can assume that Harry Potter is a human. We consider a Wikidata item to be a named entity if it is an instance of some other item. If we are interested in all the named entities belonging to some entity type (node in the graph), we can simply find all the named entities for which there is a path between the type and the entity.

We hoped that with such representation, we would be able to curate reasonable set of types. For example, we might want to have a "human" entity type and we might intuitively expect *Harry Potter Q3244512* (fictional human) to be a human. However, as he is fictional, he is not considered to be human in the wikidata ontology. Such an issue can be solved, for example, by adding another type of edges for *fictional analog of P1074*. The problem is that there are just too many similar problems and it is out of scope for this thesis to properly analyze the whole wikidata ontology.



(a) TACRED (b) S10T8

(c) CERED2 (d) CERED0

Figure 3.9: Representation of relations in different relationship extraction datasets. In CERED0, only top 250 relations are shown, the rest corresponds to the empty sector above P17. We only kept the most relevant labels.

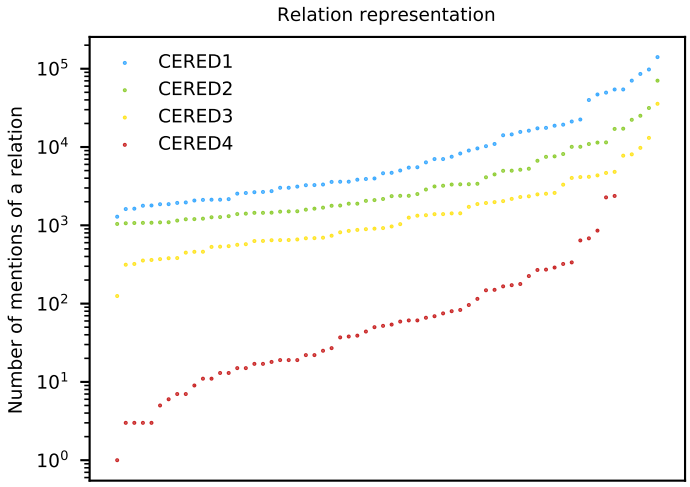


Figure 3.10: Sizes of relation representations. For each we ordered the relations by their frequency.

Part II

Training

4. Previous Work on Relationship Extraction

In this chapter, we first introduce the most popular pre-trained models that are currently used in natural language processing (NLP). Then we discuss different metrics that are used on Relationship Extraction.

4.1 Deep NLP Models

Lately, NLP tasks have been dominated by solutions using pre-trained deep neural models with the Transformer architecture. This section aims to roughly describe such a model to a reader familiar with the basics of neural networks. We introduce the two big concepts used in such models: the Transformer architecture and pre-training. Then we are able to explain how an actual modern NLP model looks and is trained. We choose the BERT model for this purpose since it is likely the best-known, and we also use it later in this thesis.

4.1.1 The Transformer Architecture

Suppose that we aim to perform some NLP task, like translation, using deep neural networks. We want the network to read a sentence in language A and output a translation of the sentence in language B. If the model was just a sequence of densely connected layers (so-called feedforward neural network), it would be nearly naive to expect the model to function. There is no structure that would help with gathering information about the entire sentence, and there is also no intuitive place where we would expect the model to switch from one language to another. There is nothing ensuring that the translated sentence is good. Those arguments are vague but might provide some intuition.

The recurrent neural networks (RNN) were used in the past to help the network capture the entire sentence. The RNNs introduce a cell that is more complicated than a typical neuron (perceptron). This cell usually has internal memory and the ability to both add and forget information. In the network, this cell is then presented with a sequence of inputs. The cell then goes trough the sequence one by one and edits its internal state. If we feed it with words of a sentence, we might expect the internal memory to contain an abstract representation of the sentence.

There are two main issues with RNNs. Working with RNNs is slow. We input the sentence "word by word", so instead of one step per layer, we need n steps for a sentence of length n. Moreover, especially if the sentence is long, it is difficult for the cells to remember the whole sentence and not just the later part. Both of these issues arise from the fact that input is fed into RNN cells sequentially.

In Vaswani et al. [2017], the Transformer architecture is introduced. It builds on a so-called encoder-decoder architecture. In the translation task, we can roughly say, that the encoder would process the sentence in language A, the decoder an already predicted part of the translated sentence, and the network would predict the next word of the translated sentence.

