Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Detekce pojmenovaných entit pomocí knihovny spaCy

Bakalářská práce

Autor: Jakub Včelák
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Martina Husáková, Ph.D.

Hradec Králové březen 2023

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 20.3.2023 Jakub Včelák

Poděkování:

Děkuji vedoucí bakalářské práce paní Ing. Martině Husákové, Ph.D. za metodické vedení, praktické rady a pomoc poskytovanou v průběhu zpracovávání této práce.

Anotace

Název: Detekce pojmenovaných entit pomocí knihovny spaCy

Bakalářská práce se zabývá oblastí detekce pojmenovaných entit (NER – Named Entity Recognition). Blíže se zaměřuje na možnosti nabízené knihovnou spaCy pro tuto oblast. V teoretické části je představena problematika zpracování přirozeného jazyka, porozumění přirozenému jazyku, detekce pojmenovaných entit a historického vývoje této oblasti. Dále obsahuje stručný přehled třech základních přístupů pro trénování nových modelů NER. První oddíl praktické části analyzuje webové stránky, natrénované modely a kód knihovny spaCy. Druhý oddíl se věnuje vývoji webového extraktoru pro demonstraci modelů poskytovaných knihovnou spaCy. Poslední oddíl praktické části je zaměřen na proces tvorby nových modelů za pomoci dvou přístupů představených v teoretické části. Tyto modely jsou následně testovány a výsledky prezentovány v závěru práce.

Annotation

Title: Named Entities Recognition using spaCy library

The Bachelor thesis deals with the field of detection of named entities (NER – Named Entity Recognition). It focuses more closely on the possibilities offered by the spaCy library for this field. In the theoretical part, the issues of natural language processing, natural language understanding, the detection of named entities and the historical development of this area are presented. It also contains a brief overview of 3 basic approaches for training new NER models. The first section of the practical part analyzes web pages, trained models and code of spaCy library. The second section is devoted to the development of a web extractor for demonstrating the models provided by the spaCy library. The last section of the practical part is focused on the process of creating new models using the two approaches presented in the theoretical part. These models are subsequently tested and the results presented at the end of the thesis.

Obsah

1 Úvod

Natural Language Processing (NLP) neboli zpracování přirozeného jazyka je rozsáhlá oblast zabývající se porozuměním přirozenému jazyku. V dnešní době se s různými formami NLP běžný uživatel moderních technologií setkává každý den. Přirozený jazyk se neustále vyvíjí a je ovlivňován vnějšími faktory. Tyto vlastnosti z něj činí složitější objekt ke zpracování na rozdíl od jazyků umělých, mezi které se řadí například jazyky programovací. NLP obsahuje mnoho podoblastí, mezi ně patří i Named Entity Recognition (NER) – detekce pojmenovaných entit. Oblast NER bývá často spojována s oblastí klasifikace, ve které se určuje konkrétní typ dané entity. Toto spojení se nazývá Named Entity Recognition and Classification (NERC).

NER se zabývá identifikováním entit jako například osob, organizací a geografických míst. V současnosti se pro detekci pojmenovaných entit používají různé varianty konvolučních neuronových sítí, rekurentních neuronových sítí a Transformer neuronových sítí. Pro přesné určení dané entity je potřeba trénovat neuronovou síť na značném množství dat. Požadovaný formát dat se liší podle použité architektury neuronové sítě.

Pro zpracování přirozeného jazyka existuje řada knihoven usnadňující tuto práci. Mezi nejpoužívanější z nich se řadí knihovna spaCy, která poskytuje řadu před-trénovaných modelů, propracovanou a přehlednou dokumentaci a také možnost natrénování vlastních modelů pro NLP. SpaCy se v oblasti trénování nových modelů spojila s platformou Hugging Face a nabízí i trénování modelů založených na architektuře Transformer.

Bakalářská práce je rozdělena na tři části. V první části je provedena literární rešerše zpracovávané oblasti spolu a formulováním cílů této práce. Následuje stanovení výzkumných otázek a k nim příslušných hypotéz. Dále je zde také určena metodika, která bude později použita k ověřování hypotéz.

Druhá část práce je věnována teoretickým předpokladům potřebných k vypracování praktické části. Jsou zde představeny základní informace v oblastech zpracování přirozeného jazyka s bližším zaměřením na sekci vyhledávání pojmenovaných entit. Nachází se zde i stručné shrnutí historického vývoje této

oblasti. V závěru teoretických předpokladů se autor zabývá současnými metodami trénování detekce pojmenovaných entit.

Poslední část této bakalářské práce je praktická. Zde je autorem analyzována knihovna spaCy se zaměřením na webové stránky knihovny, před-trénované modely knihovny a samotný kód knihovny potřebný k jejímu využití. Následuje část pojednávající o tvorbě webové aplikace s využitím spaCy a následné zhodnocení práce s knihovnou. Dále je zde popsán průběh trénování detekce nové entity v této knihovně. Práci autor uzavírá diskuzí nad výsledky praktické části a doporučuje oblasti pro další zkoumání.

2 Literární rešerše

Odborná literatura na téma Natural Language Processing neboli zpracování přirozeného jazyka je v českém jazyce zastoupena velmi málo, proto i při této literární rešerši bude vycházeno především ze zahraniční literatury.

Mezi odborné práce pojednávající o NLP v obecnějším měřítku patří například Natural Language Processing for the Semantic Web od Maynard et al. (2016). Nastiňuje zde kromě základních informací o NLP i využití této oblasti v tématu sémantických webových stránek. Xiao et al. (2021) ve své práci shrnuje problematiku trénování nových NLP modelů. Představuje oblast před-trénovaných modelů a nastiňuje potřebu vyřešení problému náchylnosti neuronových sítí na takzvaných „Adverserial example attacks" – útoků vedoucích k chybovosti daných modelů. Pais et al. (2022) uvádí přehled dvou oblastí, u kterých se očekává bližší propojení s NLP. První z oblastí je Cloud computing. Pais zde očekává nasazení NLP na cloud. Dále představuje propojení těchto dvou oblastí s Big data k extrakci informací na cloudu.

V současné době je oblast Named Entity Recognition zásadně ovlivněna architekturou Transformer, jak uvádí Roshanzmir et al. (2021). Zároveň ukazuje širší využití této architektury i NLP při analýze popisu obrázků testovanou osobou sloužící k detekci Alzheimerovy choroby. Architektura neutronových sítí založená na Transformer přinesla díky paralelizaci značný posun v oblasti trénování nových modelů. V této oblasti se nachází mnoho překážek, o nichž ve své práci píše například Goyal et al. (2017). Jak uvádí, patří mezi ně například nedostatečné množství zdrojů u některých jazyků pro kvalitní natrénování NER systému. Oblastí trénování detekce nových entit se mimo jiné zabývá ve své práci i Liu et al. (2021). Ve své práci se zabývá NER systémem pracujícím v čínském jazyce za použití BERT (Bidirectional Encoder Representations from Transformers) před-trénovaného modelu a také jednoho ze současných trendů na poli trénování detekce pojmenovaných entit.

3 Cíl práce, volba metodologie, způsob řešení

3.1 Cíl práce

Hlavním cílem této bakalářské práce je provést analýzu prostředků, které nabízí knihovna spaCy, pro práci v oblasti detekce pojmenovaných entit. Dále analyzovat webové stránky této knihovny a vytvořit webovou aplikaci extraktoru, která bude sloužit k představení možností knihovny v oblasti NER. Dalším dílčím cílem, je trénování detekce nové entity, kterou nebylo dosud možné vyhledávat, v knihovně spaCy. Tato funkce bude implementována v již zmíněné webové aplikaci.

3.2 Výzkumné otázky

Otázka č. 1

Je rozsah prostředků knihovny spaCy dostačující pro tvorbu webové aplikace pro detekci pojmenovaných entit?

Otázka č. 2

Do jaké míry umožňuje knihovna spaCy odstínění od složitosti NLP při práci v oblasti NER?

Otázka č. 3

Jak náročný je proces natrénování detekce nové entity v knihovně spaCy?

3.3 Pracovní hypotézy

Hypotéza č. 1

Knihovna spaCy poskytuje dostatečný rozsah prostředků pro tvorbu webové aplikace pro detekci pojmenovaných entit.

Zdůvodnění: Dle Cheng et al. (2020:516) patří knihovna spaCy mezi nejpoužívanější knihovny na poli akademickém i komerčním. Z tohoto tvrzení lze usoudit, že spaCy bude poskytovat více než dostatečnou podporu pro tvorbu webového extraktoru pojmenovaných entit.

Hypotéza č. 2

Knihovna spaCy umožňuje značnou míru odstínění od složitosti NLP, nikoli však úplné odstínění.

Zdůvodnění: Knihovna spaCy (2022) na svých stránkách uvádí: *„Knihovna respektuje váš čas a snaží se s ním neplýtvat. Snadno se instaluje a její API je jednoduché a produktivní.“* Reklamu lze v tomto tvrzení cítit výrazně, nicméně si autor práce dovolí v této fázi předpokládat, že knihovna nebude uživatele zatěžovat plnou šíří složitosti NLP.

Hypotéza č. 3

Proces natrénování detekce nové entity v knihovně spaCy je velmi náročný, nikoli však nemožný pro uživatele se znalostmi oblasti informačních technologií na úrovni bakalářského studia.

Zdůvodnění: Samotný proces natrénování detekce nové entity je značně náročný. Zvlášť pokud je požadována vyšší míra přesnosti detekce. Avšak knihovna spaCy je známá vysokou mírou abstrakce a snahou uživatele nezatěžovat složitostí prováděných procesů.

3.4 Metodika zpracování

Základem pro zpracování této bakalářské práce je studium odborných článků a knih, konkrétně z oblastí zpracování přirozeného jazyka a detekce pojmenovaných entit. Zdroje byly studovány převážně v elektronické podobě.

Práce je založena převážně na zahraničních zdrojích. Knihy a odborné články na toto téma nejsou v českém jazyce zastoupeny v dostatečném množství a především kvalitě, aby byly dostačujícím podkladem pro zpracování této práce. Z důvodu čerpání ze zahraničních zdrojů je v práci citováno primárně z cizích jazyků. Z tohoto důvodu byly doslovné citace přeloženy autorem.

V teoretické části je využita především analýza a studium zahraničních zdrojů, pro následné shrnutí a představení teoretických základů týkajících se oblasti zpracování přirozeného jazyka se zaměřením na oblast detekce pojmenovaných entit.

V praktické části se vychází především z informací z dokumentace knihovny spaCy a několika dalších odborných článků pojednávajících o práci s touto knihovnou. Dále je zde využito praktické zkoušky v podobě tvorby webového extraktoru pojmenovaných entit s využitím prostředků nabízených knihovnou spaCy. Následně je provedeno zhodnocení používání této knihovny z hlediska uživatelské přívětivosti a poskytovaných prostředků. V této části jsou také ověřovány stanovené hypotézy.

Hypotéza č. 1 byla ověřována studiem dokumentace knihovny spaCy na webových stránkách knihovny. Dále praktickým testování při tvorbě webové aplikace poskytující službu extrakce pojmenovaných entit. Závěry obou částí ověřování byly porovnávány s odbornou literaturou zabývající se knihovnou spaCy.

Ověřování hypotézy č. 2 bylo provedeno analýzou dokumentace spaCy a následnou praktickou zkouškou práce s danou knihovnou. Závěry byly porovnány s informacemi z teoretické části práce týkající se složitosti práce v oblasti NER.

Pro ověřování hypotézy č. 3 bylo využito analýzy možností knihovny spaCy v oblasti trénování detekce nové entit. Následně byla provedena řada cvičných trénování pomocí této knihovny. Závěrem byla zhodnocena náročnost těchto procesů.

4 Teoretická východiska

V této části budou představena teoretická východiska potřebná k vypracování praktické části bakalářské práce.

4.1 Přirozený a umělý jazyk

Přirozený jazyk lze popsat mnoha způsoby, avšak přesné definování je složité. Kumar (2011:1) popisuje skupinu přirozených jazyků jako *„jazyky, které se přirozeně vyvinuly a používají je lidé ke komunikačním účelům.“* Jak je uvedeno v definici, jejich využití nalezneme především v mezilidské komunikaci a to jak slovní, tak psané. Jako příklady přirozených jazyků lze uvést češtinu, angličtinu a případně další národní jazyky. Přirozené jazyky se na rozdíl od jazyků umělých dynamicky vyvíjí. Jejich vývoj může být ovlivňován vnějšími faktory jako například dalšími přirozenými jazyky. Pro jejich definování lze využít několik struktur. Mezi nejčastější patří slovník, tj. ucelený zápis všech slov daného jazyka. Dále lze jazyk popsat po gramatické stránce, tj. souhrnem pravidel, kterými je nutné se řídit při používání daného jazyka. Oblast, ve které se přirozený jazyk používá, se vymezuje především geografickou polohou a příslušenstvím k národu, na rozdíl od jazyků umělých, kde jde především o profesní příslušnost.

Colman (2008) ve svém slovníku definuje umělý jazyk takto: *„Jazyk záměrně vynalezený nebo zkonstruovaný, zejména jako prostředek komunikace v oblasti výpočetní techniky nebo informačních technologií.“* Za jeho vznikem často stojí technologický či vědecký důvod. Formování umělého jazyka je ovlivňováno výrazně menším počtem lidí než u jazyka přirozeného. Nejčastěji se jedná o odborníky v dané oblasti, kteří se pak dále podílejí na vývoji jazyka. Jeho využití je často omezeno na užší skupinu lidí s podobným profesním zaměřením. Využití nachází především ve vědecké činnosti a komunikaci člověka s technologiemi. Zřídkakdy jsou používány pro mezilidskou komunikaci. Mezi významnou skupinu umělých jazyků patří jazyky programovací. Využívání těchto jazyků podléhá přísným pravidlům a jejich porušení vede ve většině případů k nefunkčnosti daného sdělení.

Rozdílnost jazyků přirozených a jazyků umělých je v mnoha oblastech překážkou. Především v oblasti informačních technologií jsou rozdíly tohoto druhu

problematické. Přístroje jsou navrženy pro práci s jazyky umělými, a naopak lidé komunikují jazyky přirozenými. Proto je zde snaha, umožnit uživateli využívat přirozený jazyk ke komunikaci s přístroji. V některých oblastech, jako například hlasoví asistenti a chatboti, se již do jisté míry podařilo tuto problematiku vyřešit. Avšak ve většině případů se stále musí při komunikaci člověk přizpůsobit přístroji. V komunikaci je velmi důležitá stránka porozumění sdělení. Touto oblastí se zabývá Natural Language Processing (NLP) neboli Zpracování přirozeného jazyka.

4.2 Zpracování přirozeného jazyka

Chowdhury (2003:1) představuje NLP jako: *„oblast výzkumu a aplikace, která zkoumá, jak lze použít počítače k porozumění textu nebo řeči v přirozeném jazyce a manipulaci s ním, aby výstup vedl k užitečným věcem.“* Jak z definice vyplývá, klíčová je schopnost počítače strojově zpracovat vstup v přirozeném jazyce, porozumět mu a poté být schopen s textem pracovat. V této oblasti se využívají dva druhy přístupu ke zpracování vstupu. První z nich je přístup založený na znalostech. Tento přístup je starší a jeho základem je ruční definování pravidel. Ve většině případů byl tento starší přístup překonán novějším přístupem – strojovým učením. Dochází zde k výraznému zvýšení výkonnosti a přesnosti NLP. Strojové učení snižuje požadavky na vývojáře v oblasti lingvistiky. Nevýhodu tohoto přístupu je potřeba kvalitních dat k trénování, která jsou náročná na vytvoření. Zpracování přirozeného jazyka lze rozdělit na dvě podoblasti.

První z nich je Natural Language Understanding (NLU) neboli porozumění přirozenému jazyku. Zahrnuje přijmutí vstupu v přirozeném jazyce a následné zpracovávání vedoucí k samotnému porozumění danému vstupu. Touto oblastí se více zabývá následující kapitola.

