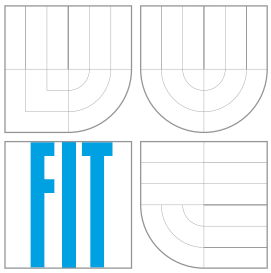
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KONTROLA KONZISTENCE INFORMACÍ EXTRAHOVANÝCH Z TEXTU

CONSISTENCY CHECKING OF RELATIONS EXTRACTED FROM TEXT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB STEJSKAL

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2016

Zadání bakalářské práce/18808/2015/xstejs24

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Stejskal Jakub**
Obor: Informační technologie
Téma: **Kontrola konzistence informací extrahovaných z textu**
Consistency Checking of Relations Extracted from Text
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s metodami extrakce informací z textu a kontroly konzistence extrahovaných dat na základě definovaných omezení.
2. Navrhněte a implementujte systém pro kontrolu konzistence informací extrahovaných z Wikipedie a dalších zdrojů
3. Vyhodnoťte výsledky systému na datových sadách získaných z rozsáhlých projektů pro extrakci informací z webových zdrojů.
4. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- Manning, C. D., Schütze, H., Foundations of Statistical Natural Language Processing, MIT Press, 1999, ISBN 0-262-13360-1.

Pro udělení zápočtu za první semestr je požadováno:

- Funkční prototyp řešení

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce je zaměřena na strojové techniky, které jsou využívány při zpracování přirozeného jazyka a extrakce informací z textu. Přibližuje obecné metody začínající zpracováním surového textu, až po extrakci vztahů ze zpracovaných jazykových konstrukcí a uvádí možnosti využití pro získaná relační data, které je možné vidět například u projektu DBpedia. Dalším milníkem této práce je návrh a realizace automatického systému pro extrakci informací o entitách, které nemají vlastní článek na anglické verzi Wikipedie. Práce představuje vytvořené algoritmy pro extrakci entit s vlastním jménem, ověření existence článků extrahovaných entit a nakonec samotnou extrakci informací o jednotlivých entitách, které lze využívat při kontrole konzistence informací. Na závěr je možné zhlédnout dosažené výsledky a návrhy dalšího vývoje vytvořeného systému.

Abstract

This bachelor thesis is dedicated to mechanical techniques that are used in the natural language processing and information extraction from particular text. It is approaching the general methods that starting to process the raw text and it continues to the relations extraction from processed language constructs, moreover it provides options for the use of obtained relational data which can be seen for example in the project DBpedia. Another milestone of the described bachelor thesis is the design and implementation of an automated system for extracting information about entities, which do not have their own article on the English version of Wikipedia. Thesis also presents algorithms developed for the extraction of entities with their own name, the verification of the articles 'existence of the extracted entities and finally the actual extraction of information about individual entities, which can be used during the information consistency checking. In the end, it can be seen the results and suggestions for further development of the created system.

Klíčová slova

Wikipedia, korpus, DBpedia, koreference, extrakce informací, NLP, rozpoznávání jmenných entit, Open Information Extraction, kontrola konzistence, extrakce entit

Keywords

Wikipedia, corpus, DBpedia, coreference, information extraction, NLP, named entity recognition, Open Information Extraction, consistency checking, entity extraction

Citace

STEJSKAL, Jakub. *Kontrola konzistence informací extrahovaných z textu.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Smrž Pavel.

Kontrola konzistence informací extrahovaných z textu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Pavla Smrže. Další informace mi poskytl Ing. Jaroslav Dytrych. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Stejskal
17. května 2016

Poděkování

Zde bych rád uvedl poděkování svému vedoucímu Doc. Pavlu Smržovi za jeho čas při odborných konzultacích, týkajících se této práce a za správné nasměrování při řešení nalezených problémů, Ing. Jaroslavu Dytrychovi za rady při experimentování ve výpočetním clusteru a za dobré typografické postřehy a v neposlední řadě své rodině a přátelům za velkou podporu.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Základním problém kontroly konzistence informací extrahovaných z textu je jejich podoba. Pokud se chceme zabývat tímto problémem nebo extrakcí informací z textu obecně, musíme prvně zpracovat samotný nestrukturovaný text, který slouží jako zdroj informací. Zpracováním nestrukturovaného textu se rozumí soubor operací, jejichž výsledkem jsou relační data, která je následně možné snáze zpracovat automatickými systémy, jenž jsou určeny pro daný relační model. Obecně můžeme říci, že automatické zpracování informací z textu vyžaduje, aby stroj, který text zpracovává, měl určitou míru porozumění danému typu textu.

V následující technické zprávě budou rozebrány a přiblíženy současné problémy automatického zpracování a extrakce informací z textu. Kapitola 1 představuje lingvistické metody pro analýzu textu, které jsou často prováděny jako předzpracování textu před pokročilejším zpracováním textového obsahu.

Kapitola 2 přiblíží historii extrakce dat, obecné, v současnosti používané, principy pro extrakci informací, zpracování a strukturování extrahovaných dat a jejich následnou dostupnost. Nakonec přiblíží současné projekty, které se tímto problémem zabývají jako například projekt *DBpedie*.

Kapitola 3 se zaměřuje na vlastní návrh systému pro kontrolu konzistence určitého typu informací z nestrukturovaného zdroje. Návrh popisuje korpus textu, ze kterého probíhá extrakce. Jak již bylo zmíněno výše, je nutné vstupní korpus upravit tak, aby systém mohl jednoduše extrahovat a zpracovat důležité informace. Tato úprava zahrnuje několik kroků a právě tyto kroky pro úpravu korpusu jsou zahrnuty v této kapitole. V poslední části jsou popsány postupy, které je nutné aplikovat pro správnou extrakci a kontrolu nalezených informací.

Po návrhu systému následuje přiblížení jeho implementace v kapitole 4. Tato kapitola zahrnuje použité technologie a popis problémů, které bylo nutné řešit.

Jako vstupní korpus pro vytvořený systém slouží data anglické verze internetové encyklopedie *Wikipedie*. Kapitola 5 se zabývá experimentováním a interpretací výsledků, které byly vytvořeným systémem dosaženy na vstupním korpusu, včetně naměřených metrik. V závěru kapitoly jsou shrnuty přínosy systému pro problém automatického zpracování textu, extrakce informací a kontrole jejich konzistence, kterým se celá práce zabývá a jeho případný další vývoj.

V závěru je připraven souhrn této práce..

Kapitola 1

Zpracování přirozeného jazyka

Než je možné přistoupit k samotné extrakci informací automatickým systémem, je nutné provést předzpracování vstupního textu do takové podoby, ze které dokáže stroj získat podstatné informace. Tato kapitola přibližuje základní kroky, které jsou využívány pro analýzu textu a slouží jako odrazový můstek pro pokročilé zpracování textu jako je například extrakce informací.

1.1 Korpus

Korpus je označení pro počítačový soubor, obsahující uložené texty, případně přepisy mluveného slova, který slouží pro účely jazykového výzkumu [10]. Korpusy mají jednotný formát z důvodu snadného vyhledávání slov a slovních spojení a jejich obsah je totožný se všemi jazykovými jevy přirozeného jazyka. Díky této vlastnosti je možné korpus využívat pro jazykový výzkum na reálných datech v rozsahu, který dříve nebyl možný. Na rozdíl od internetu je jazykový korpus neměnná referenční entita, u které se vždycky dá zjistit přesná velikost, počet různých jazykových jevů atp. Tyto vlastnosti jsou pro algoritmy, které se využívají v oblasti zpracování přirozeného jazyka, klíčové z důvodu natrénování některých svých vnitřních konstrukcí, u měnící se povahy internetu toto není možné. I když je každý korpus velice rozsáhlý, jedná se pouze o vzorek daného jazyka, je tedy nutné brát tuto skutečnost v potaz při zpracovávání textu, a to i v případě, kdy je korpus co nejvíce reprezentativní částí daného jazyka případně jeho části, která je zkoumána.

Z důvodu snadnějšího zpracování korpusu se často provádí anotace jeho textové části. Anotování textu přidává do korpusu metainformace o textech, jako například kdo je autorem textu, odkud text pochází atp. anebo může přidávat informace v jednotlivým jazykovým jevům. Mezi anotace jazykových jevů patří například *tagování* a *lemmatizace*, které jsou více popsány v podkapitolách 1.5 a 1.6.