The issue of the accumulation of the right information at the right place is addressed with an attention mechanism. Attention was used first in the RNN architecture. The traditional attention mechanism makes it possible to focus only on the important parts of the input sentence. For example, if we want to translate a specific word in a sentence, the network can learn that some parts of the input sentence are more important than others. Later, it was discovered, that a variant of the attention itself, called *self-attention*, is powerful enough to make the sequentiality of the RNN cells unnecessary.

Self-attention was introduced in the Transformer architecture. Unlike the traditional attention, which is used to combine information from different layers (for example, it can transfer information from encoder to decoder), the self-attention usually works with inputs of the same layer as where it is applied. Self-attention attempts to add context information to the current embedding from the current embedding of the input sequence. For each input vector, we first compute three vectors: key, query and value. When computing self-attention for the i-th input, we start by computing scores which are the dot product of the i-th query vector with each key vector. The j-th score represents the similarity of the i-the query and the j-the key vector. We then convert the scores into a distribution using a softmax function and we use the resulting probabilities to compute the weighted average of the value vectors to create the i-th output. We described the operations on vectors, but they can be rewritten for whole matrices, which makes them faster to compute. Precisely, let Q, K and V be matrices, where the i-th row is the i-the query, key and value vector. Then the attention can be computed as

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where d is the dimension of the key vectors.

In the Transformer architecture, both the decoder and the encoder contain a sequence of blocks, where each block is a self-attention layer followed by a feed-forward network. We include a diagram of the architecture from the original paper (Figure 4.1)

So far, we ignored the fact that words cannot be directly fed to the network as an input. Usually, words are mapped to a vector in an embedding space. Such mapping is reusable in multiple tasks and brings us to the next section, where we will explain them a bit more.

4.1.2 Pre-training

The training of deep neural networks can require a great deal of both computational resources and annotated data. The term pre-trained model indicates a (part of) neural network, that can be reusable for multiple tasks or settings. Moreover, the pre-training is often based on unsupervised data, which lowers the requirements for annotated data later in the fine-tuning. Such reusability fastens deployment of complicated models and makes them accessible for a broader spectrum of people and companies. Pre-training is time-efficient, economical and consequently ecological.

Word embeddings can be seen as an example of pre-training. Word embedding is a vector that represents a word in a vector space (as, for models, vector spaces

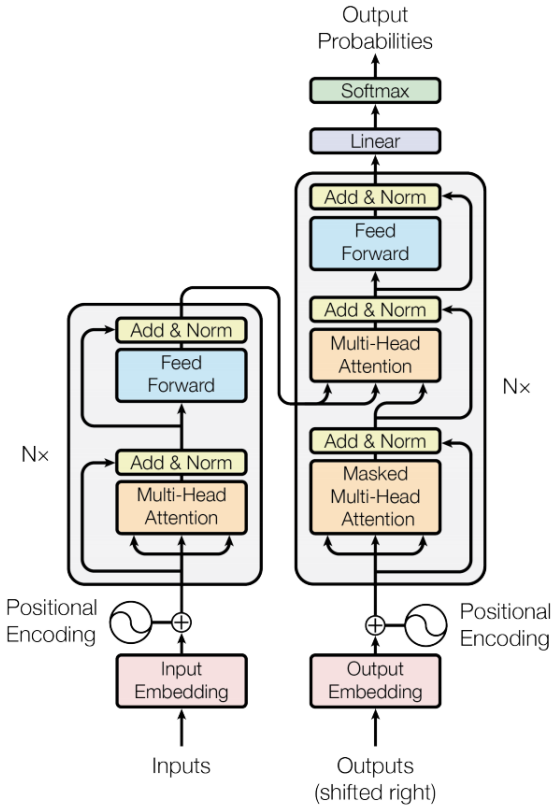


Figure 4.1: Diagram of the Transformer architecture as proposed in [Vaswani et al., 2017]. Taken from [Vaswani et al., 2017]

are easier to work with than words). The distance of embeddings of two words should reflect their semantic relation. An embedding itself is usually inferred from a pre-trained neural network and is used as an input for the task-specific model. The model that generates embeddings is usually trained by some variation of predicting a missing (masked) word from its context. The popularity of word embeddings rose in the previous years, most popular embeddings were likely the Word2vec [Mikolov et al., 2013].