Druhá oblast NLP se nazývá Natural Language Generation (NLG) neboli generování přirozeného jazyka. Touto oblastí se bakalářská práce nezabývá. Jedná se o schopnost počítače vytvářet text v přirozeném jazyce splňující všechny pravidla daného jazyka. Probíhá zde převod strukturovaných dat na nestrukturovaná. Ve výchozím stavu potřebuje generátor dostatečné množství dat v přirozeném jazyce, aby byl schopen podávat výsledky na dostačující úrovni. Ty jsou zpravidla složitá na vytvoření, jak mimo jiné uvádí Mellish (2006:83): *„Jak jsme diskutovali v preambuli,*

značné náklady na vyvinutí NLG systému pro novou doménu, je vytvoření slovníku souvisejících pojmů v doméně do slov v přirozeném jazyce, které lze použít k jejich označení.“

Obrázek 1: Schéma lingvistického procesu předzpracování.
Zdroj: Maynard et al. (2017:10)

Oblast zpracování přirozeného jazyka zahrnuje velké množství úkolů, proto se zpravidla rozděluje na několik dílčích úkolů. Toto rozdělní používá i Nadkarni et al. (2011:545) ve svém díle. Dílčí úkoly lze rozdělit na úkoly nižší úrovně a úkoly vyšší úrovně. Rozdělují se podle složitosti procesů prováděných v daném dílčím úkolu. Na schématu na obrázku 1 jsou vidět úkoly NLP nižší úrovně. Používají se pro předzpracování vstupu v přirozeném jazyce.

Při tokenizaci je vstupní text rozdělován na jednotlivé tokeny (jednotlivá slova, čísla nebo termíny). Jedná se o základní krok využívaný ve všech úkolech vyšší úrovně. Tokenizace není složitý proces, přesto se zde nachází několik klíčových komplikací, které by měl správný tokenizer zvládat vyřešit. Mezi ně patří rozhodování o složitějších slovních útvarech jako například slova spojená pomlčkou, lomítkem nebo apostrofem. V těchto případech musí tokenizer správně vyhodnotit, zda se jedná o jeden či více tokenů.

Segmentace vět je další z úkolů nižší úrovně. Dochází zde k rozdělení vstupního textu na samostatné věty. I tento úkol je klíčový pro většinu úkolů vyšší úrovně. V procesu segmentace vět bývají problematické pořadové číslice, zkratky nebo datum. Zde je nutné správně vyhodnotit, zda se jedná o konec věty či nikoli.

Maynard et al. (2017:15) definuje funkci POS taggingu takto: *„zabývá se označováním slov jejich slovními druhy např. podstatné jméno, sloveso, přídavné jméno.“* Jak je uvedeno v definici, jednotlivým tokenům vytvořeným při tokenizaci je přiřazen slovní druh. Zde dochází ke komplikacím v jazycích, ve kterých mohou být tytéž slova použita v různých případech jako jiné slovní druhy.

Morfologická analýza analyzuje jednotlivé tokeny po lingvistické stránce. Rozděluje slova na kořen, předponu, příponu. V této části procesu zpracování přirozeného jazyka je možné využít také stemming (stemování) – proces, který převede tokeny do jejich základního tvaru. Při stemmingu může nastat situace, kdy základní tvar tokenu nemá žádný význam. Tento problém je vyřešen v procesu lematizace, kde se hledá kořen slova nazývaný "lemma", který má vždy význam.

Syntaktický rozbor slouží k analýze vztahů jednotlivých tokenů ve větě. Výstupem tohoto procesu je syntaktická struktura věty, která je využita v dalších procesech k lepšímu porozumění danému textu. Existují různé přístupy k této oblasti. Jedním z nich je Constituency parsing, který vytváří strukturu věty jejím rozdělením na fráze podle gramatických souvislostí například na slovesnou část nebo jmennou část. V případě Dependency parsing je struktura věty vytvářena vztahy mezi jednotlivými tokeny v dané větě.

Chunking definuje Nadkarni et al. (2011:545) jako: *„Identifikaci frází z tokenů získaných z procesu POS tagging. Například fráze podstatného jména může obsahovat posloupnost přídavných jmen následovanou podstatným jménem."*

O úkolech NLP vyšší úrovně Nadkarni, et al. (2011:545) uvádějí, že stavějí na úkolech nižší úrovně a jsou obvykle určené pro specifický problém. Mezi tyto úkoly, patří oblast Information Extraction (IE) neboli extrakce informací. Jedná se o proces, jehož hlavním úkolem je získání strukturovaných informací z nestrukturovaných textových zdrojů. Samotný proces extrakce informací zahrnuje několik dílčích úloh jako například předzpracování textu, NER a identifikace vztahů mezi informacemi. Cunningham (2005:666) uvádí: *„Pokud do grafu znázorníme přijatelnou úroveň výkonu komponent analýzy, pak je zde zřejmý kompromis mezi složitostí procesu a specifičností informace, která má být extrahována."* IE je výkonnostně náročný proces, jak uvádí citace výše, proto je nutné najít přijatelný kompromis v poměru specifičnosti hledaných informací a složitosti jejich vyhledávání.

Do oblasti úkolů vyšší úrovně se řadí také Named Entity Recognition (NER) neboli Detekce pojmenovaných entit. Jedná se o proces, který v textu v přirozeném jazyce hledá předem stanovené tzv. pojmenované entity. Tato práce se bude více zabývat NER v následujících kapitolách.

Analýza sentimentu (někdy nazýváno jako opinion mining – těžení názorů) je velmi rozšířený úkol NLP. Dochází zde ke komplexní analýze daného textu a zhodnocení jeho obsahu. Vyhodnocuje se celkové vyznění textu a to ve většině případů jako pozitivní, negativní nebo neutrální. Zpravidla se využívá pro zpracování velkého množství dat a následným statistickým výstupům. Jak uvádí Chimaobiya et al. (2016:2) samotná analýza sentimentu nemusí být prováděna pouze na kratších textech, lze ji využít i pro analýzu obsáhlých dokumentů.

Podle definice, kterou používá Nadkarni, et al. (2011:545) k rozřazení úkolů NLP na nižší a vyšší úrovně, lze strojový překlad (machine translation - MT) zařadit mezi úkoly vyšší úrovně. Jedná se o proces, kdy stroj překládá text z jednoho přirozeného jazyka do druhého. V této oblasti se využívá několik přístupů. O v současnosti používaných metodách uvádí Popel et al. (2020:2) následující: *„V souladu s těmito pokroky se oblast MT posunula k používání metod založených na neuronovém deep-learningu, které nahradily předchozí přístupy, jako jsou systémy založené na pravidlech nebo statistické metody založené na frázích.“* Při strojovém překladu záleží, jaké oblasti se týká překládaný text. Kvalita překladu je výrazně vyšší, když byl daný model trénovaný i pro překládanou oblast.

Entity linking (EL) je oblast zabývající se přiřazováním jedinečné identity k entitám ve zpracovávaném textu a následném propojení dané entity se záznamem ve znalostní databázi, který poskytne podrobnější informace o dané entitě. Maynard et al. (2017:53) vyjmenovává tři problematické oblasti, které musí EL proces zvládnout. První z nich nazývá variace jmen. Zde je nutné vyřešit situace, kdy lze pro označení jedné entity použít více výrazů. Jako druhou oblast uvádí vysokou nejednoznačnost významu dané entity. Z důvodu rozsáhlosti znalostních databází, ve kterých se nachází miliony záznamů, je možné mít pro jednu entitu mnoho potenciálních možností k přiřazení. Poslední oblast je nazvána jako chybějící entita. Jak už název napovídá, jedná se o situaci, kdy EL nedokáže ve znalostních databázích najít žádný záznam k dané entitě. Named entity linking (NEL) je propojení oblasti Entity linking s oblastí detekce pojmenovaných entit. Na vstupu se zde nevyhledávají všechny entity, ale pouze entity pojmenované. Následně jsou vyhledané entity přiřazovány k záznamům ve znalostních databázích.

4.3 Porozumění přirozenému jazyku

Porozumění přirozenému jazyku (Natural Language Understanding – NLU) je jeden ze dvou základních podoborů Natural Language Processing. Cílem, jak už název napovídá, je porozumět obsahu dané zprávy. Jedná se o jeden ze zásadních kroků k umožnění komunikace v přirozeném jazyce mezi počítačem a člověkem. Vstupů pro samotné zpracování může být mnoho druhů. Jako příklady těchto vstupů uvádí Allen (2003:1218) prostý text, vstup z klávesnice nebo mluvený jazyk, který je strojově zaznamenáván.

Před procesem sloužícím k porozumění obsahu zprávy je nejprve nutno převést nestrukturovaný vstup do strukturovaného formátu. Tento formát zachycuje zprávu v přirozeném jazyce způsobem přístupným pro software sloužící k jejímu zpracování. Obsahuje například strukturu a provázanost zprávy. Za účelem tohoto formátování záznamu přirozeného jazyka využívá NLU posloupnost několika kroků – dílčích úkolů NLP nižší úrovně.

Při procesu porozumění přirozenému jazyku je také nutno počítat s určitou mírou chybovosti samotného vstupu jako například gramatické chyby a chyby v interpunkci. K problematice chyb ve vstupu se Engel (2006:195) vyjadřuje takto: *„Ale promluvy jsou často uživatelem realizovány syntakticky nesprávně (protože uživatel musí myslet a mluvit zároveň). Navíc obsahují mnoho chyb v rozpoznávání řeči v důsledku chybovosti slov, která se pohybuje mezi 7,9 % a 21,1 % (v závislosti na testovací sadě).“* Další krizovým bodem porozumění jsou slova, která mohou mít více významů. NLU má v této oblasti dosud značnou míru chybovosti, kterou však pomáhá snižovat rozvíjející možnost analýzy kontextu dané oblasti.

Stoprocentní porozumění je velmi obtížné a v mnoha případech se považuje za dostačující i pouze částečné porozumění jako například celkové vyznění zprávy – pozitivní či negativní. Často také postačí pouze extrakce hlavních bodů zprávy. K této problematice představuje Schmitt et al. (2019:67) konverzační systémy, které pro své fungování používají zjednodušenou formu porozumění přirozenému jazyku. NLU zde pro svoji práci využívá převážně dva procesy. Prvním z nich je Intent recognition neboli rozpoznání záměru. Zde probíhá analýza vstupu pro určení záměru dané zprávy. Jako druhý proces uvádí Entity recognition – rozpoznávání

entit. Zde se identifikují jednotlivé entity ze vstupu. Tyto dva procesy umožňují v konverzačních systémech porozumění zprávám na dostatečné úrovni pro formování adekvátní odpovědi. Toto částečné porozumění umožňuje, díky nižší náročnosti celého procesu, splnit požadavek konverzačních systémů na rychlé zpracování.

4.4 Detekce pojmenovaných entit

Named entity recognition (NER) neboli detekce pojmenovaných entit je podoblast zpracování přirozeného jazyka používaná pro analýzu textu a následnou detekci předem definovaných entit. NER je velmi často využíváno spolu s procesem klasifikace, který slouží k rozdělování nalezených entit do samostatných kategorií. Pro toto spojení je používána zkratka NERC a Maynard et al. (2017:25) ji ve své knize definuje takto: *„Ústředním bodem je proces detekce pojmenovaných entit a jejich klasifikace (NERC), která zahrnuje identifikaci entit v textech (NER) a jejich zařazení do množiny předem definovaných kategorií (NEC).“* Z této definice je patrné, že tato část NLP potřebuje ke svému fungování správně definované kategorie entit. Zde je nutné vyhodnotit, zda použít kategorie definované již dříve, či vytvořit nové kategorie podle vlastních specifických potřeb.

Rozdíl mezi NER a NERC je v přidané informaci v podobě zařazení entity do příslušné kategorie. NER definuje entitu v textu pomocí počáteční pozice a koncové pozice. NERC tuto definici rozšiřuje o další atribut a to danou kategorii, do které byla entita zařazena. Kategorie mohou být například: osoba, organizace, geografické místo nebo čas. Mezi další možnosti rozšíření NER patří Named entity linking (NEL). Zde je k informacím získaným NERC přidána informace o propojení dané entity se záznamem ve znalostní databázi, který poskytne podrobnější informace o nalezené entitě.

Tvorba kategorie pro NERC se skládá z ručního definování podmínek pro entity dané kategorie a následné trénování rozpoznávání entit z definované kategorie. Při trénování je nutné zvolit vhodná data, na kterých se bude NER učit rozpoznávat entity. Pro dosažení nejlepších výsledků by se data určená pro trénování měla co nejvíce blížit struktuře a typu dat, které bude NER zpracovávat v reálném provozu. Při větším množství dat k trénování dosahuje NER lepších

výsledků. Tvorba dat se skládá především z manuálního označování všech hledaných entit v celém rozsahu daného textu. Jak uvádí Altinok (2021:203), tento proces lze do určité míry automatizovat, ale stále zde zůstává velké množství manuální práce. Proto byly vyvinuty některé architektury NER, které pro trénování detekce nové entity využívají před-trénovaných modelů a pro jejich dotrénování požadují výrazně menší množství dat. Této oblasti se více věnuje kapitola trénování detekce pojmenovaných entit.

Existuje několik přístupů k NER problematice. Jako první lze jmenovat přístup založený na principu slovníku. Principem tohoto přístupu je vytvoření rozsáhlého slovníku, který bude obsahovat, co nejvíce entit. Tento přístup má několik zásadních nedostatků. Například velikost slovníku nabírá enormních rozměrů při využití pro obecné vyhledávání. Jak uvádí Song et al. (2018:22) je zde problém v zanedbávání kontextu, ve kterém se dané entita nachází. Toto vede k častému chybnému určování kategorie entity.

Přístup založený na pravidlech, jak už název napovídá, staví na manuálně definovaných pravidlech. Zde jsou vysoké nároky na znalosti osoby, která daná pravidla definuje. Eftimov et al. (2017:5) popisuje nevýhody této metody jako: *„Hlavní nevýhodou těchto metod je ruční konstrukce pravidel, což je časově náročný úkol a závisí na doméně.“* Jak je uvedeno v citaci výše, tento přístup je závislý na konkrétní doméně, pro kterou je vytvářen. Proto je značně problematické jeho využití pro definování obecnějších pravidel.

Mezi další možné přístupy patří využití strojového učení. Zde dochází k výraznému zvýšení výkonnosti a přesnosti procesu. V tomto přístupu se nevyužívá předem vytvořených slovníků, či předem definovaných pravidel. Využívá se zde strojové učení pro vytvoření algoritmu pro detekci pojmenovaných entit. Pro trénování je potřebné značné množství dat, která se skládají z textu a označení jednotlivých entit, které spadají do požadovaných kategorií. Tento přístup k datům se nazývá supervised learning – je zde určitý dohled nad samotným učením. Existují i verze unsupervised learning – zde jsou data pouze v textové formě bez jakéhokoliv označení vyhledávaných entit. Do kategorie supervised learning podle Eftimov et al. (2017:5) patří modely: decision trees, support vector machines (SVMs), hidden

Markov models (HMM), conditional random fields (CRFs) a maximum entropy. Jednotlivé modely lze kombinovat pro dosažení lepších výsledků.

Decision trees neboli rozhodovací stromy je algoritmus využívaný strojovým učením pro klasifikační úkoly. Jak už název napovídá, algoritmus má strukturu stromu a od kořenového bodu se větví pomocí if-else podmínek. Tento proces vede k postupnému rozdělování na základě vlastností zkoumaných v podmínkách a následné klasifikaci. Algoritmus pracuje rekurzivně.

Support vector machines je také nazývána metodou podpůrných vektorů. Ekbal et al. (2010:157) udává následující: *„SVM je známé pro svůj dobrý výkon při zobecňování a bylo aplikováno na mnoho problémů s rozpoznáváním vzorů.“* Základním principem tohoto přístupu je určování lineárních hranic pro pozdější kategorizaci vstupních dat. Zpravidla je zde využíváno rozšiřování dimenze prostoru zpracovávaných dat pro lepší oddělení jednotlivých kategorií.

Hidden Markov models je další z možných přístupů v oblasti strojového učení. Jedná se o pravděpodobnostní model. Vychází z principu modelování Markova řetězce, který obsahuje skryté stavy, které nelze přímo pozorovat. Dále definuje procesy závislé na Markově řetězci. Pozorováním procesů závislých na Markově řetězci dochází k analýze pravděpodobností stavů Markova řetězce. Výhodou tohoto přístupu je jeho nezávislost na konkrétním jazyce, toto mimo jiné zmiňuje i Morwal et al. (2012:15).

Conditional random fields definuje Song et al. (2018:23) takto: *„CRF je třída statistických modelovacích metod často používaných při rozpoznávání vzorů a strojovém učení, kde se používají pro predikci struktury.“* Zásadní vlastností CRF je využívání informacích o kontextu zkoumaného prvku při určování jeho kategorie. Tento proces využívá závislostí mezi jednotlivými zkoumanými prvky a tu zohledňuje do výsledné struktury. Proto dosahuje vyšší míry efektivity než HMM nebo maximum entropy modely. V přístupu využívajícím CRF jsou také vyřešeny některé problémy HMM modelů založených na podmínkách (např.: Maximum entropy model), jako například: problém se zkreslením označení entit vyplívající z omezeného množství vytvořených kategorií.