1.2 Segmentace textu na větné celky

Pro člověka je na denním pořádku rozeznávat v jakémkoli textu jednotlivé větné celky. Není pro nás nijak těžké určit, zda tečka ve větě ohlašuje konec věty, datum, zkrácené prostřední jméno atp. Člověk dosahuje v určování větných celků 100% úspěšnosti v závislosti na kontextu, kdežto počítač nemusí být přesný ve všech případech.

Určování větných celků je prováděno metodou, které se říká **segmentace (segmentation)**. Úlohou segmentace je rozdělit členitý přirozený text na jednotlivé věty s maxi-

mální přesností. Pro správné provedení segmentace je tedy nutné, aby počítač uměl správně rozeznat začátek věty a správně určil interpunkční znaménko, které označuje konec věty [21].

Za účelem správné segmentace se využívá několik strategií, jak správně určit konce jednotlivých vět. Jednou ze strategií je určení konce věty podle daných pravidel, což v praxi znamená, že počítač nalezne interpunkční znaménko a vyhodnotí jeho funkci na základě *tokenu*, který se vyskytuje před tečkou. V anglickém jazyce tato strategie dosahuje 95% úspěšnost. Další strategií je naučení konců vět ze vzorových dat, kde jsou konce vět označeny. Počítač si vytvoří seznam slov, za kterými je tečka, ale neznačí konec věty jako například v případě emailových adres, webových stránek atp. Tato strategie dosahuje úspěšnosti až 99,5 % na anglickém jazyce.

Tyto dvě strategie nejsou zdaleka jediné možnosti provádění segmentace textu, například dalším způsobem je využívání Neuronových sítí.

1.3 Tokenizace a tagování slovních druhů

Dalším krokem při zpracování textu a následné extrakci informací je rozdělení vět na jednotlivé větné členy. Tomuto kroku se říká *tokenizace*, jejím úkolem je správně rozeznat a označit sekvence znaků, které společně tvoří slovo s určitým sémantickým významem - *token*. Standardně jsou tokeny ohraničeny mezerou. Tokenizace je demonstrována na následující větě:

> Marie Machacek is an astrophysicist.

Výsledek tokenizace vzorové věty vypadá následovně (závorky značí jeden token):

> [Marie] [Machacek] [is] [an] [astrophysicist] [.]

Mezi problémové části textu pro tokenizaci se řadí například URL v textu, datum a slova obsahující apostrof.

Tokenizace, která se provádí pouze na základě bílých znaků, se označuje jako *naivní* a její výsledky nemusí vždy být přesné. Například čárka uprostřed věty nebude označena jako samostatný token, ale bude připojena k tokenu, u kterého je čárka zapsána. Pro větší přesnost a relevanci extrahovaných informací je potřeba větší přesnost tokenizace, kterou poskytuje například Stanford 2.0.3 tokenizer. Tabulka 1.1 zobrazuje přehled výsledků několika tokenizérů na stejné větě - *"I said, 'what're you?"*.

Další možnou anotací zpracovávaného textu je přidání metainformace, která specifikuje význam daného tokenu na základě kontextu věty. Tomuto kroku se říká **Part-of-Speech tagging - tagování**. K danému tokenu se přidá informace o slovním druhu a jeho bližší specifikaci. Tag tokenu určuje slovní druh a v závislosti na slovním druhu také jmenný rod, číslo, přivlastňovací rod, osobu a čas. Přehled tagů využívaných v Penn Treebank korpuse je možné vidět v tabulce 1.2 [23].

Během přiřazování tagu se bere v potaz kontext věty, ve které token figuruje a jeho vztah s okolními tokeny. Jako každý postup pro zpracování přirozeného jazyka i tokenizace není vždy stoprocentní, a proto automaticky zpracovaný kontext věty nemusí odpovídat skutečnosti. Zpracovávaný token může mít více možných významů v různých časech, například slovo *dogs* může značit podstatné jméno množného čísla anebo sloveso.

Pro tagování existují algoritmy založené na velkém množství ručně psaných pravidel - **Rule-based taggers**. Tyto algoritmy závisí na slovnících, kde jsou určeny správné tagy

#	Naivní	Apache Open NLP	Stanford 2.0.3	Ideální
1.	"I	"	"	"
2.	said,	I	I	I
3.	'what're	said	said	said
4.	you?"	,	,	,
5.		'what	'	'
6.		're	what	what
7.		you	're	are
8.		?	you	you
9.		"	?	?
10.			"	"

Tabulka 1.1: Tokenizace jedné věty pomocí různých systémů

vzorových tokenů a na seznamu pravidel, které správně určují tagy v závislosti na pozici ve větě. Další možností je použití pravděpodobnostních algoritmů - **Probabilistic tagger**. Tyto algoritmy používají vzorové korpusy, které jsou správně otagovány, k natrénování daného modelu textu. Pravděpodobnostní algoritmy vycházejí z principu skrytých Markovových modelů (Hidden Markov Model, HMM). Mezi největší problémy se aktuálně řadí určování pojmenovaných entit jako jsou třeba jména osob a míst.

1.4 Stemming a lemmatizace

Stemming je ve zpracování přirozeného jazyka proces, který na základě jazykových pravidel odstraní z původního slova všechny morfologické části, které nejsou součástí kořene slova a nalezený kořen pak vrátí jako výstup. Tento postup se uplatňuje například při vyhledávání informací v textu, kdy se aplikuje stemming na uživatelův dotaz, na který chceme odpověď a text, ve kterém chceme vyhledávat. Tento postup nám umožní vyhledávat informace nezávisle na tvaru slova.

Proces **lemmatizace** zpracovává slova do jejich základního tvaru, kterému se říká *lemma* [11]. Tento tvar je následně připojen k danému slovnímu tvaru jako morfologická anotace. Lemma vzniká odstraněním vlastností, které vznikly určitou úpravou, jako je například skloňování nebo časování. Lemmatizace bývá většinou součástí desambiguace slovních tvarů v textu, což je proces pro zjištění přesného významu slovního tvaru na základě kontextu. Problémem při lemmatizaci je určování lemmat víceslovných spojení, které dohromady mají jeden význam. Místo jednoho lemmatu získá ve spojení své lemma každý slovní tvar a následná extrakce informací z textu nemusí být správná. Výsledek lemmatizace usnadňuje práci se zpracovaným textem, kde je možné pracovat se základními tvary jednotlivých slov.

1.5 Syntaktická analýza

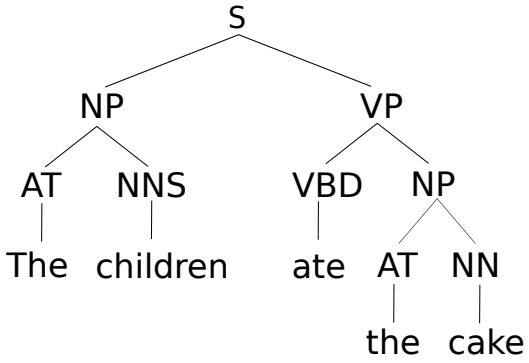
Obecně syntaktická analýza slouží k ověření správnosti gramatické struktury. U zpracování přirozeného jazyka nám syntaktická analýza udává, jak máme rozdělit význam věty na základě slov. Jedná se tedy o větný rozbor, který nám udává význam mezi jednotlivým slovy ve větě, anebo mezi částmi věty. Věty mají určitá pravidla, podle kterých jsou sesta-

#	Tag	Popis	#	Tag	Popis
1.	CC	Přiřazovací spojka	21.	RBR	Příslovce, komparativ
2.	CD	Kardinální číslo	22.	RBS	Příslovce, superlativ
3.	DT	Determinant	23.	RP	Částice
4.	EX	Existenciální "there"	24.	SYM	Částice
5.	FW	Cizí slovo	25.	TO	*to*
6.	IN	Předložka	26.	UH	Citoslovce
7.	JJ	Příd. jméno	27.	VB	Sloveso, infinitiv
8.	JJR	Příd. jméno, komparativ	28.	VBD	Sloveso, minulý čas
9.	JJS	Příd. jméno, superlativ	29.	VBG	Sloveso, předpřítomný čas
10.	LS	List item marker	30.	VBN	Sloveso, příčestí minulé
11.	MD	způsobové sloveso	31.	VBP	Sloveso, j.č, ne v 3. os.
12.	NN	Podst. jméno	32.	VBZ	Sloveso, 3. os., j. č.
13.	NNS	Podst. jméno, mn. č.	33.	WDT	Wh-determinant
14.	NNP	Vlastní podst. jméno, j. č.	34.	WP	Wh-zájmeno
15.	NNPS	Vlastní podst. jméno, mn. č.	35.	WP$	Přivlastňovací wh-zájmeno
16.	PDT	Vymezovací zájmeno	36.	WRB	Wh-příslovce
17.	POS	Přivlastňovací koncovka	37.	.	Tečka označující konec věty
18.	PRP	Osobní zájmeno	38.	,	Čárka mezi větami
19.	PRP$	Přivlastňovací zájmeno	39.	:	Dvojtečka
20.	RB	Příslovce	40.	()'"_	Další možné symboly