Nowadays, a new type of embeddings is starting to get popular. In the classical word embeddings, a word always has the same embedding. The word "book" has the same embedding in *"She wrote a book on car maintenance."* and *"We were advised to book early if we wanted to get a room."* which is intuitively wrong. Contextualized embeddings take into account the entire (or at least preceding part of) sentence. One such embedding can be created via the BERT model.

4.1.3 BERT

BERT (Bidirectional Encoder Representations from Transformers, [Devlin et al., 2019]) is a family of models that achieves state-of-the-art results on various NLP tasks. BERT model takes a sequence of words as an input and computes a representation of each word while taking into account both the left and right-side contexts of the word. BERT models are pre-trained on large corpora of unlabeled text and then fine-tuned for a specific task.

The model was pretrained on two NLP tasks – Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

MLM is an adaptation of the task of prediction of a word from its context. The model takes a sentence, where some percentage of the words is replaced with a special token. The goal is to predict the original words in the sentence. The masking is essential – without it, the task would be trivial; at the same time, some context has to stay unmasked, otherwise, the task would be impossible. If we always mask the percentage of the words with the special token, a significant mismatch between real sentences and the one used in training would be created. The masking is, therefore, more sophisticated. We choose 15% of words to be masked. If a word is to be masked, with 80% change, it is replaced with the special token, with 10% it is replaced with a random word, and in the last 10% the word is kept (which helps to bias the model towards the actual word).

In NSP, we try to guess whether two sentences were consequent in the original text. The input is structured into two sentences via a separator token and segment embeddings. This task helps the model to learn to work on the sentence level, not just on a token level.

To fine-tune the model, we usually add a shallow feed-forward network on top of the BERT, which classifies chosen features from the BERT into the result space. Then we train the model on the task-specific dataset as we would train any other model.

In the original paper [Devlin et al., 2019] two BERT models were published: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, they differ in size, but both trained on the same English corpora (the BookCorpus and English Wikipedia). Later multilingual BERT was released[1]. It was trained on the top 102 languages with the largest Wikipedias including Czech.

4.2 Metrics

This section focuses on metrics since the quality of a model in the relationship extraction task is often hard to determine. We first define some standard metrics and discuss the pros and cons of each of them. Then we overview metrics used for the relationship extraction task on different datasets.

4.2.1 Binary Classification

We start with metrics for binary classification. In binary classification, we are presented with an input vector, and the goal is to determine whether the vector is of class A or class B. Each prediction then falls into one of the following categories: correctly classified A input, correctly classified B input, wrongly classified A input as B, and a B input wrongly classified as A (Figure 4.2).

Another way to look at the same situation is just to predict whether an input is of class A or not. This way the prediction is a True/False value determining whether the input is of class A. This way, we can define the previously mentioned categories without using the specific classes as **true positive** (TP), **true negative** (TN), **false negative** (FN) and **false positive** (FP). Visualization of the

[1] `https://github.com/google-research/bert`

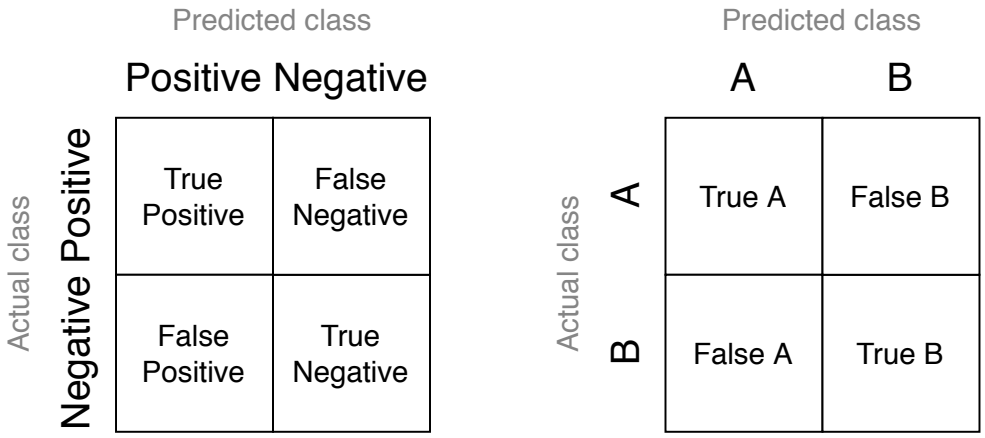


Figure 4.2: Confusion matrix for binary classification, normally count of occurrences would be written in each square.

result of classification on a dataset is called a **confusion matrix**, we include such matrix in Figure 4.2. We use the abbreviations to represent the number of predictions that belong to the given category.