Maximum entropy neboli maximální entropie používaný v NER, je přístup založený na *„odhadování pravděpodobnosti založené na principu vytváření co*

nejmenšího počtu předpokladů jiných než předem uložených omezení“, viz blíže Chieu et al. (2002:2). Předem definovaná a uložená omezení vznikají při trénování. Reprezentují nalezené vztahy mezi vlastnostmi dat a výsledkem. Tento přístup bývá často propojován s Makovým modelem za účelem využití výhod obou modelů a celkového zvýšení výkonnosti.

Jako další skupinu přístupů k problematice Named Entity Recognition lze jmenovat skupinu přístupů založených na neuronových sítích. Jedná se o podoblast strojového učení, která byla vyvinuta díky inspiraci lidským mozkem. Základní struktura je tvořena uzly (neurony) a jejich vzájemným propojením. Jednotlivé uzly mezi sebou komunikují. Na základě inspirace neurony živých organismů i neurony v tomto procesu mají několik vstupů, ale pouze jeden výstup. Celá struktura je rozdělena na několik vrstev neuronů, které v sobě mají mimo jiné informaci o své váze a hodnotě zkreslení. Pro učení potřebují neuronové sítě velké množství kvalitních dat. Využití přístupu založeném na neuronových sítích pro NER komentuje Jin et al. (2019:136695-136696) takto: *„Pro řešení problémů NER v oblasti anglického jazyka demonstrují modely založené na neuronové síti jejich vynikající výkon při identifikaci entit.“* Do této oblasti lze zařadit modely: rekurentní neuronové sítě, konvoluční neuronové sítě a architekturu Transformer. Těmto modelům se bude tato bakalářské práce více věnovat v kapitole trénování detekce pojmenovaných entit.

4.5 Vývoj oblasti detekce pojmenovaných entit

Určení přesného počátku detekce pojmenovaných entit je problematické. Obecně lze mezi prvopočátky počítat sérii konferencí Message Understanding Conference (MUC). Jak z názvu konference vyplývá, jejím cílem bylo vyvinout nové metody s lepší přesností a vyšší rychlostí extrakce informací sloužících k porozumění dané zprávě. Část konference byla věnována testování a porovnávání jednotlivých přístupů, které účastnici vyvinuli. Na prvních konferencích se jednalo o přístupy silně specializované na jednu oblast zdrojů, teprve později došlo k zobecnění využití NER.

Pro oblast detekce pojmenovaných entit byla klíčová šestá konference MUC. Zde bylo poprvé použito slovní spojení “Named entity“ (NE) neboli pojmenovaná

entita. Li et al. (2022:50) uvádí, že toto slovní spojení bylo využito k označení: *„úkolu identifikovat jména organizací, lidí, geografických míst a také měn, času a procent v textu.“* Byly zde použity dvě kategorie: entity name expressions (ENAMEX), do které byly zahrnuty organizace, geografické lokace a osoby. Jako druhá byla pak kategorie numerical expression (NUMEX) určená pro číselné informace.

Pro oblast Named entity recognition je podstatná definice „Named entity“. Těchto definicí bylo po šesté konferenci MUC vytvořeno několik a jimi byl ovlivňován směr vývoje NER. Lze zmínit například definici „Named entity“ od Petasis et al. (2000:128) z roku 2000: *„podstatné jméno, sloužící jako jméno pro něco nebo někoho.“* V roce 2007 se Nadeau et al. (2007:3) zaměřil na přídavné jméno Named (pojmenovaný) z označení NE. Jeho definice pojmu „Named entity“ zní následovně: *„slovo „Pojmenovaný“ má za cíl omezit úkol pouze na ty entity, pro které je jeden nebo více pevných označení.“*

Mezi lety 1990 a 2010 byl vývoj přístupů k NER následující. Nejprve převládalo strojové učení využívající rozhodovací stromy. Od tohoto přístupu se postupně přesouvalo ke statistickým modelům strojového učení, mezi které patří například Hidden Markov Models nebo Conditional random field. Zde docházelo k výraznému zvýšení výkonnosti modelů. Po roce 2000 začínají postupně přicházet modely založené na neuronových sítích. Okolo roku 2010 jsou již tyto modely značně rozšířené a využívané pro jejich výrazné benefity.

Po roce 2015 se začal v oblasti NER rozšiřovat Attention mechanism (mechanismus pozornosti). Tento mechanismus pomáhá zvyšovat kvalitu práce modelů založených na neuronových sítích. Pomáhá modelu zaměřit se na podstatné části zpracovávaného textu. Na základě kontextu určuje, zda je daná část méně, či více důležitá. Ve svém výzkumu strojového překladu využívajícího neuronové sítě použili mechanismus pozornosti například Bahdanau et al. (2015).

V roce 2017 byla v díle nazvaném “Attention Is All You Need“ od Vaswani et al.(2017) představena nová architektura neuronových sítí. Nese název Transformer a přinesla mimo jiné možnost trénovat paralelně za využití grafické karty. Toto vedlo ke zvýšení rychlosti trénování nových modelů a následně umožnilo trénování modelů na řádově vyšším množství dat. Významnou částí této architektury je několik vrstev pozornosti, které zajišťují vyšší kvalitu výstupu.

V roce 2018 byl společností Google nový model reprezentace jazyka nazvaný BERT. Představen byl v práci Devlin et al. (2018) nazvané: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". Jak už název napovídá, jedná se model jazyka založený na architektuře Transfomer, který umožňuje vytvářet před-trénované modely. Tato možnost rapidně zvyšuje rychlost trénování nových modelů – stačí pouze před-trénovaný model doladit na menším množství dat.

4.6 Trénování detekce pojmenovaných entit

Složení procesu trénování detekce pojmenovaných entit závisí na zvolené architektuře NER. Jedná se o velmi nákladnou oblast NER, proto je vhodné správně zhodnotit, zda je opravdu nutné vytvářet nový model. Jednotlivé přístupy byly představeny v kapitole Detekce pojmenovaných entit. V této kapitole budou blíže přestaveny přístupy založené na neuronových sítích, jelikož se jedná o v současnosti nejvýkonnější modely. Napříč konkrétními architekturami založenými na neuronových sítích se využívá několik velmi podobných kroků trénování detekce nové entity.

Prvním z nich je sběr dat, která budou použita pro trénování nového modelu. Při výběru dat je vhodné vybírat obsahově ve stejné oblasti, v jaké bude následně NER nasazeno. K tomuto kroku Altinok (2021:202) uvádí následující: *„V kroku sběru dat se musíme rozhodnout, kolik dat shromáždit: 1 000 vět, 5 000 vět nebo více. Množství dat závisí na složitosti vašeho úkolu a doméně."* Kromě správného výběru oblasti čerpání a dostatečného množství dat je podstatné provést kontrolu kvality vybraných dat (například z gramatického hlediska). Kvalita využitých dat z velké části ovlivňuje kvalitu práce natrénovaného modelu.

Následující fází procesu trénování detekce pojmenované entity je označení všech vyhledávaných entit v nasbíraných datech. Tento proces lze částečně automatizovat, ale stále je z velké části vyžadována lidská manuální kontrola. Způsob označení entit závisí na konkrétním softwaru využívaném k trénování detekce nové entity.

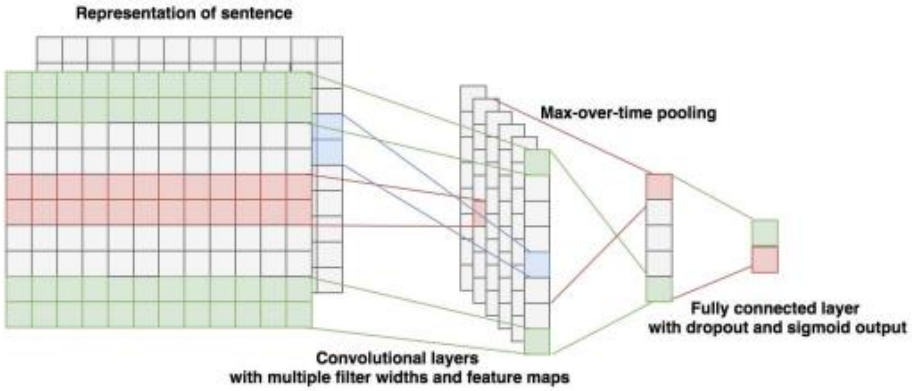
Dalším podstatným krokem je přeformátování dat s označenými entitami na formát požadovaný softwarem využívaným k trénování nového modelu. Mnoho

NLP knihoven podporuje formát JSON, některé však vyžadují formát vlastní. Například knihovna spaCy od verze 3.0 vyžaduje data ve formátu *.spacy.

Po připravení dat a nakonfigurování modelu k trénování dochází k fázi samotného trénování. Modelu jsou předána připravená data k trénování i k validaci a konfigurační soubory. Následně se model na daných datech učí.

4.6.1 CNN

Convolution neural network (CNN), neboli konvoluční neuronová síť je definována Krešňákovou a kol. (2019:145) takto: *„Konvoluční sítě jsou v jednoduchosti neuronové sítě, které používají konvoluci možných vícenásobných číselných dat v jedné z jejich vrstev.“* Data využívaná touto architekturou jsou reprezentována pomocí matic. CNN se využívá především v oblasti počítačového vidění, ale je rozšířená i v oblasti NLP. Mezi NLP knihovny, které využívají CNN ve svých modelech, patří například knihovna spaCy.



Obrázek 2: Schéma CNN
Zdroj: Krešňáková a kol. (2019:147)

Convolutional layer (konvoluční vrstva) je základem konvolučních neuronových sítí. Na obrázku číslo 2 je vidět, že je tvořena několika filtry zapsanými pomocí matic, kde každé pole má určitou váhu. Výstup této vrstvy je vytvořen vynásobením jednotlivých bodů vstupu všemi filtry. Albawi et al. (2018:4) popisuje: *„Každá z konvolučních operací je specifikována krokem, velikostí filtru a nulovou výplní.“* Tyto parametry je nutné manuálně nastavit před každým trénováním. Krok určuje vzdálenost, o kterou se má filtr posunout po vypočítání vstupu daného místa.

Jak Albawi et al. (2018:4) udává, velikost filtrů by měla být v celé vrstvě stejná. Nulová výplň zajišťuje stejné rozměry vstupu a výstupu.

Podle obrázku číslo 2 následuje po konvoluční vrstvě vrstva nazvaná pooling neboli sdružování. I zde jsou na data aplikovány filtry, ale jejich prvky nejsou vážené. Účelem této vrstvy je snížit rozměry dat. Existuje několik druhů poolingu například: sčítání, průměrování a maximální hodnota.

Poslední vrstva je Fully connected layer (Plně propojená vrstva). V této vrstvě má každý neuron spojení na každý neuron z následující vrstvy. Zde dochází ke klasifikaci jednotlivých prvků na základě dat z celého procesu. Dle Yamashita et al. (2018:616) je výstup ve tvaru pravděpodobností příslušnosti (od 0 do 1) do jednotlivých kategorií a jejich sečtení dává dohromady 1.

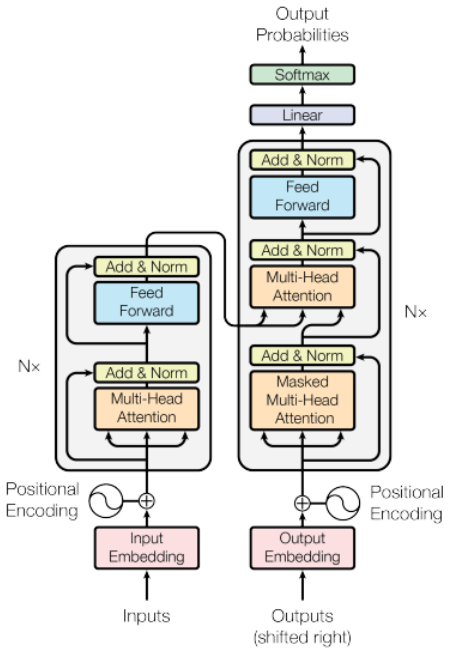
4.6.2 RNN

Recurrent neural network (RNN) neboli rekurentní neuronová síť je architektura inspirovaná biologickým mozkem využívaná převážně v oblasti NLP a počítačovém vidění. U spojení, kterými jsou propojovány jednotlivé neurony, je v této architektuře možné vytvořit smyčku – spojit neuron sám se sebou. Toto vede k možnosti neuronu využít svůj výstup jako část svého následujícího vstupu. Data jsou zde zpracovávána sekvenčně. Je zde problém s kvalitou zpracování delších sekvencí, proto byly vyvinuty další architektury, které tento problém řeší. Lze jmenovat například Long short-term memory (LSTM).

4.6.3 Transformer

Transformer je architektura neuronových sítí představená v roce 2017. Jejím základem je mechanismus pozornosti.

Na obrázku číslo 3 je možné vidět, že Transformer využívá mechanismus pozornosti v průběhu svého procesu celkem třikrát a to ve formě multi-head attention. Jedná se o několik vrstev pozornosti běžících paralelně Vaswani et al. (2017:4). Jednou z hlavních výhod této architektury je možnost zpracovávání celého vstupu najednou. Tímto se liší od předešlých architektur, kde se vstup zpracovával převážně po částech.



Obrázek 3: Architektura Transformer
Zdroj: Vaswani et al. (2017:3)

Dále umožňuje paralelizaci celého procesu a využití GPU, tím dochází ke zvýšení výkonnosti modelu i rychlosti jeho trénování. Transformer se skládá s kodéru – na obrázku číslo 3 se jedná o levou větev a dekodéru – na obrázku číslo 3 pravá větev. Jednotlivé tokeny vstupu jsou v kodéru převáděny na vektory, se kterými poté pracuje dekodér.

5 Praktická část

Praktická část této bakalářské práce je rozdělena do tří částí. V první části je stručně představena knihovna spaCy. Následně je provedena analýza webových stránek knihovny, natrénovaných modelů a samotného kódu knihovny. Druhá část shrnuje problematiku tvorby extraktoru pojmenovaných entit s využitím spaCy. V závěru druhé části je zhodnoceno využívání této knihovny pro vývoj aplikace. Poslední část je věnována trénování detekce nové kategorie pojmenovaných entit. Zde je nastíněn proces přípravy dat, následně trénování nových modelů a jejich porovnání. Poslední část uzavírá zhodnocení používání knihovny spaCy pro vývoj nových modelů.

5.1 Analýza knihovny spaCy

SpaCy je open-source knihovna poskytující sadu nástrojů pro práci v oblasti zpracování přirozeného jazyka. Byla vyvinuta a v roce 2015 představena společností Explosion AI sídlící v Německu. Od ostatních knihoven v této oblasti se liší svým zaměřením – cílí především na využití při výrobě komerčního softwaru. Ostatní knihovny jsou zaměřeny především na využití pro akademické a vědecké účely. Jedná se o jednu z nejpoužívanějších knihoven pro oblast NLP. Podle stránek Explosion AI (2022) překročil počet stažení spaCy 100 milionů. Pro její napsání zvolili zakladatelé Matthew Honnibal a Ines Montani programovací jazyky Python a Cython. Využití těchto jazyků je v této oblasti velmi výhodné, jelikož oba jazyky excelují v práci s velkým množstvím dat a především Cython je znám pro svoji výkonnost. Společnost Explosion AI vyvinula také Prodigy. Tento program slouží k anotaci dat určených k trénování nových modelů. Díky vývoji stejnou firmou je kvalitně uzpůsoben pro propojení se spaCy. Tyto dva programy dohromady poskytují dostatečné pokrytí oblasti NLP pro komerční vývoj.

V roce 2021 byla představena verze 3.0, která přinesla několik zásadních změn. Nejpodstatnější z nich bylo přidání modelů založených na architektuře Transformer, která výrazně zlepšila kvalitu práce knihovny spaCy. Dále zde bylo přidáno propojení s platformou HuggingFace, díky které bylo nově možné trénovat vlastní modely založené na Transformer architektuře. V době psaní této práce byla

k dispozici nejnovější verze 3.4, která přinesla několik výkonnostních vylepšení, nové vektory pro anglický CNN model a nový model pro chorvatštinu.