Tabulka 1.2: Přehled tagů pro slovní druhy využívaných v Penn Treebank korpusu.

vovány. Na základě těchto pravidel je možné určit, jaký dopad má pořadí slov na význam věty. Na získání struktury věty se obvykle používá jedna ze dvou metod:

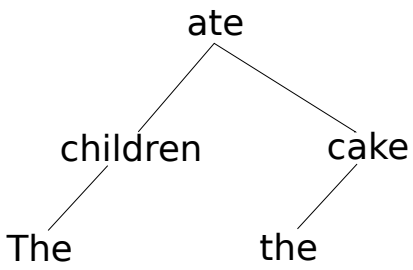
- **Složková analýza** - princip této metody spočívá v rozdělení vět na jednotlivé části, které rozděluje dále, dokud nenarazí pouze na jednu složku z věty. Výsledkem této metody je derivační strom, kde hlavním kořenem je větný přísudek. Závislosti mezi jednotlivými složkami jsou značeny orientovanými hranami. Příklad složkové analýzy můžete vidět na obrázku 1.1.



Obrázek 1.1: Příklad grafu složkové analýzy.

- **Závislostní analýza** - tato metoda funguje na principu závislostí mezi jednotlivými

členy. Nadřazený člen je vždy ten, na kterém je jiný člen závislý. Výsledkem této metody je stejně jako u složkové metody derivační strom, kde kořeny tvoří členy, na kterých jsou ostatní závislé. Příklad závislostní analýzy můžete vidět na obrázku 1.2.



Obrázek 1.2: Příklad grafu závislostní analýzy.

Výsledek syntaktické analýzy pomáhá počítači porozumět vztahům mezi jednotlivými slovy ve větě, případně mezi částmi vět.

Kapitola 2

Extrakce informací z textu

Kapitola Extrakce informací z textu uvádí základní principy, problémy a možnosti při extrakci informací z textu. Kapitola je rozdělena do několika podkapitol, které čerpají informace především z knih *Foundations of Statistical Natural Language Processing* [20] a *An Introduction to Information Retrieval* [19].

2.1 Základní seznámení s extrakcí informací

Extrakce informací z textu je v současnosti chápána jako strojem prováděná činnost, která zpracovává nestrukturovaný zdrojový text do podoby strukturovaných informací, které jsou následně ukládány do databáze, kde slouží k dalšímu využití. Vstupem systémů, které provádí extrakci informací z textu, jsou rozsáhlé textové celky - korpusy (podrobnější popis korpusů byl uveden v podkapitole 1.1).

Extrakce informací z textu - IE je součástí rozsáhlejšího celku, který se nazývá **Natural Language Processing (NLP)**. NLP se zabývá zpracováním přirozeného jazyka na takovou formu jazyka, které rozumí počítač. Mezi největší přínosy tohoto oboru patří rozbor a zpracování vět, extrakce jmenných entit a zpracování koreferencí. Získáváním informací z textu se zabývá odvětví **Information Retrieval (IR)**[19], které z textu získává strukturovaná data pro další využití.

Výsledná strukturovaná data obsahují informaci, se kterou může stroj dále pracovat. Mezi nejčastěji extrahované informace patří například jmenné entity, číselné hodnoty vztahující se k entitám, vztahy mezi entitami atp.

Databáze extrahovaných informací může mít nespočetně mnoho podob, kde každá položka databáze musí mít určitou hranici. S tímto přichází problém, jak takovou hranici určit. Počítač nemusí v každém případě přesně extrahovat požadovanou informaci, a proto je nutné vyhodnocovat přínos vyextrahované informace. V textu se může daná informace vyskytovat na více místech, je tedy nutné provést rozhodnutí, který výskyt informace je pro danou položku databáze nejvýhodnější.

2.2 Historie

Historie zpracování přirozeného jazyka sahá do padesátých let devatenáctého století, kdy vznikl první stroj pro překlad textu. Vytvořený stroj dokázal, na základě daných gramatických pravidel a slovníku, přeložit přes šedesát vět z ruštiny do angličtiny. Extrakce

informací přišla na řadu až o dvě desetiletí později, a to v sedmdesátých letech devatenáctého století. V této době se hodně programátorů zaměřilo na převod strukturovaných informací do takové podoby, které mohl rozumět počítač.

V roce 1987 vznikl první ročník **Message Understading Conferences (MCU)**, který extrakci informací značně popohnal vpřed. Tento projekt byl financovaný americkou vládní agenturou ministerstva obrany *DARPA*, zahrnoval několik skupin programátorů, kteří proti sobě soutěžili v extrakci informací z textu. Tato soutěž měla 7 ročníků v rozmezí od roku 1987 až 1997. Každý ročník přicházel s novými pravidly pro podobu extrahovaných informací a vstupními texty. Mezi nejzásadnější vývoj tohoto projektu patří 6. ročník, kde bylo zavedeno rozeznávání pojmenovaných entit a koreference [13].

Nástupcem MCU se stal v roce 1999 projekt **Automatic Content Extraction (ACE)**, který pokračoval ve stejném duchu jako jeho předchůdce. Mezi hlavní rozdíly od MCU patří zpracování entit. Již nestačilo pouze určit, zda se jedná o entitu, ale bylo stanoveno několik kategorií entit, do kterých bylo nutné každou entitu přiřadit, určit vztah mezi nalezenými entitami a v neposlední řadě detekovat a charakterizovat události, které jsou v textu zmíněny [4].

V současnosti se pořádá *Text Analysis Conference (TAC)*, která se zaměřuje na **Knowledge Base Population (KBP)** [15]. Hlavním účelem KBP je vyhledávání faktů o entitách a rozšiřování *Knowledge Base (KB)* nalezenými informacemi. V tomto případě jsou jako vstupní data extrakčním systémům předávány rozsáhlé textové dokumenty, webové stránky apod.

2.3 Metody extrakce informací

Zpracování přirozeného jazyka a extrakce informací využívá pro správné vyhodnocování metody strojového učení s učitelem [17]. Systémy se naučí principy, podle kterých zpracovávají text, ze správně zpracovaného textu v korpusu, který poskytuje učitel a následně je mohou aplikovat při zpracování přirozeného jazyka. Korpusy, ze kterých se systémy učí, musí být rozsáhlé z důvodu většího pokrytí jazykových konstrukcí, které nabízí přirozený jazyk. Přirozený jazyk je velice rozmanitý, a proto není možné pokrýt všechny jazykové konstrukce. Systémy se často musí rozhodovat, jaké je nejlepší zpracování dané části textu tak, aby výsledek byl co nejpřínosnější pro uživatele.

2.3.1 Metody založeny na pravidlech

Rule-based metody jsou založeny na vytvoření seznamu pravidel pro zpracování textu nebo extrakci informací. Systémy se následně tyto pravidla naučí, aby se mohli při zpracovávání textu rozhodovat na jejich základě. Během zpracovávání textu systémy hledají správné pravidlo, podle kterého daný text zpracují. Tyto metody mají slabé místo ve tvorbě pravidel, které musí tvořit člověk. Místo tvoření pravidel se také využívá strojové učení, které bylo popsáno v podkapitole 1.8. Jejich výkonnost je dobře vidět při zpracování menších textových celků, na druhou stranu pro větší textové celky se hůře implementují.