Accuracy expresses the ratio between correct predictions and all predictions.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Precision expresses the ratio of correctly predicted positives within all predicted positives. Therefore, precision is a good metric if we want to avoid mistakenly classified negatives as positives.

$$Prec = \frac{TP}{TP + FP} \tag{4.2}$$

Recall is a complementary metric to precision. It expresses the ratio of all positives that were correctly predicted. In other words, it should be used when we need to find the maximum of positives in the data.

$$Rec = \frac{TP}{TP + FN} \tag{4.3}$$

Often we might want a trade-off between being as precise as possible and recalling as much as possible. **F1 score** is a harmonic mean of precision and recall (scaled to range from 0 to 1):

$$F1 = \frac{2 \cdot Prec \cdot Rec}{Prec + Rec} \tag{4.4}$$

F1 is quite widely used in competition tasks.

To emphasize that the right choice of metric is significant suppose that we have balanced data (both true and false classes are equally represented). If our classifier just predicted that every input is positive, we would obtain the following: 0.5 accuracy, 0.5 precision, and 1 recall. If we were to predict all negatives, accuracy and precision would remain 0.5 but recall suddenly drops to 0. If we were

to predict the result with even chances for both classes randomly, the expected results would be 0.5 for all of those metrics. We just described three very different classifiers, and the only thing we learned from accuracy and precision was that they were equally bad, without any insight about them. Recall, in contrast, gave us some insight about the predictions, but evaluated a bad classifier with the highest possible score. We should also note, that the metrics will likely change if we exchange which class corresponds to True and which to False.

This whole section is in this thesis mostly to remind us that if we want to score well in a given metric, we will likely exploit the metric even if it might worsen our classifier. The choice of a metric for a task determines what gets optimized. Later in this section, we debate such issues in our scenario, the relationship extraction task.

4.2.2 Multiclass Classification

We already run into issues with the asymmetry of precision and recall in binary classification (it is dependent on which class is chosen to be the positive one). We can address this by creating metrics per class. In the previous subsection, where we had a classifier for classes A and B, we would get two sets of metrics, each describing the ability of the classifier to recognize given class apart from the rest.

Now we can easily extend this per class approach to multiclass classification. The formulas will remain the same with the TP, FP, TN, and FN values corresponding to TP, FP, TN, and FN of individual binary classifications. If we compute those values out of confusion matrix (for class B), then TP is the value on position [B, B], FP is the sum of all in column B without TP, FN the sum of row B without TP and the sum of cells outside of the Bth row and column are the TN (Figure 4.3).

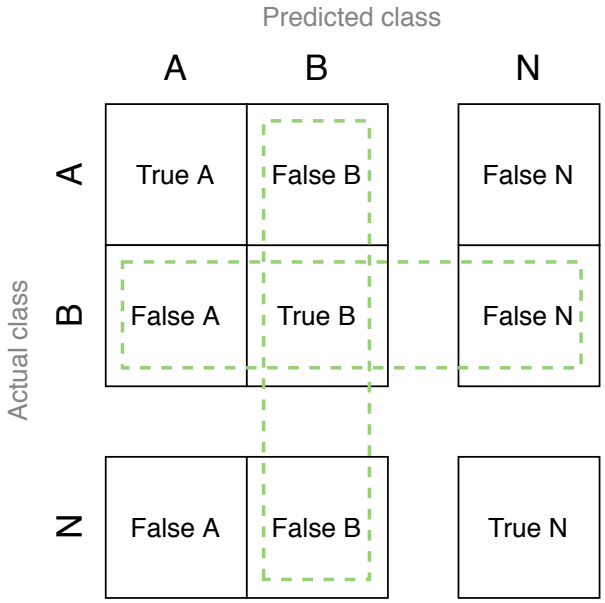


Figure 4.3: Confusion matrix for N-class classification

As a robust way of examining the quality of the classifier, one could simply look at the confusion matrix and all the per-class metrics. Although this would

be insightful, it is not the most practical in terms of a clear comparison of two classifiers. Ideally, we aim for a metric or metrics that are as descriptive and comprehensive as possible and also define an ordering of the classifiers.