5.1.1 Analýza webových stránek knihovny spaCy

Webové stránky spaCy prošly v roce 2021 s novou verzí knihovny velkou proměnou. Pro udržení zpětné kompatibility byla stará verze zachována a pouze přesunuta na jinou adresu. Na obrázku číslo 4 je vidět aktuální struktura webových stránek.

Obrázek 4: Struktura webových stránek spaCy
Zdroj: spaCy (2022)

Pod nápisem spaCy se nachází úvodní stránka. Zde je knihovna představována s jasným záměrem přesvědčit čtenáře k jejímu vyzkoušení. Využity jsou interaktivní prvky. Například okno s demo aplikací, kterou je možné vyzkoušet. Dále se zde nachází výčet funkcí knihovny, výsledky výkonnostních testů vybraných modelů a představení možnosti vytvořit si vlastní model.

Záložka Usage skrývá několik podkategorií. Jsou v nich podrobněji představeny informace z úvodní stránky. První z nich je stručný průvodce instalací knihovny. Následuje krátké představení poskytovaných modelů a podporovaných jazyků spolu s návodem na jejich instalaci a první použití. Dále je zde záložka s fakty o knihovně například výsledky výkonnostních testů. Mezi další záložky patří podrobnější představení spaCy nazvané “spaCy 101“ a přehled několika posledních aktualizací včetně změn, které přinesly. Zbylá část záložky Usage je věnována podrobnějším návodům na jednotlivé funkce knihovny.

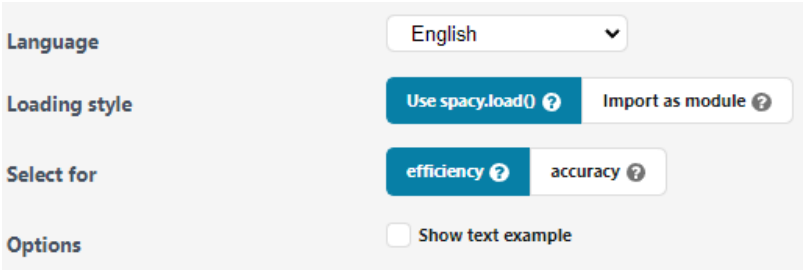
Následuje záložka Models. Jak už z názvu vyplývá, jsou zde představeny všechny před-trénované modely, které spaCy nabízí. V úvodu je shrnut postup instalace spolu se stručným představením principů pojmenování balíčků modelů a nastínění použité architektury modelů. Dále je zde detailně představen každý model z aktuálně 24 (2022) podporovaných jazyků. Včetně modelu multi-language, který podporuje několik jazyků najednou.

Api je sekce, kde je detailně popsána každá část knihovny spaCy. Je zde představena architektura knihovny, datové formáty, funkce, kontejnery a natrénované pipeliny. U každé pipeliny jsou popsány všechny její funkce a možnosti konfigurace.

V záložce nazvané Universe se nachází sekce shrnující *„zdroje vyvinuté s nebo pro spaCy. Zahrnuje samostatné balíčky, pluginy, rozšíření, vzdělávací materiály, provozní nástroje a vazby pro jiné jazyky.“* spaCy (2022). Díky open-source formátu knihovny jsou zde stručně představeny aplikace využívající spaCy spolu s možností jejich využití.

5.1.2 Analýza natrénovaných modelů knihovny spaCy

Knihovna spaCy nabízí velké množství před-trénovaných modelů. Pro jejich využití je nejprve nutné si dané modely stáhnout. První možností je využít instalačního formuláře na webových stránkách spaCy v záložce Usage. Zde se jedná především o instalaci samotné knihovny, ale lze k ní navolit i instalaci modelů pro vybrané jazyky. Druhá možnost je k dispozici v záložce Models, kde se nachází obdobný formulář jako na záložce Usage. Jeho struktura je zachycena na obrázku číslo 5. V obou formulářích je možné vybrat, pro který jazyk bude daný model stažen. Dále je zde možnost zvolit preference daného modelu. Možnosti jsou: zaměření modelu na efektivitu (rychlejší a menší model, co se týče objemu dat) za cenu nižší přesnosti prováděných operací nebo zaměření na přesnost (pomalejší a objemnější model).



Obrázek 5: Instalační formulář
Zdroj: spaCy(2022)

Pro verzi zaměřenou na efektivitu dává spaCy k dispozici verzi modelu označenou jako “sm“ – jedná se o zkratku small (malý). U verze zaměřené na přesnost záleží na zvoleném jazyce. U jazyků, kde je již vytvořený model Transformer, je zvolen tento model a u jazyků, kde dosud tento model není, uživatel stahuje verzi označenou jako “lg“ - zkratka large (velký). Toto však nejsou všechny modely, které má spaCy k dispozici. Pro přístup ke všem verzím je nutné přejít na záložku konkrétního jazyka. Zde se nacházejí všechny poskytované modely pro daný jazyk.

Jak již bylo výše zmíněno, jednotlivé modely jsou systematicky pojmenovány. Části jsou v názvu odděleny podtržítkem. První část názvu označuje zkratku daného jazyka, pro který byl trénován, například “en“ pro angličtinu. V následující části označující typ modelu jsou dvě varianty. První je nazvaná “core“ neboli jádro. Jedná se o obecnější verzi modelu. Druhá možnost “dep“ neobsahuje modul NER a je zaměřená především na závislosti. Dále se v názvu nachází část označující zdroj dat využitých k trénování daného modelu. Nejčastěji se jedná o zkratku “web“ – zdrojem byly materiály z webových stránek. Poslední část názvu modelu definuje velikost a použitou architekturu. SpaCy je označuje jako “sm“ pro nejmenší modely (zde se velikost pohybuje od 10MB do 20MB), “md“ pro střední velikost (tyto modely se pohybují od 40MB do 80MB) a “lg“ pro největší (rozpětí velikosti je zde od 200MB do 600MB), co se týče objemu dat. Platí zde pravidlo: čím větší velikost, tím přesnější model. Všechny výše zmíněné kategorie využívají architekturu CNN. Dále je zde verze označená zkratkou “trf“. Jedná se o verzi postavené na architektuře Transformer. Objemem dat se pohybuje okolo velikosti verze “lg“. Ne všechny jazyky mají k dispozici tuto variantu.

Bližší informace o jednotlivých modelech se nachází v záložce každého jazyka. Jsou zde představeny všechny před-trénované modely pro daný jazyk a každý model je detailně popsán. Nachází se zde informace o nejnovější verzi modelu spolu s odkazem na GitHub větev, kde byl nahrán. Dále je zde popsána oblast, ze které byly čerpány data pro trénování spolu s odkazy na konkrétní zdroje. U každého modelu se nachází i výčet komponent, které obsahuje (ne všechny modely obsahují stejné komponenty). Dále se uvádí množství použitých vektorů (u verze “sm“ se vektory nepoužívají). Následují obecnější informace o autorovi a licenci, pod kterou je daný model distribuován. Důležitou částí je sekce představující výsledky

testování přesnosti jednotlivých částí daného model. Celý popis uzavírá část představující schéma popisků používaných v daném modelu. Tato sekce je velmi užitečná, jelikož se často stává, že jednotlivé před-trénované modely označují stejnou entitu jiným popiskem než ostatní.

V době psaní této práce (2022) poskytuje spaCy modely pro 24 jazyků. Jejich rozsah a kvalita se u jednotlivých jazyků velmi liší. Některé nemají k dispozici model založený na Transformer. Dále se liší například poskytovanými komponentami nebo jejich funkcemi. Pro tuto bakalářskou práce je podstatné, že jednotlivé modely poskytují různou škálu kategorií pro detekci pojmenovaných entit. Například pro portugalštinu jsou k dispozici 4 kategorie, oproti tomu pro japonštinu jich je 22. Model Multi-language má nižší kvalitu výstupu i omezenější funkce, jelikož se jedná o více jazykových problematik zároveň.

Pro značné množství světových jazyků dosud spaCy nenabízí před-trénované modely, ale pro dalších více než 72 jazyků poskytuje podporu pro vytvoření vlastních modelů. V poslední praktické části této práce bude využito přítomnosti češtiny mezi těmito podporovanými jazyky.

5.1.3 Analýza kódu pro využití knihovny spaCy

SpaCy je knihovna napsaná v programovacích jazycích Python a Cython. Mezi jejich přednosti patří například minimální množství kódu oproti dalším programovacím jazykům, jako je Java nebo C#. Touto cestou se vydala i samotná knihovna, a proto je kód potřebný k jejímu využívání velmi stručný a přehledný. Na obrázku číslo 6 je ukázán zdrojový kód využitý při procesu detekce pojmenovaných entit. Úspornost kódu knihovny dokazuje, že na celý proces stačilo 5 řádků kódu. Kromě této úsporné verze vhodné především k rychlému a efektivnímu využití nabízí spaCy i možnost manuální konfigurace jednotlivých částí procesu. Zde se však rapidně zvětšuje objem potřebného kódu i složitost celého procesu. I v tomto případě se však stále jedná o relativně kompaktní množství kódu. SpaCy nabízí jak verzi kódu pro okamžité nasazení, která odstiňuje uživatele od složitosti zpracovávané oblasti, tak verzi poskytují prostor pro složitější konfiguraci. Díky tomu ji lze využít i v případech vyžadujících velmi specifickou funkcionalitu. Jelikož

byly k implementaci knihovny zvoleny objektově založené programovací jazyky, i samotný kód využití knihovny je objektově orientovaný.

```
import spacy

nlp = spacy.load("en_core_web_lg")
doc = nlp("Input text...")

for ent in doc.ents:
    print(ent.text + " - " + ent.label + " " + ent.start_char + ":" + ent.end_char)
```

Obrázek 6: Příklad využití spaCy
Zdroj: autor

Na obrázku číslo 6 je představen klasický postup při využití knihovny spaCy ve vlastním kódu. Nejprve je nutno samotnou knihovnu importovat do daného projektu. Následně se vytváří objekt, skrze který je spaCy volána, a do něhož se načte konkrétní jazykový model. Zde se mohou načíst modely oficiálně distribuované i modely soukromě vytvořené. Dále je potřebné vytvořit objekt dokumentu, který je naplněn výsledky celého procesu. Zde se využije dříve vytvořený objekt spaCy a jako parametr zpracovávaný text. Poté lze z daného dokumentu zobrazovat požadovaná data.

5.2 Vývoj extraktoru

Tato část bakalářské práce se zabývá vývojem extraktoru pojmenovaných entit za využití knihovny spaCy. Jsou zde také stručně popsány technologie využité při samotném vývoji. Dále tato část obsahuje shrnutí poznatků z používání knihovny spaCy během celého procesu.

Primárním cílem této praktické části je vytvořit aplikaci, na které budou představeny možnosti detekce pojmenovaných entit poskytované knihovnou spaCy. V aplikaci bude k dispozici 23 před-trénovaných modelů pro knihovnou podporované jazyky a dále model pro český jazyk natrénovaný autorem za využití této knihovny.

Jazykem, ve kterém budou napsány všechny texty uvnitř aplikace, bude angličtina. Tímto bude docíleno vyššího dosahu aplikace. Oproti českému jazyku

bude mít přístup k aplikaci ve svém rodném jazyce výrazně vyšší množství uživatelů. V současné době je již standardem, že jsou nové aplikace plně responzivní (jejich obsah se pružně přizpůsobuje velikosti obrazovky používaného zařízení a jejich obsah lze tudíž zobrazovat nejen na osobních počítačích). Tento standard bude dodržen i u této aplikace. Důsledkem tohoto kroku bude opět zvýšený dosah aplikace, jelikož bude přístupná na osobních počítačích, tabletech, mobilních zařízení i na všech dalších zařízeních, která obsahují internetový prohlížeč.

Posledních několik let je v oblasti informačních technologií trendem přecházet z desktopových aplikací na webové, proto i tato aplikace bude vytvořena jako webová. Tento typ aplikace byl zvolen, protože jeho využití sebou nese mnoho benefitů. Mezi hlavní z nich lze jmenovat přístup k aplikaci odkudkoliv a skoro z jakéhokoliv zařízení, dále odstínění uživatele od aktualizací daného softwaru nebo například snížení nároků na uživatelův hardware. S nasazením aplikace jako webové je spojeno vybrání vhodného názvu, aby byl pro uživatele snadno zapamatovatelný a zároveň ji dobře popisoval. Proto byl zvolen stručný a aplikaci vystihující název: “NER_It“.

5.2.1 Použité technologie

Jelikož je aplikace vyvíjena jako webová, bude se skládat z front-end části, která poběží na straně klienta a back-end části, která poběží na serveru. Pro každou z nich bude využit jiný programovací jazyk (případně jeho framework) a zároveň jiný podpůrný software.

Pro naprogramování front-end části aplikace bude využit programovací jazyk JavaScript konkrétně jeho knihovna React. Jedná se o knihovnu vyvíjenou společností Meta Platforms sloužící k tvorbě uživatelských rozhraní. Poskytuje uživateli množství prostředků pro tvorbu webové aplikace typu single page – jednostránková aplikace. Jak už název napovídá, tento přístup staví na webovém prostoru, který má pouze jednu stránku. Dynamika je zde tvořena přepisováním částí stránky bez nutnosti načítat celý obsah znovu. Této vlastnosti bude využito pro ukládání historie vyhledávání v aplikaci tvořené v této bakalářské práci. Historie bude ukládána po dobu jednoho načtení stránky a při jejím znovu načtení bude vymazána.

Jelikož je hlavním cílem této aplikace představit knihovnu spaCy, byl i tomuto cíli podřízen výběr programovacího jazyka pro back-end část aplikace. Knihovna spaCy je napsaná v jazyce Python, proto bylo vhodné využít stejný jazyk i pro server část aplikace, kde bude knihovna využívána. Pro Python se nabízí dva hlavní frameworky a to: Django a Flask. Jelikož nebudou v back-end části této aplikace vykonávány příliš složité operace, je vhodnější varianta framework Flask. Dalším důvodem zvolení Flasku před Djangem jsou jeho kompaktnější rozměry. Tato vlastnost je klíčová, jelikož je důležité udržet velikost serverové části aplikace co nejmenší. Na serveru budou nasazeny všechny modely knihovny spaCy, které i přes využití nejmenších možných formátů budou dosahovat řádově nižších stovek megabytů.

Jako další budou v této aplikaci využity jazyky HTML a CSS. HTML bude využito převážně pro vytvoření rámce aplikace pro vložení jádra napsaného v Reactu. CSS bude použito pro definování vzhledu celé aplikace. Poskytuje plnou kontrolu nad výsledným zobrazováním a zároveň umožňuje oddělit části kódu, které jsou využity pro strukturu a části definující vzhled. Toto vede k zvýšení přehlednosti zdrojového kódu.

Podstatnou částí funkční stránky aplikace bude knihovna spaCy, která je detailně popsána výše v této práci. Dále bude v této aplikaci využíván Gunicorn, který se využívá pro produkční nasazení aplikací napsaných ve Flasku. Jedná se o poměrně kompaktní WSGI server, řešící komunikaci mezi webovou aplikací a serverem.

Pro samotný vývoj aplikace byl zvolen PyCharm Professional. Jedná se o komplexní IDE (Integrated Development Environment neboli Integrované vývojové prostředí) od společnosti JetBrains. Placená verze Professional umožňuje, na rozdíl od bezplatné verze Community, pracovat i s jinými jazyky než jen Python. Podporuje HTML, JavaScript a SQL. PyCharm byl zvolen pro svou širokou škálu zabudovaných podpůrných funkcí usnadňujících vývoj webových aplikací.