2.3.2 Skryté Markovovy modely

Skryté Markovovy modely (Hidden Markov model - HMM) [24, 25] je statistická metoda, kde z modelovaného systému neznáme posloupnost stavů, které vedou k výsledku, ale známe pouze pravděpodobnostní funkci pro jednotlivé stavy. Tato metoda

se díky svým vlastnostem, které jsou silně podloženy statistikou a jsou vhodné pro rychlé zpracovávání přirozeného jazyka, využívá nejen ve zpracování prostého textu, ale také při zpracování ručně psaného textu, rozpoznávání fyzických gest, anebo v bioinformatice. Nevýhodou HMM je nutnost znát topologii modelovaného systému. Stejně jako většina statistických metod i HMM potřebují rozsáhlé množství dat pro trénování vnitřních konstrukcí.

2.3.3 Rozpoznávání jmenných entit - NER

Pojem jmenná entita označuje slovo nebo několik slov, které označují a pojmenovávají určitou reálnou nebo smyšlenou entitu. **Rozpoznávání jmenných entit (Named Entity Recognition - NER** [16] je proces, který tyto entity nalezne v nestrukturovaném textu, označí je a nakonec rozhodne, o jaký typ entity se jedná (osoba - person, organizace - organization, místo - location apod.). Pro extrakci informací je mnohdy potřebné správně rozeznat pojmenovanou entitu v textu, a proto se NER řadí mezi důležité úlohy při extrakci informací a je mu věnováno spoustu výzkumů. Počátek rozeznávání jmenných entit se datuje do roku 1995, kde bylo zařazeno mezi úkoly na 6. ročníku *Message Understading Conferences (MCU)*.

Identifikace jmen probíhá na základě množiny vzorů. Vzory jsou tvořeny na základě rysů, které mohou jména obsahovat jako je například velké počáteční písmeno a okolní větné členy. Příklad je předveden na neanotované větě:

> Mark Zuckerberg is co-founder of Facebook.

NER systémy rozpoznají jmenné entity a přiřadí je do správné kategorie. Výsledek vypadá takto:

> [Mark Zuckerberg]$_{person}$ is co-founder of [Facebook]$_{organization}$.

Problém při rozeznávání jmenných entit nastává při zpracování kontextové závislosti pro danou entitu. Například v textu nalezená entita *"JFK"* může referovat na osobu amerického prezidenta *Johna F. Kennedyho* anebo na místo *JFK Mezinárodní Letiště*, letiště pojmenované po zmíněné osobě. Zpracování kontextu je tedy nedílnou součástí NER systémů, aby bylo správně určeno, o jaký typ entity se jedná.

Mezi první metody, které NER začal využívat pro rozpoznávání entit, jsou metody založené na pravidlech. Problém používání těchto metod je stejný jako při použití u extrakce informací, tedy je nutné vytvářet pravidla, na základě kterých bylo možné entity rozpoznat. Novější NER systémy využívají statistické metody.

2.3.4 Extrakce vztahů

Extrakce vztahů [14] patří podobně jako NER k důležitým úlohám při extrakci informací z textu. Úkolem této úlohy je nalezení a následné charakterizování sémantiky vztahu mezi entitami v textu. Pro příklad extrakce vztahu je využita stejná věta jako v předchozí podkapitole:

> Mark Zuckerberg is co-founder of Facebook.

Budeme uvažovat, že na testovací větě už byla provedena anotace, a tudíž jsou správně označené entity. Na základě zpracování kontextu systém vyextrahuje tento vztah:

FounderOf(Mark Zuckerberg, Facebook)

Pro extrakci vztahů jsou využívány metody anotace textu jako jsou tagování, lemmatizace a pro vyhodnocení závislosti mezi entitami využívají některé systémy syntaktickou analýzu. Jedním z hlavních výzev při extrakci vztahů je správně vyřešit *koreferenci* . Koreferencí se rozumí odkaz na entitu v dané větě ve formě zájmena. Ve větě pak není přítomná přímo entita, ale pouze zájmeno, které na ni odkazuje. Člověk koreferenci řeší prakticky pořád při jakékoli čtení textu, ale pro automatický systém není tato metoda jednoduchá. Extrakce vztahů se hodně využívá v systémech, které odpovídají na otázky.

2.3.5 Open Information Extraction

Dosavadní principy pro extrakci vztahů z textu potřebovaly lidský zásah ve formě vytvoření pravidel pro extrakci nebo ručně vytvořeného korpusu, kde by si systémy mohli nacvičit jednotlivé rozhodovací principy. **Open Information Extracion - OIE** [6] se v tomto velmi liší od standardních metod. Pro extrakcí relačních vztahů stačí pouze jeden průchod přes vstupní korpus, díky němuž je systém škálovatelný na velikost korpusu a jako vstup vyžaduje pouze korpus textu, ze kterého chceme extrahovat vztahy. Tento průchod je proveden bez lidského zásahu a na výstup jsou získány nalezené vztahy mezi entitami. Tato metoda se velmi hodí pro extrakci vztahů z internetových korpusů, které se od sebe velikostně mohou velmi lišit.

2.4 Metriky systémů pro extrakci

Každý systém na extrakci informací a zpracování přirozeného jazyka funguje správně jen na určité procento. Vytvořené systémy v tomto odvětví zpravidla nedosahují 100% úspěšnosti, a proto jsou zde metriky, které po vypočítání udávají kvalitu existujících systémů. Mezi nejhlavnější metriky patří **přesnost - precission(P)**, **pokrytí - recall(P)** a **F-hodnocení - F-measure(F)**. Pro výpočet těchto metrik je zásadní, že výstup jakéhokoliv systému na extrakci informací nebo zpracování přirozeného jazyka, dokážeme ohodnotit jako správný nebo špatný.

Pro lepší pochopení nám poslouží kontingenční tabulka 2.1 a vysvětlení jejich prvků.

- **Správně pozitivní - TP** - množina dat, které systém označil jako přínosné, bude s nimi nadále pracovat a ve skutečnosti jsou také přínosná.
- **Špatně negativní - FN** - množina dat, které systém označil jako nepřínosné, nebude s nimi nadále pracovat, ale ve skutečnosti se jedná o přínosná data.
- **Špatně pozitivní - FP** - množina dat, které systém označil jako přínosné, bude s nimi nadále pracovat, ale ve skutečnosti jsou data nepřínosná.
- **Správně negativní - TN** - množina dat, které systém označil jako nepřínosné, nebude s nimi nadále pracovat, a ve skutečnosti jsou data také nepřínosná.

Přesnost nám udává množinu dat, které systém vrátí jako výsledek, jenž jsou systémem vyhodnoceny správně. Přesnost můžeme vypočítat pomocí rovnice 2.1 a dosazením údajů, které jsou popsány v tabulce 2.1.

$$P = \frac{TP}{TP + FP} \tag{2.1}$$

	Relevantní	Nerelevantní
Získané	Správně pozitivní (TP)	Špatně pozitivní (FP)
Nezískané	Špatně negativní (FN)	Správně negativní (TN)

Tabulka 2.1: Kontingenční tabulka využívaná pro určování kvality extrakčních systémů

Pokrytí systému nám udává hodnotu, která odpovídá množině pozitivních nálezů dat. Pokrytí lze vypočítat pomocí rovnice 2.2.

$$R = \frac{TP}{TP + FN} \tag{2.2}$$

F-hodnocení vytváří harmonický průměr z přesnosti a pokrytí a komplexněji hodnotí systém na základě získaných výsledků a vypočítaných metrik pro hodnocení.

$$F = \frac{2PR}{P + R} = \frac{2TP}{2TP + FP + FN} \tag{2.3}$$

2.5 Současné projekty pro extrakci

V současnosti se extrakcí informací zabývá velké množství programátorů, a proto v posledních letech vzniklo několik automatických systémů, které řeší jednotlivé části zpracování textu anebo přímo extrakci informací, které nejsou specializovány jen na určité korpusy, ale jsou snadno škálovatelné na různé druhy korpusů jako například internet (WWW), Wikipedia, sociální sítě apod. Několik těchto systémů a projektů je přiblíženo v následujících podkapitolách.

2.5.1 DBpedia

DBpedia [18] je v současnosti rozsáhlý projekt, který extrahuje nestrukturovaná data z Wikipedie a vytváří znalostní databázi, kterou poskytuje online pro další zpracování. Zajímavost DBpedie spočívá v tom, že Wikipedia je nejvíce využívaná online encyklopedie na světě, obsahující články ve více než 250 jazycích z celého světa. DBpedia extrahuje informace ze 125 jazykových verzí Wikipedie. Největší znalostní databáze DBpedie, extrahovaná z anglické verze Wikipedie, obsahuje přes 400 milionů faktů a popisuje přes 4.58 milionů položek, které jsou nadále klasifikovány a rozděleny podle významu do tříd jako například lidé, místa, umělecké práce, organizace apod. Pro představu je anglická znalostní databáze poloviční velikosti oproti spojeným ostatním znalostním databázím, obsahující data z Wikipedie. DBpedie zveřejňuje všechny své extrahované znalostní databáze a navíc poskytuje online přístup ke 14 znalostním databázím pomocí dotazovacího jazyku SPARQL[1].