Ideally, we would compute one number that fully describes the quality of the classifier. The common practice is to assign each metric a weight and combine them. Before we do so, we should acknowledge that the dataset we evaluate the classifier on needs to be taken into consideration.

An ideal dataset would be perfectly balanced. However, the class representation distribution (CRD) is often not uniform – classes are not equally represented.

Macro-averaged Metrics

An easy way to combine a metric over multiple classes into a single value is to compute the arithmetic mean. In most libraries and papers, the macro- prefix (macro-recall, macro-F1, etc.) is used. Macro-averaged metrics tend to be the easy option that is used without much thought. Moreover, despite the existence of two different definitions of macro-F1s, the exact formula is often not included in papers. (The more common formula is the arithmetic mean of classes F1s, but the less commonly used formula, where the F1 is computed from macro-recall and macro-precision, is also used, as [Opitz and Burst, 2019] reports.)

Micro-averaged Metrics

The second most used approach of computing a single metric over multiple classes is to compute a weighted average with the weights corresponding to the support of the individual classes. In most libraries and papers, the micro- prefix (micro-recall, micro-F1, etc.) is used.

The main difference between the micro metrics and the macro metrics is that macro metrics treat all classes equally, i.e. if there is a massively under-represented class, the metric influences the model to pay more attention to the class. On the other hand, we should use a micro metric if we want to optimize the overall accuracy of the model.

4.2.3 Relationship Extraction Metrics

In Chapter 2 we described three relationship extraction datasets. We now focus on the metrics they use and propose the recommended metric for CERED.

S10T8 metric

For the S10T8 datasets, Hendrickx et al. [2010] define the official metric directly in the paper: "Our official scoring metric is macro-averaged F1-Score for (9+1)-way classification, taking directionality into account." This means that the model classifies into all $2 \cdot 9 + 1$ classes, but before the macro is computed, the directions are merged (samples, which had wrong direction but correct relation are ignored, otherwise the TP, NP, TN and TP are summed) and the negative relation is ignored. Therefore the average is only from the 9 direction-less positive classes.

TACRED

On TACRED's git page,[2] we can find the official implementation of an evaluator computing a slightly modified version of micro metrics. The authors modified the metric such that the weight of the NO RELATION class is zero. The micro-F1 is computed from the micro-precision and micro-recall and is the primary metric for comparing results.

The decision to exclude the NO RELATION class is understandable. The class makes up 80% of the dataset, therefore the micro metrics would be mostly influenced by the performance on this class. Such behaviour is unwanted considering the fact, that in usual use cases we care more about the model to correctly classify positive relations rather than the negative one. We would like to note that it is not possible to cheat the metric by forbidding the classifier to predict the negative relation. If we do so, it would negatively influence the false positives of all classes and, therefore, decrease the overall performance.

Riedel NYT

For the Riedel NYT dataset, there was no official metric determined by the authors. Precision@N is commonly used since [Bollacker et al., 2008].

CERED

The Riedel NYT dataset shows that we should decide, what is the primary metric for CERED. We decided to use micro-F1. The negative relation in our datasets is not as dominant as in TACRED, so we do not think it is necessary to exclude it. On the other side, the datasets is not as nicely balanced as S10T8 and any macro-metric would be incline the models to overly focus on smaller classes.

[2] `https://github.com/yuhaozhang/tacred-relation`

5. Relationship Extraction on CERED

The goal of this part of the thesis is to train a relationship extraction model for the Czech language. To the best of our knowledge, there is no baseline for such a model, since there is no Czech relationship extraction datasets. In the first part of the thesis, we generated CERED, a family of relationship extraction models that we can use.

If we just trained the model on our dataset, we would be unable to determine, whether the model is well designed. We therefore adapt the model to be trainable both on Czech and on English. We assume that if the English version is competitive on popular English relationship extraction dataset, the Czech version is reasonably good.