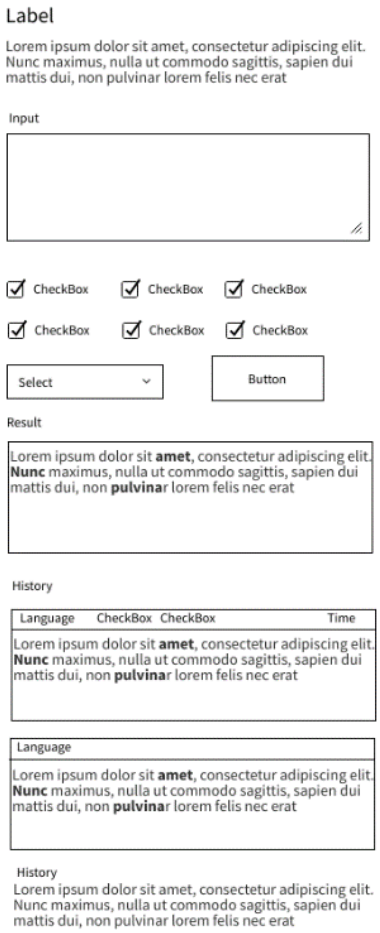
Pro snazší manipulaci s aplikacemi nasazenými na serverech Heroku byl využíván Heroku CLI (Command Line Interface – Rozhraní příkazové řádky). Jedná se o rozšíření příkazové řádky umožňující manipulaci s aplikacemi umístěnými na stejnojmenné platformě přímo z příkazové řádky vývojového prostředí. Při vývoji

webové aplikace se jedná o častý úkon, proto je toto usnadnění velmi užitečné. Používání Heroku CLI je spojené s verzovacím systémem GIT, který využívá. Tento verzovací systém dále pomáhá se zálohováním aplikace během jejího vývoje a umožňuje vracet se k jejím jednotlivým verzím.

5.2.2 Vývoj front-end části extraktoru

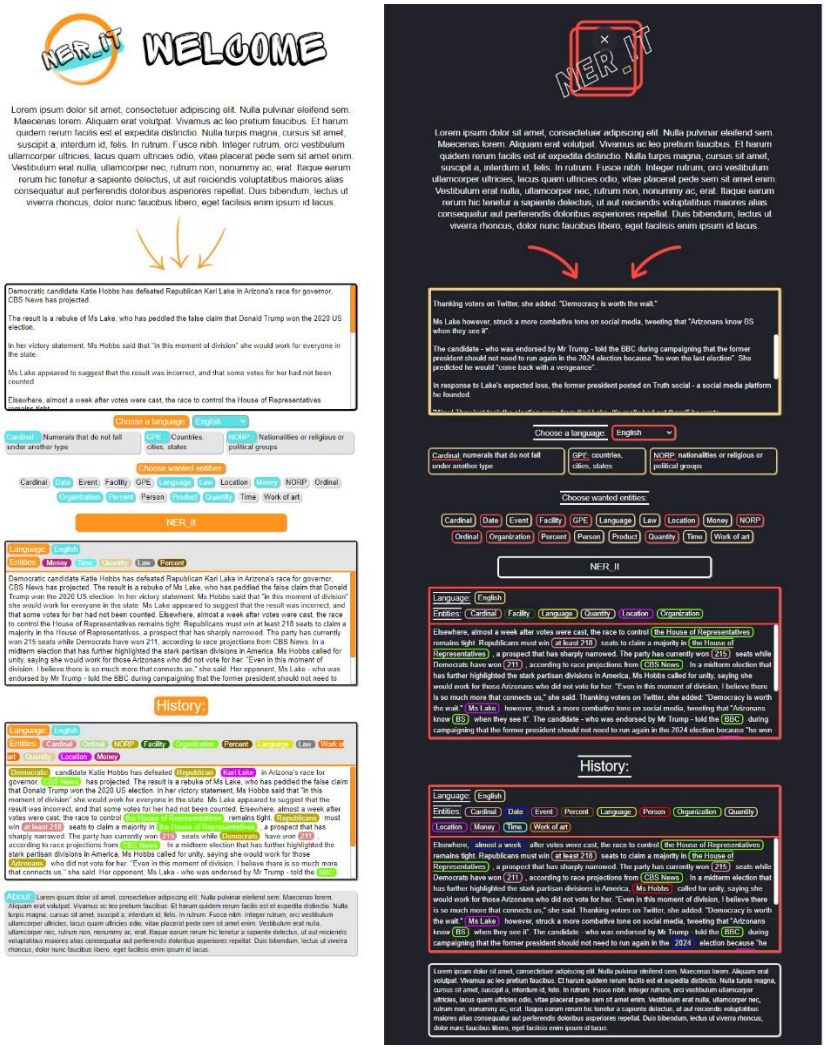
Vývoj front-end části aplikace se skládá z několika fází. Mezi jednotlivé fáze patří: definování požadavků na danou aplikaci a následný výběr vhodných přístupů, grafický návrh uživatelského rozhraní, návrh vnitřní struktury aplikace a samotný vývoj. Výčet není obsáhlý, jelikož se jedná o relativně menší aplikaci. Přestože požadavky na aplikaci zůstávaly v průběhu celého vývoje stejné, docházelo k menším změnám ve zvolených přístupech či v grafické oblasti. Veškeré materiály potřebné pro front-end část extraktoru jsou uloženy v adresáři aplikace nazvaném "client".

Hlavním požadavkem na vyvíjenou aplikaci bylo představení možností, které poskytuje knihovna spaCy v oblasti NER. Dále byla požadována snadná přístupnost aplikace spolu s intuitivním ovládáním. Na základě těchto požadavků bylo vyhodnoceno, že se bude jednat o webovou aplikaci. Zde bude dosaženo snadné přístupnosti aplikace pro uživatele. Jelikož nebude nutné obsah stránky složitě větvit, bude použit formát one-page (webová stránka obsahující pouze jednu HTML stránku), který zjednoduší využívání vyvíjené aplikace. Pro lepší přístupnost pro různě veliká zařízení bude rozložení aplikace plně responzivní. Budou vytvořeny tři velikostní varianty aplikace. První varianta bude pro zařízení se šířkou obrazovky větší než 1280px. Jednat se bude o větší notebooky a monitory. Druhá varianta bude pro zařízení se šířkou v rozmezí od 1280px do 600px. V této kategorie se bude jednat především o tablety. Poslední bude mobilní verze pro zařízení se šířkou obrazovky do 600px. Dlouhodobá historie výsledků nebyla požadována, proto byla zvolena verze ukládání výsledků aplikace po dobu jednoho načtení aplikace. K tomuto účelu byla vybrána knihovna React, která umožňuje překreslovat jednotlivé komponenty bez nutnosti nového načtení stránky.



Obrázek 7: Návrh rozložení komponent
Zdroj: autor

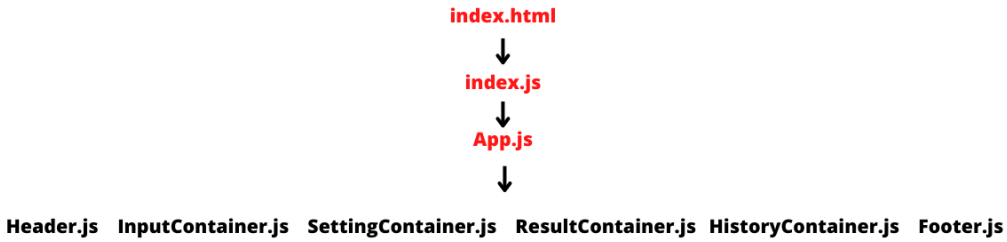
Na obrázku číslo 7 je představen první návrh rozložení komponent v aplikaci. Přesné rozložení finální verze se od prvotního návrhu liší především v oblasti nastavení vyhledávání. Změny se týkají pouze vnitřní struktury komponent, nikoli však jejich umístění uvnitř aplikace. Návrh byl koncipován tak, aby uživatele plynule prováděl procesem detekce pojmenovaných entit. První oblastí je stručné představení aplikace a problematiky NER, které uživateli objasní následující obsah stránky. Dále je zde vstupní pole pro zadávání textu ke zpracování. Následně má uživatel možnost zvolit požadovaný jazyk a vybrat konkrétní kategorie entit, které mají být v textu vyhledány. Další komponenta zobrazuje výsledek vyhledávání. Následující oblast poskytuje historie aplikace po dobu jejího načtení. Jako poslední se v návrhu nachází patička aplikace, ve které se bude nacházet stručný přehled informací o aplikaci.



Obrázek 8: Grafické varianty aplikace
Zdroj: autor

Na obrázku číslo 8 jsou představeny dvě zvažované grafické varianty aplikace. První varianta (na obrázku číslo 8 vlevo) byla vyhodnocena jako příliš barevná, a proto byla vytvořena verze druhá (na obrázku číslo 8 vpravo) s využitím méně výrazných barev. Další výhodou druhé varianty je její tmavé provedení, které bude výrazně šetřit oči uživatelů a v případě využití OLED displeje i baterii zařízení. V části aplikace, která představuje výsledky NER, bylo z grafického hlediska problematické rozlišit veliké množství kategorií pojmenovaných entit. V nejkomplexnějších modelech se jedná až o 22 možností. Zde bylo nutné využít mnohem pestřejší škálu barev, než které se nacházely v paletě barev používané v ostatních částech aplikace. Cílem celého procesu grafického návrhu bylo

uživatelské prostředí, které bude zvyšovat atraktivitu aplikace a její přehlednost. Paleta barev využívaných v aplikaci byla sjednocena s grafickým zpracováním loga aplikace.



Obrázek 9: Struktura front-end části aplikace
Zdroj: autor

Na obrázku číslo 9 je představena struktura front-end části vyvíjené aplikace. Lze zde vidět vnořování jednotlivých komponent aplikace. Červeně označená část je struktura automaticky vygenerovaná příkazem*: npx create-react-app nerIt* vloženým do konzole. Tento příkaz se používá pro automatické vytvoření základní struktury aplikace založené na knihovně React. Dále vytvoří dva adresáře public a src, do kterých jednotlivé komponenty rozdělí. Index.html se nachází v adresáři public. Index.js a App.js jsou uloženy ve složce src. Černě označená část na obrázku číslo 8 značí rozvržení aplikace navrhnuté autorem. Ke každé komponentě aplikace byl také vytvořen CSS soubor definující její vzhled.

První část stránky index.hml představuje jedinou HTML stránku aplikace. Zde jsou definovány základní atributy stránky. Jedná se především o titulek stránky, ikonu stránky, uvedení autora stránky a popisu. Nachází se zde také nastavení viewportu, které je důležitou součástí responzivního designu. Především je zde definován vstup aplikace pomocí elementu div s atributem id= “root“.

Index.js je část aplikace, ve které je definována instance ReactDOM (Document Object Model) kontejneru. Je vytvořen pomocí funkce createRoot() volané na třídě ReactDOM. Jako parametr funkce je zde vloženo id, které slouží k identifikaci HTML elementu, ve kterém má být daný objekt vykreslen. Skrze tento kontejner je zobrazována celá aplikace. Do něho je vložena hlavní komponenta aplikace App.js

App.js tvoří rámce celé aplikace a jsou v něm vnořeny všechny další části aplikace. Vložené komponenty je nutné nejprve importovat. Tyto komponenty jsou zde seřazeny podle požadovaného pořadí při vykreslení. Jsou zde definovány kontexty využívané v dalších částech aplikace. V App.css se nachází styly jednotné pro celou aplikaci. Jedná se především o odsazení rámce aplikace od krajů obrazovky.

První komponentou v App.js je Header.js. Skládá se z loga a úvodního textu. Logo je vloženo pomocí HTML tagu <img/>. Text se nachází v tagu označující paragraf <p>. Grafický vzhled a umístění obou elementů je definováno v souboru Header.css. Mimo to je zde definována i samotná komponenta Header především po stránce rozložení.

Dále se v App.js nachází InputContainer.js. Jedná se o kontejner sdružující obrázek šipek směřujících pozornost uživatele na vstupní pole, které je druhou částí tohoto kontejneru. Pro vstup byla zvolena HTML komponenta <textarea>, u které byla z grafických důvodů vypnuta kontrola pravopisu. Vše v této části aplikace je stylováno pomocí InputContainer.css.

Nejobsáhlejší částí App.js je komponenta SettingContainer.js. Nachází se v ní veškeré nastavení detekce pojmenovaných entit, které je uživateli k dispozici. První možností je volba jazyka, ve kterém je zadaný zpracovávaný text. Je zde na výběr z 24 jazyků. Pro tuto funkci byla využita HTML element <select>. V následující části se nachází všechny možné kategorie NER pro vybraný jazyk. Pro velké množství možností byla pro tuto část vytvořena funkce mapující jednotlivé kategorie na HTML elementy <input> s atributem type="checkbox". Dále je v této komponentě vnořen AbbreviationsContainer.js, ve kterém se nachází vysvětlivky všech zkratek použitých u daného jazyka. Oblast nastavení uzavírá tlačítko potvrzující nastavení, které spustí vyhledávání pojmenovaných entit. Vzhled je definován v souboru SettingContainer.css.

Výsledek NER se zobrazuje v komponentě ResultContainer.js. Do tohoto kontejneru je vložena komponenta Result.js, ve které je opravdu zobrazen výsledek NER. Samotný kontejner dodává rámec celému výsledku a přidává další informace o vyhledávání jako například jazyk a kategorie entit zvolené pro dané vyhledávání. ResultContainer.css udává parametry vzhledu tohoto kontejneru.

Předposlední částí aplikace je HistoryContainer.js. Zde se nachází historie výsledků NER provedených po jedno načtení aplikace. V této oblasti je využito znovupoužitelnosti komponent. Výsledky jsou zde mapovány na seznam komponent ResultContainer.js, který je dále zobrazován. V HistoryContainer.css je definován pouze vzhled nadpisu oblasti a případné hlášky, která signalizuje, že v aplikaci nejsou žádná předešlá vyhledávání.

Poslední oblastí App.js je Footer.js. V této komponentě se nachází jediný HTML paragram s textem obsahujícím základní informace o aplikaci. Vzhledem k jednoduchosti komponenty je i Footer.css stručný. Definuje především styl použitého písma.

5.2.3 Vývoj back-end části extraktoru

Veškeré materiály pro potřeby back-endové části extraktoru jsou uloženy v adresáři aplikace nazvaném "server". Jedná se o část aplikace, která poběží na serveru. Z důvodu jeho omezeného výkonu byl back-end rozdělen na 4 samostatné části, kde každá část byla nasazena na jiném serveru. Více se touto problematikou zabývá kapitola „Nasazení extraktoru". Pro všechny části jsou společné následující charakteristiky. Byl zde využit jazyk Python, konkrétně jeho frameworku Flask. Základní struktura byla vygenerována automaticky za využití IDE Pycharm. Nachází se zde například i venv (virtual environment) nástroj umožňující spravovat a vytvářet izolovaná prostředí pro jednotlivé projekty. Dále byl do struktury přidán spaCy model natrénovaný autorem a další soubory potřebné pro nasazení aplikace na server. Patří mezi ně především soubor Procfile, který definuje příkazy, které mají být na serveru vykonány při spuštění aplikace. Dále soubor requirements.txt, do kterého jsou vygenerovány všechny závislosti aplikace. Hlavní částí je soubor nazvaný app.py, ve kterém se nachází veškerý zdrojový kód back-end části aplikace.

```python
from flask import Flask, request
from flask_cors import CORS
import spacy

app = Flask(__name__)
CORS(app)

nerCa = spacy.load("ca_core_news_sm")

@app.route('/catalan', methods=["POST"])
def catalan():
    return nerCa(request.json['text']).to_json()


if __name__ == '__main__':
    app.run(debug=True)
```

Obrázek 10: Ukázka back-end
Zdroj: autor

Na obrázku číslo 10 je znázorněna struktura kódu back-endu. Tento kód specificky by poskytoval funkci detekce pojmenovaných entit pro katalánštinu a přijímal by parametr textu určeného ke zpracování. Pro vyvíjenou aplikaci se zdrojový kód liší od obrázku číslo 10 především v množství. Jednotlivé části pro každý jazyk však zůstávají stejné. Jedná se o vytvoření instance spaCy. Následně načtení požadovaného modelu. Poté vytvoření a definování routy (cesty), která bude poskytovat požadovanou službu.

Mezi části kódu, které se ve zdrojovém kódu neopakují, patří importování Flasku a request z balíčku flask. Dále importování spaCy a CORS, který slouží ke sdílení mezi různými doménami. Na začátku zdrojového kódu back-end části je vytvořena instance Flask aplikace a následně je zabalena do objektu CORS. V závěru je definována podmínka spuštění aplikace.

Pro všech 24 poskytovaných modelů jsou vytvořeny instance spaCy. Jejich pojmenování se skládá z označení akce, pro kterou budou využívány – NER a zkratky jazyka, pro který budou používány. Následně je do každé instance načten požadovaný model NER.

Pro každý model byla vytvořena cesta a k ní definován proces poskytování požadované funkce. Jak již bylo zmíněno v analýze kódu knihovny spaCy, její kód je

velmi úsporný, proto i kód jednotlivých cest je poměrně krátký. Nachází se v něm volání instance spaCy pro daný jazyk s parametrem textu, který je získáván z requestu (žádosti). Výsledek této operace je převeden na formát JSON a vracen jako odpověď serveru.