Extrahovaná data jsou ukládána na serverové uložiště, namodelovaná podle standardu RDF[2] pro výměnu dat přes webové rozhraní. Princip extrakce informací na DBpedie je rozdělen do čtyř kroků:

- **Vstup** - na vstup extraktoru jsou přiváděny stránky z Wikipedie, které jsou čteny z externího zdroje, jimž může být stažený dump anebo je možné stahování stránek online pomocí MediaWiki API.

[1] SPARQL je dotazovací jazyk, pomocí kterého je možné se dotazovat na data namodelovaná podle RFD standardu.

[2] RDF je standard pro vyjadřování informací ve formě, která je čitelná jak pro lidi, tak pro stroje.

- **Zpracování** - jednotlivé stránky jsou zpracovány pomocí wiki parseru, který přemění kód stránek na abstraktní syntaktický strom[3].
- **Extrakce** - vytvořené abstraktní syntaktické stromy každé stránky jsou předávány extraktorům, které vyextrahují potřebné informace. Každý extraktor se specializuje na určitý typ informací, které se v textu mohou vyskytovat. Výsledky extraktorů jsou vždy sady RDF příkazů.
- **Výstup** - shromážděné RDF příkazy jsou zapisovány do výsledné znalostní databáze, které jsou uloženy na serverovém uložišti.

Extrahovaná data jsou mapována do infoboxů, které určují, o jaký typ informace se jedná a ukládají informace do jisté struktury. Po namapování infoboxů se jejich obsah konvertuje na RDF trojice, které jsou následně zapisovány do databáze.

```
{{TemplateMapping
| mapToClass = Actor
| mappings =
   {{ PropertyMapping | templateProperty = name |
    ontologyProperty = foaf:name }}
   {{ PropertyMapping | templateProperty = birth_place |
    ontologyProperty = birthPlace }}
}}
```

Z infoboxu můžeme vyčíst, že typ informací je *Actor* a ukládají se dvě věci: jméno herce a místo narození. Pro tento infobox tedy vzniknou tři trojice RDF pro uložení v následující podobě:

```
dbpedia:Vince_Vaughn  rdf:type                dbpedia-owl:Actor   .
dbpedia:Vince_Vaughn  foaf:name               "Vince Vaughn"@en   .
dbpedia:Vince_Vaughn  dbpedia-owl:birthPlace  dbpedia:Minneapolis .
```

2.5.2 KnowItAll

KnowItAll je autonomní systém pro extrakci informací a vztahů z webových stránek [5, 12]. Systém vychází z malého množství generických šablon, pomocí kterých se generují pravidla pro extrakci jednotlivých informací dané třídy a z rozšiřitelné ontologie[4]. KnowItAll je dále založený na doménově a jazykově nezávislé architektuře, která pomocí několika modulů slouží k plnění ontologií konkrétními fakty a vztahy.

KnowItAll využívá internetové vyhledávací systémy pro výpočet statistik o vstupním korpusu (web), které využívá pro ohodnocení správnosti extrahovaných informací přiřazením pravděpodobnosti. Následně může automaticky zkontrolovat správnost informací. Extrakční mechanismus je plně automatický a pro rozšíření hledání stačí pouze rozšířit ontologii pomocí nové třídy nebo vztahu, systém vytvoří dle šablon pravidla pro hledání a zařadí extrakci dalších informací.

Problémem tohoto systému je rychlost, která souvisí s využíváním dostupných vyhledávačů informací na webu. Proto byl vyvinut následník, **KnowItNow**, který využívá vlastní

[3] Jedná se o datovou strukturu

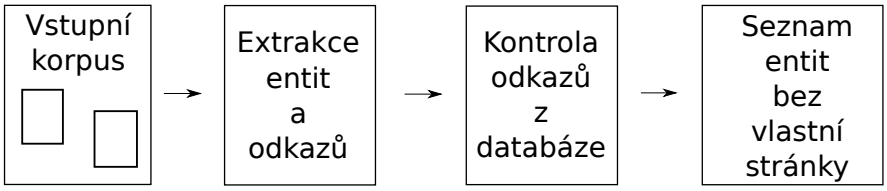
[4] Ontologie slouží k popisu oblasti lidského zájmu, která obsahuje objekty a jejich vztahy

vyhledávací princip označovaný jako *Bindings Engine* [8]. Tento princip eliminuje omezení v podobě počtu přístupů, což značně urychluje extrakci z několika dní na několik desítek minut.

Kapitola 3

Návrh systému na extrakci entit

Tato kapitola se zabývá návrhem systému, který ze vstupního korpusu dokáže extrahovat entity a informace, které se nalezených entit přímo týkají. Kapitola zahrnuje postup anotace a čištění korpusu, principy při hledání potřebných dat a přehled možností navrženého systému. Schéma navrženého systému je vyobrazeno na obrázku 3.1.



Obrázek 3.1: Stručné schéma funkčnosti výsledného systému.

3.1 Vstupní korpus

Každý systém pro extrakci informací z textu potřebuje jako vstup rozsáhlý korpus obsahující text v přirozeném jazyce. Rozsáhlejší systémy si dovedou poradit s větším počtem jazykových konstrukcí, než malé experimentální extraktory a od toho se také odvíjí zdroj korpusu. Zatímco výkonnější systémy dokážou zpracovávat zaznamenanou konverzaci, například z internetových fór nebo sociálních sítí, menší systémy se zaměřují spíše na spisovnou formu jazyka. Toto je i případ této práce, kde jako vstupní data byl zvolen volně dostupný korpus anglické verze *Wikipedie*.

Wikipedie obsahuje spisovnou formu anglického jazyka ve velmi rozsáhlém množství, a proto je tento korpus ideální pro výzkumné účely tohoto typu. Anglický jazyk je na rozdíl od češtiny celosvětově zkoumaný jazyk, na který se zaměřuje mnoho systémů v tomto odvětví, dále existuje mnoho podpůrných nástrojů pro práci s tímto jazykem, které jsou volně dostupné pro výzkumné účely. Anglický jazyk se řadí mezi analytické jazyky, což podporuje snadnější zpracování automatickými systémy než třeba čeština, která disponuje velkou volností co se ohýbání slov týče. Korpus anglické verze Wikipedie je také největší, který tato encyklopedie nabízí s celkovým počtem článků přesahující 5 milionů.

3.2 Úprava korpusu

Korpus distribuovaný Wikipedii není vhodný k okamžitému zpracování přirozeného textu a extrakci informací. Stažený soubor (dump) je ve formátu XML [3], kde krom samotného textu jsou uloženy také značky označující jednotlivé články, podrobnější informace k článkům jako třeba tituly, číslo revize apod. a v samotném textu značky, které upravují vzhled textu v závislosti na webové verzi článku. Tento soubor je tedy nutno upravit do takové podoby, která je přehlednější a vhodnější pro automatické zpracování přirozeného jazyka.

Zpracovaný dump je již k dispozici na školních serverech v anotované podobě a je rozložen na více stanic z důvodu rychlejšího zpracování jednotlivých částí. Není tedy nutné zpracovávat korpus znovu, ale je využita jeho zpracovaná podoba. Následující podkapitoly 3.2.1 až 3.2.3 stručně přibližují jednotlivé úpravy a výslednou podobu zpracovaného korpusu, který je použit v této práci.