In this chapter, we first describe the model, then we evaluate it on English datasets and lastly we report the results on CERED.

5.1 Model

We based our model on the BERT model described in the Subsection 4.1.3. The model architecture is quite straightforward. The goal of the whole model is to predict a relation based on a given relation mention. We modify the input by adding several special tokens as we show in Figure 5.1. Then we use the tokenizer, which is attached to the specific BERT, and we feed the output of the tokenizer to the model. The model computes abstract representations of all input tokens. We concatenate the representations that correspond to some of the special tokens ([E1], [E2] and [CLS]). On top we apply a dropout layer and finally a dense layer which predicts the class.

For Czech, we fine-tune the multilingual BERT, for English we fine-tune $\text{BERT}_{\text{BASE}}$. We use the transformers library (Wolf et al. [2019]) for BERT manipulation.

Rose Weasleyová je nejstarším dítětem **Rona Weasleyho** a Hermiony Weasleyové.
[CLS] [E2] Rose Weasleyová [/E2] je nejstarším dítětem [E1] Rona Weasleyho [/E1] a Hermiony Weasleyové.

Figure 5.1: Special tokens added for each mention. The sentence is a mention of FATHER(Ron Weasley, Rose Weasley). We add tokens to mark where each entity starts and ends. The [CLS] token is part of BERT input and was pre-trained to capture the whole input sequence via the NSP loss.

5.2 Results

We trained the model described in the previous section on the three English datasets introduced in Chapter 2 and on CERED. We hoped for our model to be competitive with other models, that build on BERT. At the same time, we did not use any additional data for the training and we only used BERT base,

because the multilingual BERT is not available in the large size. Therefore we do not expect to score as well as state-of-the-art results. We referred to results provided by nlpprogres[1] when researching the scores achieved on each dataset.

We did not fine-tune the hyperparameters (such as the learning rate or dropout rate).

5.2.1 S10T8

On S10T8 the model stopped training after 10 epochs (i.e. in less than 20 minutes on a singe GPU) and achieved 86.54% in the official metric. The state-of-the-art result was achieved by Baldini Soares et al. [2019] with 89.5%, we include more results in Table 5.1.

Table 5.1: Results on S10T8.

Paper/Source	F1
Our model	86.54
Baldini Soares et al. [2019]	**89.5**
Wu and He [2019]	89.25
Lee et al. [2019]	85.2

5.2.2 TACRED

Training on TACRED did not take much longer (4 epochs and under two hours). The state-of-the-art on this dataset was also achieved by Baldini Soares et al. [2019] with 71.5% F1 (micro-averaged over instances with positive relationships). We scored 65.65, we include more results in Table 5.2

Table 5.2: Results on TACRED.

Paper/Source	F1
Our model	65.65
Baldini Soares et al. [2019]	**71.5**
Zhang et al. [2018]	68.2
Zhang et al. [2017a]	65.1

5.2.3 Riedel NYT

Our model was unable to properly train on this dataset. It learns to always predict the negative relation (about 80% of the dataset is the negative relation). The distribution of relations in the dataset itself is most likely not the main issue (since TACRED has a similar distribution of relations). It is an interesting open problem, why a straightforward BERT model does not work on this dataset, Moreira et al. [2020] encounter similar (less severe) issues. However, our main goal was to provide a model for Czech, so we leave the investigation for future work.

[1] `http://nlpprogress.com/english/relationship_extraction.html`

5.2.4 CERED

We trained our model on all CERED versions. The performance of respective models was mostly as expected. We pinpoint few things we observed in the results.

We assume that the highest macro-F1 was achieved on CERED2 because we choose the relation inventory for CERED1-4 based on CERED2. CERED0 has a huge relation inventory and therefore it is not surprising that when weighting over classes, the metric is low.

The fact, that model trained on CERED4 preforms relatively well, shows, how remarkable BERTs (and the concept of pre-trained models) are for tasks with small datasets. The low macro-F1 on CERED4 can be explained quite easily. When we put constraints on which mentions will be kept in CERED4, we did not ensure, that the dataset will be balanced. In Figure 3.10 we see, how imbalanced the dataset really is.