5.2.4 Nasazení extraktoru

První fází nasazení extraktoru je výběr vhodné platformy, na které bude aplikace fungovat. V době energetické krize se autor setkal u poskytovatelů platforem s trendem snižování výkonnosti a velikosti poskytovaného uložiště u bezplatných účtů. Z tohoto důvodu, a z také z hlediska stability, byla využita placená verze platformy. Jako konkrétní typ byl vybrán PaaS (Platform as a Service), která poskytne vše potřebné pro nasazení celé aplikace na jednom místě. V back-end části aplikace má být nasazeno 24 modelů knihovny spaCy, které jsou značně objemné. Proto bylo další podmínkou na platformu dostatečná poskytovaná velikost úložiště. Mezi zvažované poskytovatele patřili Render, Heroku a Cyclic. Z těchto možností nabízel nejlepší podmínky poskytovatel Cyclic. Jedná se o poměrně novou firmu založenou v roce 2021, která cílí na poskytnutí alternativy k již zavedenému řešení poskytované Heroku. Zde se však vyskytl problém s podporou aplikací napsaných v programovacím jazyce Python. Podpora tohoto jazyka byla v době nasazení aplikace teprve ve vývoji a jeho podpora byla pouze v beta verzi, která nebyla dostatečně stabilní. Render poskytoval lepší cenové nabídky než Heroku, ale samotný servis byl značně omezenější než u jeho konkurentů. Z těchto důvodů byl nakonec vybrán poskytovatel Heroku, který i přes výrazně vyšší ceny nabízel kvalitní a stabilní platformy. Provoz samotných modelů knihovny spaCy je velmi náročný na kapacitu operační paměti. Z tohoto důvodu, a také z důvodu vysokých cen výkonnějších serverů bylo přistoupeno k rozdělení back-endové části aplikace na 4 samostatné části. Každá z nich byla nasazena na samostatném serveru.

Před samotným nasazením bylo nutné provést několik příprav. U front-end části aplikace se jednalo především o definování enginů (jáder aplikace). Konkrétně zde byla definována verze využívaného Nodu a npm. Tato definice se nachází v souboru package.json. Ve stejném souboru bylo dále nutné definovat skripty na postavení a spuštění aplikace, které budou na serveru využívány. U back-end části

aplikace byla vytvořena složka Procfile, ve které byly definovány příkazy, které má Heroku provést při spouštění aplikace. Dále bylo nutné vygenerovat složku requirements.txt, ve které bude Heroku předán seznam závislostí aplikace. Jako poslední fáze byla instalace Gunicorn jako WSGI serveru.

Proces nasazení aplikace se skládá z několika kroků. Jednotlivé části aplikace byly nasazeny zvlášť, ale tento proces byl využit ve všech případech. V prvním z nich je potřeba vytvořit účet na Heroku, pod kterým bude aplikace spravována. Dále se provede založení instance aplikace. Zde se definuje jméno aplikace a region, ve kterém má být nasazena. Po založení nabízí Heroku dvě možnosti nahrání obsahu aplikace. První z nich je propojení s GitHub repozitářem, na kterém již je aplikace nahraná. Dále nabízí možnost nahrát aplikaci pomocí Heroku CLI. Tato verze byla zvolena. Jedná se o sérii tří kroků v příkazovém řádku vedoucích k nasazení aplikace. První z nich je přihlášení, následuje inicializace Git repozitáře a jeho propojení s Heroku. Posledním krokem je přidání požadovaných souborů do Gitu, vytvoření commitu a odeslání. Doména aplikace je na této platformě složená z názvu aplikace s příponou *.herokuapp.com. Informace o provozu domény lze najít na stránkách Heroku, případně pomocí CLI příkazu *Heroku apps:info –s*.

Aplikace je dostupná na adrese: https://nerit.herokuapp.com . Tato adresa byla automaticky vygenerována platformou Heroku. Pro demonstraci modelů NER poskytovaných knihovnou spaCy je tato doména dostačující. V produkčním nasazení by byla využita kratší a výstižnější doména.

5.2.5 Zhodnocení používání knihovny spaCy

Používání knihovny spaCy při tvorbě aplikace pro detekci pojmenovaných entit bylo z pohledu autora velmi intuitivní. Stránky této knihovny poskytují přehledně uspořádané velké množství podpůrných materiálů. Díky rozšíření knihovny byla přínosem i poměrně velká komunita uživatelů pro případné technické dotazy. Z hlediska práce se zdroji knihovny bylo využívání příkazové řádky rychlé a efektivní.

Během práce s knihovnou byly v oblasti NER nalezeny jisté nedostatky spaCy. První z nich je problematika popisků kategorií v jednotlivých předtrénovaných modelech. Část modelů používá pro stejné kategorie jednotné

označení, ale je zde nemalá část modelů, které používají označení naprosto odlišné. Tento fakt značně snižuje uživatelskou přívětivost knihovny, jelikož je pro definování jedné kategorie entit nutné uvést výčet všech možných označení používaných v před-trénovaných modulech. Na webových stránkách spaCy není uveden důvod tohoto přístupu, lze však předpokládat, že se jedná o problematiku spojenou s označením používaným v datech určeným k trénování.

Dále je nutné podotknout, že neplatí přímá úměra mezi rozšířením jazyka ve světě a samotným rozsahem a kvalitou modelu pro daný jazyk. Příkladem může být rumunština, která nabízí 16 kategorií pojmenovaných entit, a francouzština, která nabízí pouze čtyři možnosti.

Jak už bylo uvedeno dříve v této práci, spaCy poskytuje několik velikostí před-trénovaných modelů. Zde vyvstává problém s velikostí modelů, které jsou u některých jazyků i v nejmenší verzi značně veliké z hlediska nasazení na server. Problematické jsou především jazyky, které nepoužívají latinku – čínština, japonština, ruština, atd. U těchto jazyků je kromě samotného modelu nutné nainstalovat jazykovou sadu, která objem dat dále zvyšuje. Například čínština se u nejmenšího modelu pohybuje okolo 46MB. Při využití nejpřesnějších modelu, které spaCy nabízí, se pohybuje velikost modelu v řádech nižších stovek MB. Pro webové nasazení může být také problematické využití modelů založených na architektuře Transformer, které jsou napsané pro práci na grafických kartách.

5.3 Trénování detekce nové entity

Třetí a poslední oblastí praktické části této bakalářské práce je využití knihovny spaCy pro trénování detekce nové entity. Tato knihovna nabízí široké spektrum modelů, které s ní lze trénovat. Tato práce se však zaměří pouze na oblast NER. V těchto kapitolách bude představen celý proces od přípravy dat přes definování konfigurační složky samotného trénování, až po funkční model. Dále zde budou představeny výsledky, kterých vytvořený model dosahuje. V samotném závěru této části práce bude zhodnoceno používání spaCy pro účely trénování nové kategorie NER.

Kromě samotného představení možností poskytovaných spaCy v oblasti trénování je cílem této části vytvořit funkční model NER. Po analýze již vytvořených

modelů a podporovaných jazyků, které zatím nebyly natrénovány, byl vybrán český jazyk. Pro tento jazyk nenabízí spaCy již hotový model, ale poskytuje podporu pro jeho natrénování. Dále bylo nutné vybrat kategorii entit, na kterou bude daný model zaměřen. Vybrána byla kategorie Person (osoba). Více kategorií nebylo vybráno z důvodu výrazně vyšší časové náročnosti přípravy dat pro větší množství kategorií entit.

SpaCy nabízí podporu pro trénování v 72 světových jazycích. Dále na svých stránkách nabízí přehledného průvodce celou problematikou trénování NER. Pro jazyky, pro které již disponuje před-trénovanými modely, dává k dispozici vektory z těchto modelů. Dále poskytuje možnost před-trénování dat určených k trénování nových modelů. Je zde také prostor pro upravení přednastaveným parametrů celého procesu. Samotné modely lze trénovat jak na procesoru, tak na grafické kartě. Pro velké množství dat je zde možnost paralelního trénování. V samotném procesu trénování detekce nové entity nebyly využity všechny možnosti poskytované knihovnou spaCy. Důvodem byly především vysoké nároky knihovny na výkonnost počítače, na kterém probíhá trénování.

SpaCy pro své modely využívá neuronové sítě, konkrétně konvoluční neuronové sítě a architekturu Transformer. Pro trénování jsou k dispozici obě tyto varianty. V této oblasti se liší především v požadavcích na externí software, výkonnost hardwaru a také v časové náročnosti celého procesu. Data sloužící k trénování lze použít pro obě možnosti stejná.

5.3.1 Příprava dat pro trénování

Příprava dat, na kterých se bude nový model trénovat, je podstatnou částí celého vytváření modelu NER. Od kvality zpracování této části se bude odvíjet kvalita samotného modelu, proto je tato oblast klíčová. Samotný proces přípravy dat má několik fází. První z nich je sběr dat. Tato data by měla být ze stejné domény, ve které bude nasazen výsledný model. V případě nasazení modelu v jiné oblasti než, na kterou byl trénován, jeho funkčnost výrazně klesá. Další fází je anotace sebraných dat. Existuje řada různých softwarů, které tento proces usnadňují. Dva z nich budou představeny dále v této kapitole. Zde je podstatné pečlivé vypracování, jelikož každá chybná anotace bude snižovat kvalitu výsledků trénovaného modelu. Poslední fází,

která přibila u knihovny spaCy od verze 3.0 je přeformátování anotovaných dat. V této verzi došlo ke změně podporovaných formátů dat. Dříve podporovaný formát JSON již déle nelze využívat. Místo něho byl představen nový formát *.spacy.

V oblasti NER existují pro většinu světových jazyků již vytvořené datasety obsahující anotovaná data pro trénování neuronových sítí. Pro český jazyk se jedná například o CNEC 2.0 (Czech Named Entity Corpus). V tomto korpusu je anotováno přes 32 tisíc pojmenovaných entit. Jedná se o druhou rozšířenou verzi CNEC 1.0 od Ševčíková a kol. (2007). V rámci představení celého procesu trénování detekce pojmenovaných entit bude zvolen přístup vytvoření vlastních anotovaných dat od úplného začátku bez využití dostupných korpusů.

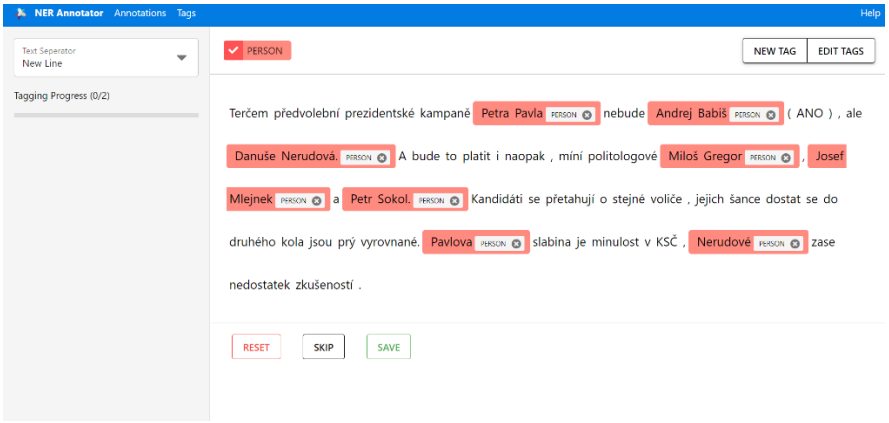
Množství dat potřebných k natrénování nového modelu se liší podle vybrané oblasti, množství kategorií, vybrané metodě trénování a také požadavcích na úspěšnost vyhledávání pojmenovaných entit. Pro model s nižší mírou přesnosti lze pracovat s daty, ve kterých jsou řádově stovky označených entit. V případě, že je na výsledcích modelu závislý další proces a je požadovaná vysoká přesnost, jsou potřebná data s řádově tisící až desetitisíci označených entit. SpaCy požaduje pro proces tvorby nového modelu dvě sady dat. První je sada dat určená k trénování a druhá je sada určená k vyhodnocení úspěšnosti současného modelu. Tyto dvě sady by neměly obsahovat totožná data. V této bakalářské práci byl vytvořen soubor dat pro trénování o velikosti 1408 označených entit a také soubor pro vyhodnocení výsledků modelu čítající 81 entit. Jedná se o výrazně nižší čísla než u CNEC 2.0, ale pro představení možností knihovny spaCy je toto množství dostačující.

Jako cílová oblast, ve které bude model využíván, byly zvoleny internetové články. Jedná se o oblast, ve které se hojně vyskytují osoby, které lze detekovat. Proto byla data čerpána z obdobných zdrojů. Využita byla především česká internetová zpravodajství. Přesněji webové stránky spadající pod společnost Seznam.cz a to: Novinky.cz a Seznam Zprávy. Jednotlivé články jsou k práci přiloženy ve formátu *mhtml.

Pro anotaci nabízí výrobce spaCy – firma Explosion vlastní software nazvaný Prodigy. Jedná se o webovou aplikaci, která poskytuje nástroje pro anotaci dat určených k trénování. V praktické části bakalářské práce však nebude využito této

aplikace, jelikož se na rozdíl od spaCy jedná o placený software. Cena za jednu licenci se pohybuje okolo 400 dolarů.

Samotný program sloužící k anotaci dat není příliš složitý na naprogramování, proto se například na stránkách GitHub nachází několik možností nabízející tuto funkcionalitou s různou mírou funkčnosti. V této práci bude jeden z těchto programů využit. Konkrétně byl zvolen Ner-annotator od Arunmozhi (2022). Jedná se o zdarma přístupnou webovou aplikaci, která poskytuje základní nástroje pro anotaci dat. Výstupním formátem dat s označenými entitami je JSON. Na obrázku číslo 11 je představena ukázka práce v této aplikaci.



Obrázek 11: Webová aplikace pro anotaci dat
Zdroj: autor

Jak je znázorněno na obrázku číslo 11, k označení vyhledávané kategorie pojmenovaných entit bylo použito označení Person. Tento název byl vybrán na základě analýzy pojmenování této kategorie v modelech poskytovaných knihovnou spaCy. Aby autor předešel přidání dalšího pojmenování pro stejnou kategorii, bylo využito nejčastější pojmenování v již stávajících modelech.

Jak již bylo zmíněno výše, výstupem webové aplikace použité pro anotaci dat v této práci je formát JSON. Tento formát již není dále spaCy podporován, proto bylo nutné výsledný soubor přeformátovat. Styl formátování výsledku v této aplikace je mírně odlišný od formátu, které používá Prodigy, proto nelze na přeformátování využít konvertory od spaCy. Tento problém lze vyřešit využitím částí kódu, které vytvořila komunita okolo projektu Ner-annotator.

5.3.2 Trénování detekce nové entity pomocí CNN

Po vytvoření dat určených pro trénování a hodnocení výsledků modelu je následující fází samotné trénování pomocí neuronových sítí. Jedná se o velmi složitý proces, proto spaCy nabízí přednastavené konfigurační soubory, které poskytují základní nastavení všech potřebných částí. V případech, kdy je požadováno specifické nastavení trénování, umožňuje knihovna uzpůsobit celý průběh pomocí definování položek v konfiguračním souboru. Další možností je nastavení parametrů u příkazů sloužících k zahájení celého procesu. V těchto parametrech lze nastavit například využití GPU místo CPU.

První částí samotného trénování je nastavení celého procesu. K tomuto účelu používá knihovna spaCy konfigurační soubor *config.cfg*. První možností vytvoření tohoto souboru je využití webových stránek knihovny. V záložce Usage se nachází karta Training Models a v ní se nachází komponenta umožňující grafické nastavení konfigurační složky. Po vyplnění příslušných polí je možné danou složku stáhnout jako celek nebo pouze zkopírovat její obsah. Před použitím je nejprve nutné daný soubor inicializovat. V tomto procesu dochází k vyplnění nastavení, která nebylo možné vyplnit již na webové stránce. Druhou možností je vytvoření konfiguračního souboru pomocí příkazu v příkazovém řádku. Při zvolení této možnosti nenastává krok inicializace souboru. V obou možnostech je nutné definovat jazyk, zaměření (výkonnost/přesnost) a komponenty, které mají být do procesu zařazeny.

V této bakalářské práci bylo vyzkoušeno několik různých nastavení konfiguračního souboru. U trénování s konvolučními neuronovými sítěmi byly časy trénování u modelů zaměřených na výkonnost a přesnost podobné, proto bylo zvoleno nastavení preferující přesnost. Z hlediska relativně malého množství dat k trénování dosahovaly modely s tímto zaměřením výrazně přesnějších výsledků. Množství dat poskytnutých k tréninku bylo postupně navyšováno, avšak pro modely zaměřené na výkonnost nebylo toto množství dostatečné.