3.2.1 Anotace textu

Jak již bylo zmíněno v podkapitole 1.1, texty v korpusu jsou často anotovány z důvodu snadnějšího zpracování přirozeného jazyka. Korpus využívaný v této práci také prošel anotací textu a to takové, že každý slovní tvar obsažený v korpusu má přiřazeno až 13 tagů upřesňující, případně rozšiřující, informace o daném slovním tvaru. Mezi hlavní anotace použité v tomto korpusu můžeme zařadit, z pohledu využitelnosti v této práci, segmentace na větné části (1.2), tokenizace a tagování (1.3), lemmatizace (1.4) a zpracování koreferencí. Každý slovní tvar, token, má tedy přiřazený tag o slovním druhu a svůj základní tvar. Mezi další anotované informace patří například position - pořadí tokenu ve větě, lower - verze tokenu napsaná pouze malými písmeny, link - URL tokenu (pokud je referencovaný na Wikipedii) a nertag - tato anotace obsahuje zařazení entity do třídy. Kompletní přehled anotací použitých v tomto korpusu je vidět v podkapitole 3.2.3.

3.2.2 Rozložení korpusu do výpočetního clusteru

Anotování korpusu je záležitost jednoho průchodu přes vstupní korpus, a proto není nutné anotovaný text soustředit do jednoho velkého souboru. V tomto případě je anotovaný korpus rozdělen mezi školní servery v rovnoměrném poměru. Hlavní důvod vyplývá z účelu, pro který byl vytvořen, a tím je extrakce informací. Zatímco anotovat text stačí pouze jednou a vícekrát zdroj procházet nemusíme, u extrakce informací zpracováváme anotovaný text několikrát, ať už z důvodu prostého ladění systému, testování stávajících algoritmů anebo přidání nových možností systému. Samotná extrakce nemusí být časově náročná, ale anotace rozsáhlého korpusu může zabrat desítky hodin a provádět tuto úlohu po každém spuštění systému by bylo velmi neefektivní.

Extrakci je pak možné jednoduše provést spuštěním systému na každém serveru, kde systém nebude procházet jeden velký soubor, ale několik menších, což výrazně zrychlí extrakci. Více informací o rychlosti extrakce této práce je k nalezení v kapitole číslo 5.

3.2.3 Výsledný vzhled korpusu pro extrakci

Výsledný korpus je rozdělen mezi několik serverů. Data na každém serveru jsou navíc rozdělena do několika souborů, řádově 5 až 10, kvůli zvýšení výkonu při extrakci (data

korpusu jsou využívána dotazovacím systémem, který pracuje na principu MG4J[1]). Každý soubor obsahuje následující informace:

- Název souboru - název souboru je vždy na prvním řádku a jednoznačně označuje daný soubor.
- Označení jednotlivých článků - články jsou označeny dvěma řádky, první řádek obsahuje značku `%%#DOC`, za kterou následuje hash označení daného článku, druhý řádek obsahuje značku `%%#PAGE`, za kterou následuje název článku a URL článku.
- Označení jednotlivých odstavců - odstavce jsou označeny značkou `%%#PAR X`, kde X značí pořadí odstavce v článku.
- Označení jednotlivých vět - jednotlivé věty jsou označeny značkou `%%#SEN Y`, kde Y označuje pořadí věty v článku.
- Tokeny jednotlivých vět - tokeny následují bezprostředně na dalším řádku po označení věty a všechny jsou anotovány pomocí metod zmíněných v kapitole 2. Příklad jednoduchého souboru, který simuluje zpracovaný korpus je k dispozici mezi přílohami.

3.3 Princip hledání dat pro extrakci

V podkapitole 3.2.1 bylo zmíněno, že anotovaný korpus obsahuje informaci, podle které lze nalezenou entitu zařadit do třídy. Vzniká tedy otázka, proč navrhovat systém pro extrakci entit z textu, když jsou již anotované a lze je snadno získat. Problém těchto anotací spočívá v tom, že nejsou správně anotovány a do tříd zařazeny všechny entity, které se v textu vyskytují.

3.3.1 Zpracování vstupu

Spuštěný systém prochází postupně jednotlivé soubory, které obsahují rozdělený korpus. Při procházení těchto souborů je však nutné oddělit od sebe jednotlivé články, které jsou následně zpracovány. Systém prochází jednotlivé řádky a všímá si značek, které na daném řádku jsou. Může tak zjistit, zda se jedná o začátek nové věty, odstavce anebo stránky, případně zda se jedná o token.

Pro efektivní práci byl zvolen postup, který z procházených řádků tvoří jednotlivé články. Udržuje strukturu odstavců a řadí za sebe věty ve formě, která stačí pro vyhledávání potřebných informací. Každý token již neobsahuje všechny anotované informace ze vstupního souboru, ale obsahuje pouze informaci o slovním druhu a lemmu, v případě, že daný token má vyplněnou anotaci o zařazení entity (nertag) a URL (link) jsou přidány i tyto informace, ale nejsou bezprostředně nutné u všech druhů tokenů. Každý vyextrahovaný článek je označen názvem článku společně s URL, které odkazuje na webovou verzi článku. Tyto informace je nutné přenášet z důvodu dohledatelnosti místa, odkud byla entita extrahována. Díky již provedené segmentaci textu na věty, není nutné hledat a vyhodnocovat konce vět na základě interpunkčních znamének, ale systém odděluje věty automaticky na základě označení začátku nové věty automaticky.

[1] MG4J je fulltextový vyhledávací princip pro vyhledávání v rozsáhlých kolekcích dokumentů [7]

3.3.2 Vyhledávání entit

Předání vyextrahovaného článku je začátek pro vyhledávání entit. Nyní přichází na řadu prohledávání vět a samotné vyhledávání entit v textu. Pro začátek je nutné si uvědomit, jak vlastně taková entita může vypadat z gramatického hlediska. Entita, ať už se jedná o člověka, místo anebo pojmenovaný časový úsek, může obsahovat jedno nebo více slov, ovšem většina by měla mít velké písmeno. Existují určité výjimky jako třeba předložky v pojmenování (*King Robert of the House Baratheon, Battle of Midway atp.*), které se v entitách mohou vyskytovat. Jednoslovné entity by měli mít minimálně začínající písmeno velké (*Facebook, Madona, IBM atp.*). Tyto skutečnosti je nutné brát v potaz při hledání entit v textu. Samotná extrakce entit zahrnuje několik postupu ověřování, zda nalezený token může být entitou.

Systém nejprve rozdělí článek na jednotlivé věty a následně prochází všechna slova, která jsou obsažena ve větě. Při procházení vět si systém pamatuje celou větu a vyhodnocuje procházené tokeny nejprve na základě jejich typu. Typ tokenu je rozpoznán podle anotovaného tagu, který nese informaci o slovním druhu. Jelikož systém věty prochází postupně token po tokenu, není možné ověřovat hned entitu jako celek, ale pouze vyhodnotit, zda se token může v entitě nacházet, a v případě kladného vyhodnocení si stávající token zapamatovat. Entita jako celek je ověřována až v případě, že systém si zapamatoval několik po sobě jdoucích tokenů, které by mohli entitu tvořit a tento sled byl ukončen tokenem, který se v entitě vyskytovat nemůže. Systém tedy prochází větu a vytváří si posloupnost tokenů, které po přerušení jejich sledu vyhodnotí jako celek.

Získaná posloupnost tokenů je nyní podrobena několika ověřením, která rozhodnou, zda se skutečně jedná o relevantní entitu:

- **Hledání entitní třídy** - systém zachovává u všech tokenů jejich anotace, které obsahovali při vstupu a je proto možné odhalit ty entity, které již byly objeveny a zařazeny. Pokud některý token z posloupnosti obsahuje třídu, do které se entity řadí, je posloupnost vyhodnocena jako validní a tudíž zapsána jako nalezená entita.
- **Ověření počtu velkých písmen** - dalším ověření je porovnání celkového počtu tokenů a tokenů začínajících na velké písmeno. V tomto případě nejsou validně vyhodnoceny případy, kdy posloupnost tokenů neobsahuje minimálně dva tokeny s velkým počátečním písmenem.
- **Vyloučení často používaných podstatných jmen** - v některých případech systém nalezen tokeny, které odpovídají entitě, ale z hlediska kontextu se o entitu nejedná. Pro příklad si můžeme uvést entitu *Bank of America*, kde se jedná o určitou entitu, ovšem pokud v textu nalezneme pouze *Bank* jinde, než na začátku věty, nejedná se s největší pravděpodobností o entitu, ale pouze o zvýraznění v textu pomocí velkého písmena. Princip tohoto ověření je popsán v podkapitole 3.3.3.