Table 5.3: Results on CERED, we measured micro and macro F1 on the corresponding test sets for each version. Moreover, we measured the performance of models trained on CERED1-4 on CERED2 test set, to see the direct comparison.

Dataset	# micro-F1	macro-F1	micro-F1 on CERED2 test
CERED0	84.49	44.33	-
CERED1	83.94	76.58	81.72
CERED2	84.78	79.79	84.78
CERED3	77.29	73.96	82.33
CERED4	77.66	58.84	79.79

Conclusion

In this thesis, we proposed a methodology for generating relationship extraction datasets using Wikipedia and Wikidata. We analysed several pitfalls we faced, and generated CERED (Czech Relationship Extraction Dataset).

We then designed a neural network architecture for the relationship extraction task. In the network, we employ BERT contextual embeddings, which increase the quality of the network. We demonstrated, that the proposed architecture preforms reasonably well on two English relationship extraction datasets, and we reported the performance of our model on CERED.

5.3 Future work

5.3.1 Other Languages

This thesis focused on relationship extraction in the Czech Language. We believe, that our work could be easily extended into different languages. In the CERED generation process, there are three language-dependent steps:

- Wikidata preprocessing – we only worked with items that had at least one Czech name. The filtering can be easily changed to arbitrary language. For languages with small Wikipedia, it might be reasonable to consider relaxing the condition.
- Wikitext parsing – names of templates are in the language of the Wikipedia. We could remove the content of all templates, but it would of course negatively impact the size of the dataset.
- Entity matching – this step is heavily dependent on the language. We believe that for some languages it might be satisfactory to change the language in MorphoDiTa (for example MorphoDiTa supports the Slovak language). We might consider exchanging MorphoDiTa with a more universal (UDPipe [Straka and Straková, 2017]) or mainstream (spaCy [Honnibal and Montani, 2017]) tool that supports more languages.

If we implemented the changes proposed above, we would be able to create a dataset and a model for each language that the multilingual BERT supports (coincidentally, BERT was trained on the 102 biggest Wikipedias).

5.3.2 Wikidata ontology

In Subsection 3.6.6, we mentioned that additional information for training could be obtained from Wikidata. Such information is often available in other relationship extraction datasets. In the future, we would like to use Wikidata for extracting such information and add it to CERED.

Bibliography

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1279. URL https://www.aclweb.org/anthology/P19-1279.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *In SIGMOD Conference*, pages 1247–1250, 2008.

Linguistic Data Consortium and New York Times Company. *The New York Times Annotated Corpus*. LDC corpora. Linguistic Data Consortium, 2008. URL https://books.google.cz/books?id=D4F2AQAACAAJ.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S10-1006.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Joohong Lee, Sangwoo Seo, and Yong Suk Choi. Semantic relation classification via bidirectional LSTM networks with entity-aware attention using latent entity typing. *CoRR*, abs/1901.08163, 2019. URL http://arxiv.org/abs/1901.08163.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

Johny Moreira, Chaina Oliveira, David Macedo, Cleber Zanchettin, and Luciano Barbosa. Distantly-supervised neural relation extraction with side information using bert, 04 2020.

Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *ArXiv*, abs/1911.03347, 2019.

Sebastian Riedel, Limin Yao, and Andrew Mccallum. Modeling relations and their mentions without labeled text. pages 148–163, 09 2010. doi: 10.1007/978-3-642-15939-8_10.

Milan Straka and Jana Straková. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K/K17/K17-3009.pdf.

Jana Straková, Milan Straka, and Jan Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Wikipedia contributors. Wikipedia — Wikipedia, the free encyclopedia, 2020. URL https://en.wikipedia.org/w/index.php?title=Wikipedia&oldid=947302871. [Online; accessed 28-March-2020].

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Shanchan Wu and Yifan He. Enriching pre-trained language model with entity information for relation classification. *CoRR*, abs/1905.08284, 2019. URL http://arxiv.org/abs/1905.08284.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark, September 2017a. Association for Computational Linguistics. doi: 10.18653/v1/D17-1004. URL https://www.aclweb.org/anthology/D17-1004.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45, 2017b. URL https://nlp.stanford.edu/pubs/zhang2017tacred.pdf.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings*

of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2205–2215, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1244. URL https://www.aclweb.org/anthology/D18-1244.

A. Attachments

A.1 First Attachment