Pro mnoho funkcí nabízí spaCy alternativní možnost spuštění skrze příkazovou řádku. Tento přístup výrazně zvyšuje efektivitu a rychlost práce s knihovnou. V případě trénování detekce pojmenované entit nabízí příkaz zobrazený na obrázku číslo 12. Samotné trénování lze provést i pomocí spaCy

komponent přímo v kódu. Využití příkazové řádky je však značně rychlejší přístup, proto byl v této práci zvolen.

```
python -m spacy train config.cfg --output ./  --paths.train ./training_data.spacy --paths.dev ./evaluation_data.spacy
```

Obrázek 12: Příkaz pro trénování nového modelu
Zdroj: autor

Na obrázku číslo 12 je představena obecná struktura příkazu poskytovaného knihovnou spaCy pro započetí trénování pomocí CNN. Na svých webových stránkách podrobně popisuje všechny dostupné argumenty příkazu. Zde budou stručně představeny pouze použité možnosti. Jako první se volá knihovna spaCy konkrétně její funkce train. Prvním argumentem příkazu je konfigurace celého procesu. Zde se zadává cesta k požadovanému konfiguračnímu souboru. Následuje nastavení adresáře, do kterého mají být umístěny výsledné modely. Při tomto nastavení budou v adresáři přepsány soubory pojmenované totožně jako výsledné modely. Dále je nutné definovat cestu k datům, které budou sloužit trénování. Jako poslední argument je na obrázku číslo 12 zobrazena cesta k datům sloužícím k vyhodnocení funkčnosti jednotlivých modelů. U všech zadávaných cest je nutné začít v adresáři, ve kterém bude příkaz zadáván.

```
✓ Initialized pipeline

============================= Training pipeline =====================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00     27.00    3.23    2.27    5.56    0.03
163     200         77.55    664.04    5.88    6.25    5.56    0.06
363     400          0.00      0.00    5.71    5.88    5.56    0.06
563     600          0.00      0.00    5.71    5.88    5.56    0.06
763     800          0.00      0.00    5.88    6.25    5.56    0.06
```

Obrázek 13: Příklad výsledků trénování
Zdroj: autor

Na obrázku číslo 13 je zobrazen výstup procesu trénování detekce nové pojmenované entity pomocí knihovny spaCy v příkazové řádce. Ve výstupu jsou

vypsány všechny trénované komponenty. V příkladu na obrázku 13 se jedná o tok2vec a ner. Nejdůležitější částí je tabulka zobrazující aktuální výsledky trénování. První sloupec označený E informuje o počtu provedených epoch (průchod celého datasetu skrze CNN). Následující sloupec se symbolem # odkazuje na batch. Jedná se o část epochy. V konfiguračním souboru lze nastavit maximální počet epoch/batch, které mají být provedeny. Tímto lze definovat délku procesu trénování. Dále tabulka obsahuje dva sloupce s označením začínajícím LOSS. Značí míru chybovosti jednotlivých komponent. Sloupec s označením ENTS_F ukazuje harmonický průměr hodnot následujících dvou sloupců. Prvním z nich je sloupec ENTS_P, který udává procentuální úspěšnost v detekování pojmenovaných entit. Druhým sloupcem je ENTS_R, který značí citlivost daného modelu. Tato hodnota je vypočítávána z poměru správně nalezených entit a entit označených v datech určených k trénování. Poslední sloupec je nazván SCORE. V tomto sloupci je uvedeno celkové skóre daného modelu. Je vypočítáváno dle předem daných poměrů z dat v předchozích sloupcích. Na základě této hodnoty je v průběhu procesu trénování vyhodnocován nejlepší model, který je na závěr uložen spolu s modelem, který byl vytvořen jako poslední. Celkovým výstupem trénování CNN s využitím knihovny spaCy jsou dva modely – nejlepší a poslední.

5.3.3 Trénování detekce nové entity pomocí Transformer

Kromě trénování nových modelů pomocí CNN nabízí knihovna spaCy od verze 3.0 také architekturu Transformer. Ta poskytuje lepší výsledky, ale její náročnost na trénování je výrazně vyšší. Celý proces vytvoření modelu pro detekci nové entity je velmi podobný jako při využití CNN. Jednotlivé rozdíly budou představeny dále v této kapitole.

SpaCy nenabízí vlastní verzi Transformer neuronových sítí. Jedná se o spolupráci se společností HuggingFace, která poskytuje vlastní knihovnu pro práci s Transformer nazvanou Transfomers. Ta samotná využívá pro své fungování PyTorch, TensorFlow a JAX. Nabízí širokou škálu využití například v NLP, počítačovém vidění či zpracování zvuku. V této bakalářské práci bude tato knihovna použita pouze pro trénování nového modelu NER. SpaCy zprostředkovává využívání této knihovny skrze vlastní rozhraní.

Samotná architektura Transformer byla primárně napsána pro práci na grafické kartě. SpaCy umožňuje trénovat tuto architekturu i na procesoru. Tato volba je základním nastavením konfiguračních souborů. Při využití této možnosti však nastává problém s efektivitou samotného trénování. Délka celého procesu se oproti využití GPU násobně prodlužuje. Naopak nároky na hardware jsou u využití CPU nižší. Proto je využití této možnosti vhodné pouze v případech, kdy není k dispozici dostatečně výkonná grafická karta.

Pro spuštění procesu trénování za využití Transformers je nejprve nutné provést instalaci požadovaného softwaru. První fází je aktualizace ovladačů grafické karty. Dále je nutné provést instalaci CUDA, který poskytne podporu při využití GPU pro účely trénování. Následně je potřebné nainstalovat framework PyTorch, který je využíván knihovnou Transfomers. V tomto kroku je důležité využít kompatibilní verze CUDA a PyTorch. Posledním krokem je instalace doplňků pro knihovnu spaCy.

Konkrétně se jedná o doplňky transformers a cuda, u kterého se za název doplní aktuální verze nainstalovaného CUDA. Objem dat všech těchto souborů dohromady by měl dosahovat řádově nižších jednotek GB.

Jak již bylo výše zmíněno, architektura Transformer byla vytvořena pro práci na grafické kartě. Požadavky na hardware pro provoz této architektury jsou poměrně vysoké. Na svých webových stránkách uvádí spaCy (2022) následující: *„Pro práci s modely transformer doporučujeme GPU NVIDIA s alespoň 10 GB paměti.“* Náročnost celého procesu lze do jisté míry modifikovat v nastavení konfiguračního souboru. Například snížením hodnoty atributů batch_size nebo max_batch_items. V této bakalářské práci byla využita grafická karta MSI GeForce GTX 1050 Ti OC s pamětí o velikosti 4 GB. Bylo vyzkoušeno několik optimalizací dat, výkonu grafické karty a konfiguračního souboru, ale ani po těchto úpravách nebyla tato karta dostatečně silná pro trénování s Transformer. Problematickým bodem byla především kapacita paměti grafické karty. Při spuštění trénování na GPU docházelo po proběhnutí několika epoch k vyčerpání paměťového místa. Proto byl na trénování s touto architekturou využit procesor AMD Ryzen 5 1600 spolu s 16 GB 2400MHz operační paměti. Tato kombinace hardwaru již byla dostačující, ačkoliv byl samotný proces časově velmi náročný. Celková doba zpracování při použití dat k trénování o velikosti 1408 označených entit a dat k vyhodnocování o velikosti 81

označených entit se pohybovala okolo 26 hodin. Zpracování 200 batchů trvalo v průměru dvě hodiny. Z časového hlediska byl rozdíl mezi nastavením preferujícím přesnost a výkonnost zanedbatelný.

I u této architektury spaCy nabízí velké množství nastavení konfiguračního souboru. Hlavní nastavení jsou však totožná s těmi u CNN. Jedná se tedy o: jazyk modelu, seznam požadovaných komponent a zaměření na výkonnost nebo přesnost. Oproti CNN zde přibývá možnost nastavovat CUDA, PyTorch a v konfiguračním souboru spaCy se také nachází více položek.

Pro zahájení samotného procesu trénování lze u Transformer využít stejný příkaz jako u CNN. Při použití tohoto příkazu bude proces využívat CPU. Jedná se o základní nastavení, které je nutné změnit. Pro přepnutí na grafickou kartu je nutné do příkazu přidat argument *--gpu-id 0*. Informační výstup v průběhu trénování je totožný s CNN.

5.3.4 Porovnání výsledků s CNN a Transformer

Posledním výstupem této bakalářské práce byly dva vytvořené NER modely. Každý model byl trénován pomocí jiné architektury neuronových sítí. Pro lepší představení těchto modelů bylo provedeno testování jejich výkonnosti a přesnosti na několika internetových článcích.

Oba modely byly vytvářeny pro český jazyk pomocí knihovny spaCy. Byly zaměřeny na detekci pojmenovaných entit patřících do kategorie osoba. Dále jim byla dána totožná data k trénování. Konfigurační soubory obou modelů byly nastaveny na prioritu přesnosti. V obou případech bylo trénování prováděno na procesoru. Rozdílná je výsledná velikost souborů. U modelu vytvářeného s CNN se jedná o 34,5 MB. U modelu z architektury Transformer je celková velikost 642 MB. Dále se liší v časové náročnosti procesu trénování. U CNN se celková doba trénování pohybovala okolo 2 hodin, u Transformer pak okolo 26.

Tabulka 1: Srovnání CNN a Transformer

článek		CNN model		Transformer model	
číslo článku	počet entit	správné detekce	chybné detekce	správné detekce	chybné detekce
1	**13**	13	1	13	0
2	**34**	25	2	34	2
3	**18**	15	3	18	0
4	**11**	10	4	11	0
5	**22**	13	0	21	1
6	**36**	25	2	30	2

Zdroj: autor

V předchozím odstavci byly shrnuty shody a rozdíly obou modelů a s těmito informacemi bude vyhodnocena tabulka číslo 1, která zobrazuje výsledky, kterých dosáhly. Tato tabulka představuje shrnutí testování obou modelů na 6 různých internetových článcích. U každého článku je zapsán reálný počet entit, které se v něm nacházejí. Dále pak výsledky, kterých modely dosáhly. U každého z nich jsou kategorie správné a chybné detekce. Představují základní parametry kvality daného výstupu. U položek správná detekce byly tolerovány entity obsahující čárku, tečku nebo pomlčku. Chybná detekce ve většině výsledků obsahovala některé slova navíc. V několika případech došlo k označení entity, které ani ze své části nespadala do vyhledávané kategorie.

I přes relativně malý testovaný vzorek lze z tabulky číslo 1 vyhodnotit, že model založený na Transformer dosahuje konzistentně lepší výsledků. Ve všech případech, kromě prvního, nalezl více entit než model CNN. Dále má také výrazně nižší chybovost. V polovině případů neudělal žádnou chybu a ve zbylých naprosté minimum. Naproti těmto výsledkům stojí časově výrazně náročnější trénování modelu. Dále také násobně větší velikost i požadování GPU pro fungování. Na základě výše zmíněných informací se model založený na Transformer hodí spíše pro lokální nasazení než pro webovou aplikaci. Naproti tomu model CNN svojí velikostí i zpracováním na CPU je vhodnou variantou pro nasazení na server webové aplikace.

5.3.5 Zhodnocení používání knihovny spaCy

Proces trénování modelu detekce pojmenované entity lze považovat za náročný úkon. Knihovna spaCy však poskytuje nadstavbu, které dává možnost uživateli celý proces výrazně zjednodušit. Jedná se především o přednastavené

konfigurační soubory. Dále také na svých webových stránkách přehledně a systematicky provádí uživatele celou problematikou.

Problematickou částí procesu trénování je u spaCy příprava dat. Jak již bylo výše zmíněno, knihovna poskytuje vlastní software Prodigy, který nabízí dostatečnou podporu v oblasti označování entit v textu. Svojí cenou však cílí především na firemní využití než na jednotlivce. Ti jsou v tomto případě odkázání na označování entit ručně (při větším množství dat nepoužitelná metoda), případně pomocí programů vytvořených komunitou spaCy, které bývají často naprogramovány značně nekvalitně. Tímto se celá příprava dat stává velmi zdlouhavou a neefektivní.

Od verze 3.0 přestala spaCy využívat formát JSON a přešla na vlastní formát *.spacy. Důvodem tohoto kroku byla podle autorů optimalizace a možnost používat na vstupu totožný formát jako na výstupu. Přes všechny výhody, které díky této změně nastaly, zde přibyl další krok v procesu přípravy dat. Všechna data dříve vytvořená ve formátu JSON je nutné před použitím přeformátovat do současného formátu.

Pro trénování s architekturou Transformer jsou nároky na grafickou kartu tak vysoké, že se její využívání stává pro běžného uživatele nemožné. A to ani při relativně malém množství dat, jak bylo ukázáno v této bakalářské práci. Pro nasazení v reálném provozu je potřebné trénovat na násobně větším objemu dat. V tomto případě se jedná o proces vyžadující nadstandardní hardwarové vybavení.

Během trénování pomocí spaCy byla nalezena chyba na webových stránkách této knihovny. Při definování konfiguračního souboru online docházelo k opakovanému zadání jiného jazyka, než který byl vybrán. Při bližším zkoumání byl nalezen pravděpodobný zdroj - posun indexu v elementu select, který způsoboval zadání následujícího jazyka místo jazyka vybraného. Důsledkem byla opakovaná nefunkčnost procesu trénování. Tato chyba byla v průběhu zpracovávání bakalářské práce odstraněna.

Přes výše zmíněné nedostatky poskytuje spaCy kvalitní a stabilní prostředí, které umožňuje uživateli bez předchozích zkušeností vytvářet nejenom NER modely. Výhodou knihovny je také poměrně široká komunita, která poskytuje podporu při počátečních i složitějších úkonech.

6 Shrnutí výsledků

Cílem této bakalářské práce bylo představit možnosti využití knihovny spaCy v oblasti detekce pojmenovaných entit. V teoretické části byly představeny informace shrnující tuto problematiku. Jedná se především o oblasti zpracování a porozumění přirozenému jazyku, detekce pojmenovaných entit a představení třech aktuálních přístupů k trénování nových modelů NER. Tyto znalosti byly potřebné pro kvalitní vypracování praktické části bakalářské práce.

První oddíl praktické části byl věnován analýze knihovny spaCy. Webové stránky spaCy prošly v roce 2021 výraznou obměnou a výsledkem jejich analýzy bylo vyzdvihnutí jejich přehlednosti a uživatelské přívětivosti. Dále tento oddíl analyzoval jednotlivé modely poskytované knihovnou. SpaCy jich poskytuje širokou škálu s různou kvalitou zpracování. Pro jazyky, které dosud nemají vlastní model, nabízí podporu pro jeho vytvoření. Poslední část tohoto oddílu analyzovala kód pro využití knihovny. Kód byl zhodnocen jako přehledný a úsporný.

Následující oddíl byl věnován vývoji extraktoru pojmenovaných entit. Zde byl primární cíl demonstrovat možnosti v oblasti NER poskytované spaCy. Dále zde byla zhodnocena práce s knihovnou a to jako velmi intuitivní. Představeny byly i nedostatky především modelů spaCy, které byly zjištěny během procesu vývoje extraktoru. Jedná se o rozličné pojmenování totožných kategorií u jednotlivých modelů. Dále nepoměr mezi rozšířením daného jazyka a kvalitou jeho modelu. V oblasti nasazení se jako problematický ukázal datový objem modelů a následně jejich vysoká spotřeba operační paměti.

Poslední oddíl praktické části byl věnován trénování dvou nových modelů NER pomocí knihovny spaCy. Oba modely byly vytvořeny pro český jazyk a pro detekci osob se zaměřením na internetové články. Použity byly architektury CNN a Transformer. Modely byly otestovány na 6 článcích a výsledky následně porovnány. Architektura Transformer dosáhla výrazně lepších výsledků (více nalezených entit a méně chybně identifikovaných entit) ve všech testovaných případech. Z důvodu velikého datového objemu a požadavku fungování na GPU byl tento model vyhodnocen jako nevhodný pro nasazení na server. Dále byla manuálně vytvořena

sada anotovaných dat ve formátu *.spacy čítající 1408 entit. A dále sada anotovaných dat pro vyhodnocení výsledků o velikosti 81 entit.