3.3.3 Filtrování negativních nálezů

Jednou ze zásadních věcí, kterou musí systém provádět, je odfiltrování negativních nálezů, což znamená, že musí odhalit slovní spojení, které se tváří jako entita, ale jedná se pouze o slovní tvar nebo spojení, které obsahuje velká písmena. Takové nálezy je třeba odfiltrovat, protože navržené vyhledávání entit, které je v systému využito, tyto nálezy vyhodnotí jako správné a to hlavně na základě tagů, které souhlasí s tagy u entit.

Základem filtru negativních nálezů je databáze nejčastěji používaných slov v anglickém jazyce. Pro účely výsledného systému byl tento seznam sestaven ručně pomocí volně dostupných statistik, které obsahují informace o použití všech slovních druhů z různých zdrojů. Aktuálně využívaný seznam obsahuje přes 1600 slov, které samostatně nemohou označovat entitu. Tento seznam je systémem využíván při extrakci jednotlivých entit, které jsou na základě tohoto seznamu ověřovány.

Ke správné funkci filtrování však nestačí pouze samotný seznam a ověření, zda se v entitě nevyskytuje nějaké slovo, které by naznačovalo nesprávnost entity. Je nutné ověřit kontext entity v takové míře, aby od sebe byly rozeznány případy, které byly zmíněny v podkapitole 3.3.2. V případě zmíněného slova *Bank* je potřeba zjistit, zda se jedná o určitou banku pomocí zbylé části extrahované entity. Pokud bychom chtěli systém vylepšit a získat informace také o entitách, které jsou tímto filtrem odstraněny, bylo by nutné místo filtrování slov zapojit zpracování kontextu věty spolu se zbytkem článku. Pokud bychom pak našli entitu *Bank*, bylo by možné zjistit informace, kde taková banka stojí a ze zbytku článku přiřadit další doplňující informace. Tento systém však tuto schopnost neobsahuje.

3.3.4 Vytvoření seznamu nálezů

Seznam nalezených entit, který systém tvoří, neobsahuje pouze nalezené entity. Pro dohledatelnost každého nálezu je ke každému nálezu přiřazeno URL článku, ze kterého byla entita extrahována a věta, ve které byla nalezena. Tato trojice je nejprve uložena v paměti systému a po úspěšném zpracování celého článku je zapisována do souboru. Kvůli práci systému ve výpočetním clusteru však není možné, aby entity nalezené na všech serverech, mohli být zapisovány do jednoho výsledného souboru kvůli referenci na soubor, který se může měnit několikrát za vteřinu. Tento problém je řešen samostatným výstupním souborem pro každý server.

Vytvořené soubory jsou nakonec spojeny do jednoho souboru, ve kterém jsou seřazeny všechny vyextrahované entity. Algoritmus 1 obsahuje pseudokód extrakce entit z jednotlivých článků.

Algorithm 1: Pseudokód extrakce entit pro jeden článek

```
Data: Extrahovaný článek
Result: Seznam nalezených entit
inicializace seznamů;
while nejsi na konci článku do
    vezmi token z věty a zjisti typ;
    if token je podstatné jméno then
        vlož token do seznamu;
    else if token je spojka nebo pomlčka then
        if seznam tokenů není prázdný then
            vlož token do seznamu;
    else if seznam tokenů není prázdné then
        if vyhodnoť správnost entity then
            vlož entitu do seznamu spolu s větou a URL na článek;
        smaž seznam tokenů;
return seznam entit
```

V případě kontroly existence článků nejsou výsledné soubory spojeny hned po extrakci z důvodu větší rychlosti systému.

3.4 Extrakce entit bez článků na Wikipedii

Hlavním účelem navrženého systému je vyhledat entity ze třídy *person* (dále jen entity), které nemají svůj vlastní článek na Wikipedii. Princip vyhledávání těchto entit je stejný, jako je popsán v podkapitole 3.3.2. Rozdíl tkví v automatickém vyřazování entit, které již odkaz mají, případně jsou zmíněny na své vlastní stránce a nejsou zde odkazovány. V současnosti jsou stránky vytvářeny pomocí prefixu `https://en.wikipedia.org/` a názvu článku, v tomto případě názvu entity. Výsledné URL může vypadat třeba takto: `https://en.wikipedia.org/Albert_Einstein`. Problém nastává při špatném přiřazení odkazu v průběhu anotace. Pokud je entita anotována odkazem se stejným názvem, ale referuje o jiné entitě, současně navržený systém nerozpozná chybu. Více informací o tomto problému je k dispozici v podkapitole 4.4.3.

Každý odkaz na článek je originální, jinými slovy článek z jednoho odkazu nepojednává o dvou různých entitách, ale může existovat více entit se stejným jménem. V takovém případě by mohl jeden odkaz odpovídat více entitám, ale informace v článku se budou týkat pouze jedné z nich. Není tedy možné ověřovat existenci článku pouze na základě odkazu, ale je nutné porovnat informace o entitách, které odpovídají odkazu na daný článek. V této části přijdou na řadu věty, ve kterých se extrahované entity vyskytovali při jejich nálezu a odkazy všech stávajících článků společně se základními informace z článku.

3.4.1 Extrakce seznamu článků

Seznam odkazů na všechny články systém extrahuje během extrakce entit. Na každém serveru vytvoří soubor, do kterého postupně zapisuje odkazy všech procházených článků, které se vyskytují v části korpusu na serveru. Jelikož systém prochází každý článek, větu po větě, pro extrakci entit, z první věty každého článku je vyextrahována informace, o čem daný článek je. Vyhledává ve větě anotované tokeny, které odpovídají podmětu a přísudku a vystihují základní informace o stránce. V případě článku o entitě *Albert Einstein* systém vyextrahuje informaci *was physicist*. Samotnou entitu není třeba extrahovat, neboť je obsažena v získaném odkazu. Nakonec je nutné spojit seznamy odkazu na jednotlivých serverech do jednoho uceleného souboru na sdíleném serveru a nahrát všechny nalezené odkazy do databáze.

Wikipedie, jako každý webový portál, aktualizuje svoje informace, a tím se také mění. Změnou jsou poznamenány také odkazy na články, kdy na jeden článek může odkazovat několik odkazů, někdy velmi podobných. Přehled odkazů, které odkazují na určitý článek, nejsou ve zpracovaném korpusu, který je využíván pro extrakci entit k dispozici, a je nutné najít tyto odkazy jinde. Jako zdroj pro extrakci těchto odkazů slouží původní nezpracovaný korpus ve formě XML, který obsahuje také články, které samotný text nemají, ale pouze odkazují na jiný článek, který pojednává o stejné věci. Systém spustí na jednom ze serverů extrakci, která jako vstup použije neanotovaný XML korpus a vytvoří seznam dvojic odkazů v následujícím formátu: `odkaz -> odkazuje na`. Nálezy jsou poté připojeny do stejné databáze, jako seznam existujících článků, pouze s lehce odlišnými informacemi.

Extrakce tohoto seznamu bude vždy stejná, pokud se nezmění vstupní korpus, a proto stačí vyextrahovat tento seznam pouze jednou.

3.4.2 Kontrola existence článků nalezených entit

Konečná kontrola existence článků nalezených entit porovnává nalezené entity s vyextrahovanými seznamy odkazů uložených v databázi dvěma principy. První princip kontroluje odkazy vyextrahované z anotované verze korpusu, které obsahují informace o článku. Z procházeného seznamu nalezených entit systém vybere první položku a hledá ji v databázi extrahovaných odkazů na články. Pokud je entita nalezena, systém přejde k porovnání informací o entitách. Získané informace o článku jsou hledány ve větách, ve kterých se vyskytovala nalezená entita a na základě shody informací systém vyhodnotí, zda jsou obě totožné.

Druhý princip kontroly se nezaměřuje na existující články, ale kontroluje odkazy, které na existující články odkazují. V případě nalezené schody systém entitu vyloučí z výsledného seznamu. Tato část kontroly se zaměřuje hlavně na takové typy entit, které v minulosti existovaly pod jiným jménem. Pro představu můžeme uvést basketbalový klub *Newcastle Eagles*, který se v dřívějších letech jmenoval *Newcastle Comets*. Oba odkazy (`https://en.wikipedia.org/Newcastle_Eagles` a `https://en.wikipedia.org/Newcastle_Comets`) odkazují na tutéž stránku.

Výsledkem kontroly je vyfiltrovaný seznam entit s odkazy na článek a větami, ve kterých se vyskytuje. Výsledný seznam je ještě zpracován z důvodu zpracování vět, ve kterých byly entity nalezeny do podoby, která zachycuje hlavní informace o entitě.