Hypotéza číslo 1: *„Knihovna spaCy poskytuje dostatečný rozsah prostředků pro tvorbu webové aplikace na detekci pojmenovaných entit.“* Byla potvrzena druhým oddílem praktické části této práce, kde byla vytvořena webová aplikace za využití knihovny spaCy poskytující služby detekce pojmenovaných entit.

Hypotéza číslo 2: *„Knihovna spaCy umožňuje značnou míru odstínění od složitosti NLP, nikoli však úplné odstínění.“* Byla potvrzena analýzou modelů a kódu knihovny spaCy. Kód knihovny byl shledán úsporným a přehledným. Jednotlivé modely byly i přes rozdílné pojmenování totožných kategorií entit srozumitelné a přehledné. V průběhu vývoje webového extraktoru nebyl shledán žádný výrazný požadavek na znalosti z oblasti NLP. Pouze při vývoji nových modelů byla k jejich trénování potřebná základní znalost NER.

Hypotéza číslo 3: *„Proces natrénování detekce nové entity v knihovně spaCy je velmi náročný, nikoli však nemožný pro uživatele se znalostmi oblasti informačních technologií na úrovni bakalářského studia.“* Tato hypotéza byla v případě autora potvrzena. Pro její plnohodnotné potvrzení nebo vyvrácení by však bylo vhodné provést testování na větším vzorku studentů informačních technologií.

7 Závěry a doporučení

Hlavním výstupem této bakalářské práce bylo demonstrovat možnosti poskytované knihovnou spaCy v oblasti detekce pojmenovaných entit. Tohoto bylo dosaženo vytvořením webového extraktoru pro 24 světových jazyků. Dále byla provedena analýza webových stránek, před-trénovaných modelů a kódu knihovny. Zde byly nalezeny určité nedostatky – například různá pojmenování totožné kategorie entit u jednotlivých modelů. Představení potenciálu knihovny spaCy bylo podpořeno i vytvořením dvou nových modelů NER pro český jazyk.

Samotné výsledky práce byly ovlivněny mnoha faktory. Například při nasazování backend části hotového extraktoru na server nastala komplikace v podobě vysoké náročnosti na kapacitu operační paměti. Toto bylo způsobeno nasazení 24 modelů NER na jeden server. Problém byl vyřešen rozdělením backend části aplikace na několik serverů.

V průběhu vývoje nových modelů NER bylo problematické využití architektury Transformer. Tato architektura je vytvořená pro práci na GPU. Avšak hardwarové požadavky násobně převyšovaly výkon zařízeních využitých při vývoji. Zde bylo využito možnosti pracovat s Transfomer na CPU i za cenu výrazně delšího trvání procesu trénování.

Samotné představení knihovny spaCy mohlo být provedeno také ve formátu desktopové či mobilní aplikace. Obě tyto možnosti mají své výhody i nevýhody. Z hlediska současného trendu v oblasti informatiky se však formát ve stylu webové aplikace zdá nejvhodnější. Sekce nastiňující potenciál spaCy pro trénování modelů NER mohla být demonstrována na rozšíření stávajících modelů o další kategorie místo tvorby celých nových modelů.

Mezi otevřené problémy pro další studium lze řadit například sjednocení názvů kategorií u jednotlivých před-trénovaných modelů. Tato problematika značně snižuje uživatelskou přívětivost knihovny. Jedním z možných přístupů k řešení je vytvoření nadstavby nad modely spaCy, která by sjednotila jednotlivé názvy pod jedno univerzální označení kategorie.

Dále je zde prostor pro rozšíření a zkvalitnění modelů NER vytvořených v této bakalářské práci. Pro produkční nasazení by bylo vhodné provést několik úprav. Lze

rozšířit škálu nabízených kategorií pojmenovaných entit. Dále zkvalitnit práci modelů pomocí rozšíření dat k trénování a optimalizací konfigurace samotného procesu trénování.

V průběhu anotování dat pro vytvářené modely byl nalezen nedostatek v množství kvalitních bezplatných programů pro tuto činnost. Zde se naskytuje problém pro další studium - vyvinutí bezplatného kvalitního softwaru pro anotaci dat. U tohoto softwaru by bylo vhodné implementovat určitou formu strojové anotace, která by uživateli s celým procesem pomáhala.

8 Seznam použité literatury

1. MAYNARD, Diana, Kalina BONTCHEVA a Isabelle AUGENSTEIN. Natural Language processing for the Semantic Web. London: Morgan & Claypool Publishers, 2017. Synthesis Lectures on the semantic web: theory and technology, 15. ISBN 978-1-62705-632-8.

2. XIAO, Yu a Zhezhi JIN. Summary of Research Methods on Pre-Training Models of Natural Language Processing. OALib. 2021, **8**(7), 1–7. ISSN 2333-9721. Dostupné z: doi:10.4236/oalib.1107602

3. ROSHANZAMIR, Alireza, Hamid AGHAJAN a Mahdieh SOLEYMANI BAGHSHAH. Transformer-based deep neural network language models for Alzheimer's disease risk assessment from targeted speech. BMC Medical Informatics and Decision Making. 2021, **21**(1), 92. ISSN 1472-6947. Dostupné z: doi:10.1186/s12911-021-01456-3

4. GOYAL, Archana et al. Named Entity Recognition: Applications, Approaches and Challenges. International Journal of Advance Research in Science and Engineering. 2017, **6**(10). ISSN 2319-8354.

5. LIU, Shuang, Hui YANG, Jiayi LI a Simon KOLMANIČ. Chinese Named Entity Recognition Method in History and Culture Field Based on BERT. International Journal of Computational Intelligence Systems. 2021, **14**(1), 163. ISSN 1875-6883. Dostupné z: doi:10.1007/s44196-021-00019-8

6. PAIS, Sebastião, João CORDEIRO a M. Luqman JAMIL. NLP-based platform as a service: a brief review. Journal of Big Data. 2022, **9**(1), 54. ISSN 2196-1115. Dostupné z: doi:10.1186/s40537-022-00603-5

7. SpaCy [online]. Dostupné z: https://spacy.io/

8. CHENG, Xinyun, Xianglong KONG, Li LIAO a Bixin LI. A Combined Method for Usage of NLP Libraries Towards Analyzing Software Documents. In: Schahram DUSTDAR, Eric YU, Camille SALINESI, Dominique RIEU a Vik PANT, ed. Advanced Information Systems Engineering. Cham: Springer International Publishing, 2020, Lecture Notes in Computer Science, s. 515–529. ISBN 978-3-030-49434-6. Dostupné z: doi:10.1007/978-3-030-49435-3_32

9. KUMAR, Ela. Natural Language Processing. Nové Dillí: L. K. International Publishing House, 2011. ISBN 978-93-80578-77-4.

10. COLMAN, Andrew M. A Dictionary of Psychology. Oxford University Press, 2008. ISBN 9780199534067.

11. CHOWDHURY, Gobinda G. Natural language processing. Annual Review of Information Science and Technology, 2005, **37**(1), 51–89. ISSN 00664200. Dostupné z: doi:10.1002/aris.1440370103

12. NADKARNI, Praskash M., Lucila OHNO-MACHADO a Wendy W. CHAPMAN. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*. 2011, **18**(5), 544-551. ISSN 1067-5027. Dostupné z: doi:https://doi.org/ 10.1136/amiajnl-2011-000464

13. CUNNINGHAM, Hamish. Information extraction, automatic. *Encyclopedia of Language & Linguistics*. 2. vyd. Oxford: Elsevier Science, 2006, s. 665-677. ISBN 9780080547848.

14. CHIMAOBIYA, Okoro Jennifer a Hari PRIYA. Sentiment Analysis and Opinion Mining. International Journal of Engineering Research & Technology. 2016, **4**(27). ISSN 2278-0181.

15. POPEL, Martin, Marketa TOMKOVA, Jakub TOMEK, Łukasz KAISER, Jakob USZKOREIT, Ondřej BOJAR a Zdeněk ŽABOKRTSKÝ. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. Nature Communications. 2020, **11**(1), 4381. ISSN 2041-1723. Dostupné z: doi:10.1038/s41467-020-18073-9

16. ALLEN, James F. Natural language processing. RALSTON, Anthony, Edwin D. REILLY a David HEMMENDINGER. *Encyclopedia of Computer Science*. 4. vyd. Baffins Lane Chichester West Sussex: John Wiley and Sons, 2003, s. 1218-1222. ISBN 978-0-470-86412-8.

17. ENGEL, Ralf. Natural Language Understanding. In: Wolfgang WAHLSTER, ed. SmartKom: Foundations of Multimodal Dialogue Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. Cognitive Technologies, s. 195–207. ISBN 978-3-540-23732-7. Dostupné z: doi:10.1007/3-540-36678-4_13

18. SCHMITT, Tobias, Cedric KULBACH a York SURE-VETTER. Improving NLU Training over Linked Data with Placeholder Concepts. In: Maribel ACOSTA, Philippe CUDRÉ-MAUROUX, Maria MALESHKOVA, Tassilo PELLEGRINI, Harald SACK a York SURE-VETTER, ed. Semantic Systems. The Power of AI and Knowledge Graphs. Cham: Springer International Publishing, 2019, Lecture Notes in Computer Science, s. 67–82. ISBN 978-3-030-33219-8. Dostupné z: doi:10.1007/978-3-030-33220-4_6

19. ALTINOK, Duygu. Mastering spaCy. Packt Publishing, 2021. ISBN 9781800563353.

20. EFTIMOV, Tome, Barbara KOROUŠIĆ SELJAK a Peter KOROŠEC. A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. PLOS ONE. 2017, **12**(6), e0179488. ISSN 1932-6203. Dostupné z: doi:10.1371/journal.pone.0179488

21. SONG, Hye-Jeong, Byeong-Cheol JO, Chan-Young PARK, Jong-Dae KIM a Yu-Seop KIM. Comparison of named entity recognition methodologies in biomedical documents. BioMedical Engineering OnLine. 2018, **17**(2), 158. ISSN 1475-925X. Dostupné z: doi:10.1186/s12938-018-0573-6

22. EKBAL, Asif et al. Named Entity Recognition using Support Vector Machine: A Language Independent Approach. International Journal of Electrical, Computer, and Systems Engineering. 2010, (39), 155-169.

23. MORWAL, Sudha et al. Named Entity Recognition using Hidden Markov Model (HMM). International Journal on Natural Language Computing. 2012, **1**(4), 15–23. ISSN 23194111. Dostupné z: doi:10.5121/ijnlc.2012.1402

24. CHIEU, Hai Leong a Hwee Tou NG. Named entity recognition: a maximum entropy approach using global information. In: the 19th international conference: Proceedings of the 19th international conference on Computational linguistics. Taipei, Taiwan: Association for Computational Linguistics, 2002, s. 1–7. Dostupné z: doi:10.3115/1072228.1072253

25. JIN, Yanliang, Jinfei XIE, Weisi GUO, Can LUO, Dijia WU a Rui WANG. LSTM-CRF Neural Network With Gated Self Attention for Chinese NER. IEEE Access. 2019, 7, 136694–136703. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2019.2942433

26. LI, Jing, Aixin SUN, Jianglei HAN a Chenliang LI. A Survey on Deep Learning for Named Entity Recognition. IEEE Transactions on Knowledge and Data Engineering. 2022, **34**(1), 50–70. ISSN 1041-4347, 1558-2191, 2326-3865. Dostupné z: doi:10.1109/TKDE.2020.2981314

27. PETASIS, Georgios, Alessandro CUCCHIARELLI, Paola VELARDI, Georgios PALIOURAS, Vangelis KARKALETSIS a Constantine D. SPYROPOULOS. Automatic adaptation of proper noun dictionaries through cooperation of machine learning

and probabilistic methods. In: the 23rd annual international ACM SIGIR conference: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '00. Athens, Greece: ACM Press, 2000, s. 128–135. ISBN 978-1-58113-226-7. Dostupné z: doi:10.1145/345508.345563

28. NADEAU, David a Satoshi SEKINE. A survey of named entity recognition and classification. Lingvisticae Investigationes. 2007, **30**(1), 3–26. ISSN 0378-4169, 1569-9927. Dostupné z: doi:10.1075/li.30.1.03nad

29. BAHDANAU, Dzmitry, Kyunghyun CHO a Yoshua BENGIO. Neural Machine Translation by Jointly Learning to Align and Translate. 2014. Dostupné z: doi:10.48550/ARXIV.1409.0473

30. VASWANI, Ashish, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Lukasz KAISER a Illia POLOSUKHIN. Attention Is All You Need. arXiv. 2017. Dostupné z: http://arxiv.org/abs/1706.03762. arXiv:1706.03762

31. DEVLIN, Jacob, Ming-Wei CHANG, Kenton LEE a Kristina TOUTANOVA. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018. Dostupné z: doi:10.48550/ARXIV.1810.04805

32. KREŠŇÁKOVÁ, Viera Maslej, Martin SARNOVSKY a Peter BUTKA. Deep learning methods for Fake News detection. In: 2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo): 2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo). Szeged, Hungary: IEEE, 2019, s. 000143–000148. ISBN 978-1-72815-625-5. Dostupné z: doi:10.1109/CINTI-MACRo49179.2019.9105317

33. YAMASHITA, Rikiya, Mizuho NISHIO, Richard Kinh Gian DO a Kaori TOGASHI. Convolutional neural networks: an overview and application in radiology. Insights into Imaging. 2018, **9**(4), 611–629. ISSN 1869-4101. Dostupné z: doi:10.1007/s13244-018-0639-9

34. ALBAWI, Saad, Oguz BAYAT, Saad AL-AZAWI a Osman N. UCAN. Social Touch Gesture Recognition Using Convolutional Neural Network. Computational

Intelligence and Neuroscience. 2018, 1–10. ISSN 1687-5265, 1687-5273. Dostupné z: doi:10.1155/2018/6973103

35. Explosion AI: Software [online]. [cit. 2022-11-02]. Dostupné z: https://explosion.ai/software

36. ŠEVČÍKOVÁ, Magda a kol. Named Entities in Czech: Annotating Data and Developing NE Tagger. Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue. Heidelberg: Springer, 2007, 4629(17), 188-195. ISSN 0302-9743.

37. Arunmozhi, ner-annotator, (2022), GitHub repository, https://github.com/tecoholic/ner-annotator

9 Seznam obrázků

10 Seznam tabulek

11 Přílohy

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2021/2022

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení:	**Jakub Včelák**
Osobní číslo:	**I2000434**
Adresa:	Vavřinecká 312, Opatovice nad Labem, 53345 Opatovice nad Labem, Česká republika
Téma práce:	Detekce pojmenovaných entit pomocí knihovny spaCy
Téma práce anglicky:	Named Entities Recognition using spaCy library
Jazyk práce:	Čeština
Vedoucí práce:	Ing. Martina Husáková, Ph.D. Katedra informačních technologií

Zásady pro vypracování:

Cíl: průzkum možností extrakce pojmenovaných entit z textů, které nabízí knihovna spaCy, spolu s vývojem vlastního extraktoru ve zmíněné knihovně
Obsah:

1. Úvod
2. Literární rešerše
3. Cíl práce, volba metodologie, způsob řešení
4. Teoretická východiska
4.1. Přirozený a umělý jazyk
4.2. Zpracování přirozeného jazyka
4.3. Porozumění přirozenému jazyku
4.4. Detekce pojmenovaných entit
4.5. Vývoj oblasti detekce pojmenovaných entit
4.6. Trénování detekce pojmenovaných entit
5. Praktická část
6. Shrnutí výsledků
7. Závěry a doporučení

Seznam doporučené literatury:

Zdroje:

- spaCy homepage: https://spacy.io/
- Vasiliev, Y. Natural Language Processing with Python and spaCy: A Practical Introduction. No Starch Press, 2020.
- Srinivasa-Desikan, B. Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras. Packt Publishing, 2018.
- Antic, Z. Python Natural Language Processing Cookbook: Over 50 recipes to understand, analyze, and generate text for implementing language processing tasks. Packt Publishing, 2021.
- Maynard, D., et al. Natural Language Processing for the Semantic Web. 2016. DOI:10.2200/s00741ed1v01y201611wbe015
- Mattingly, W. J. B. Introduction to Named Entity Recognition: With a Case Study of Holocaust NER.2021 (2nd ed.). URL: https://ner.pythonhumanities.com/intro.html#

Podpis studenta: Dušek Datum: 14. 3. 2023

Podpis vedoucího práce: Kusáková Datum: 14.3.2023