3.5 Zpracování nalezených informací

S každou extrahovanou entitou je extrahována také věta, případně věty, ve kterých se nalezená entita vyskytuje v textu. Získané věty mají více účelů, ale mezi hlavní se řadí získání informací o extrahované entitě a jejich naformátování do strukturované formy, aby bylo možné jejich další využití. Navržená část využívá anotace u jednotlivých tokenů pro identifikaci jednotlivých větných členů a skládá je do podoby využitelných informací. Princip je předveden na jedné z vět extrahované entity *Dickie Bachmann*, z článku o basketbalovém klubu *Alaska Aces PBA*[2]:

> Dicky Bachmann - played for Alaska and is the current assistant coach of the team.

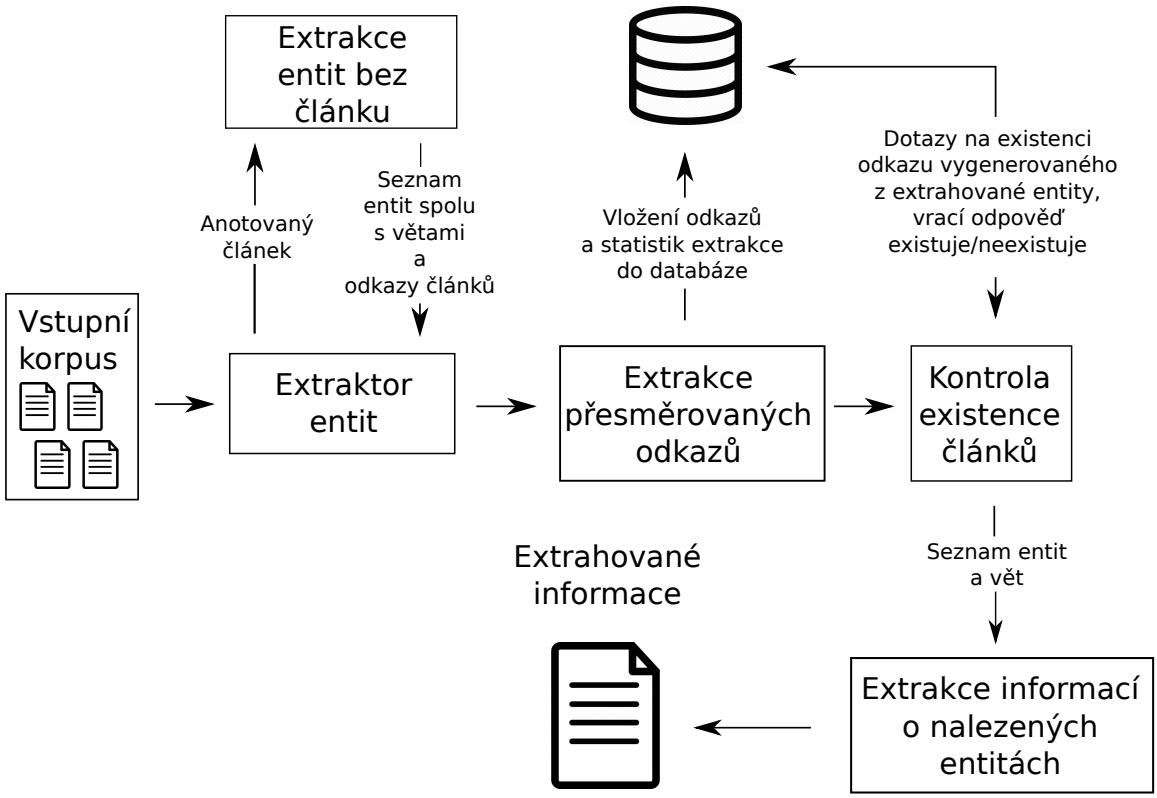
Prvním krokem zpracování je identifikace entity ve větě. Následně v pravé části věty od entity je hledáno sloveso. Nyní nastávají dvě možnosti, jak systém zareaguje:

- **Sloveso nalezeno** - v tomto případě je ve zpracovávané části věty hledán předmět, také na základě anotací textu. Při nálezu předmětu je sestavena výsledná informace ve formátu [entita] [sloveso] [předmět]. Sloveso je upraveno tak, aby si informace zachovala původní smysl. Informace nakonec vyhodnotí vztah k článku a v případě užitku rozšíří informaci o tento vztah.
- **Sloveso nenalezeno** - systém vyhledá předmět, v případě nálezu je vytvořena výsledná informace obsahující entitu a nalezený předmět. Jako sloveso je dosazeno slovo *byl* nebo *je* na základě slovesného času, který se nachází ve větě. Zde může nastat chyba při špatném určení času.

[2] Dostupné na: `https://en.wikipedia.org/wiki/Alaska_Aces_(PBA)`

Zbylou část věty systém zpracuje podobně, ale pro udržení celistvosti informace je v případě existence slovesa v levé části uložen zbytek věty. V uvedeném příkladu systém nalezne následující informace:

- Entity:Dickie_Bachmann played for https://en.wikipedia.org/wiki/Alaska_Aces_(PBA)
- Entity:Dickie_Bachmann is assistant coach of https://en.wikipedia.org/wiki/Alaska_Aces_(PBA).



Obrázek 3.2: Schéma výsledného systému.

Kapitola 4

Implementace systému

Tato kapitola přibližuje implementaci systému pro extrakci entit z pohledu využitých technologií a nalezených problémů. Popisuje důvody volby jednotlivých technologií, jejich výhody a přínosy usnadňující implementaci navrženého systému. Dále je zde uveden stručný přehled a řešení nalezených problémů.

4.1 Skriptovací jazyky Python, Bash a verzovací systém

Systém se skládá z několika částí, kde každá má za úkol jistou část extrakce/ověřování. Většina těchto částí je napsána ve vysoko-úrovňovém imperativním programovacím jazyce **Python** [22]. Mezi hlavní výhody toho jazyka patří dynamická kontrola datových typů, což značně zrychluje vývoj softwaru v tomto jazyce. Mezi další výhody patří možnost využití několika programovacích paradigmat, jako je například objektově orientované, procedurální a funkcionální. Díky volně dostupných knihovnám a různým rozšířením, je možné v tomto jazyce psát téměř jakékoliv programy, včetně těch s grafickým uživatelským rozhraním. Rozšíření pro tento jazyk obsahují již naimplementované některé rozšířené postupy, které mnohdy usnadní programátorovi práci.

Pro jednodušší části systému a pro různé podpůrné skripty byl využit skriptovací jazyk **Bash**. Bash byl vyvinut na konci osmdesátých let minulého století pro Projekt GNU. Jedná se o jazyk, který nabízí interakci mezi uživateli a unixovými systémy jako *POSIX shell*. Mezi jeho výhody patří rychlost operací, možnost kontroly procesů a také spouštění externích programů [2].

Verzovací systém **GIT** slouží v této práci pro správu jednotlivých verzí. Jedná se o distribuovaný systém správy verzí, který umožňuje uživateli uchovávat jednotlivé verze vyvíjeného softwaru a zpřístupnit provedené změny mezi jednotlivými verzemi. Díky tomu je možné efektivněji odhalovat vytvořené chyby, případně získat nazpět funkční kód minulé verze bez jakéhokoli programování navíc [9].

4.2 Databáze Elastic a formát JSON

Informace, které tento systém extrahuje, čítají většinou až 10 milionů řádků s proměnnou délkou, což může být nežádoucí při další práci s takto rozsáhlými daty. Pro uložení a urychlení zpracování takto rozsáhlých dat byl využit databázový systém **Elastic** a knihovna pro její ovládání **Elasticsearch** pro jazyk Python. Elasticsearch poskytuje distribuované fulltextové vyhledávání na serveru, kde se databáze nachází. Mezi hlavní výhody tohoto typu

databáze je vyhledávání dat a jejich využití nebo analýza v reálném čase. Ke všem operacím Elastic využívá *RESTful API*, které zasílá dotazy ve formátu **JSON** přes HTTP protokol a provede požadované akce. Není proto potřeba určitého programovacího jazyka pro obsluhu databáze [1].

Elastic nabízí možnost uložení dat v clusteru a velikou škálovatelnost, což je výhodné při neočekávanému růstu databáze. V takovém případě stačí pouze připojit nový server ke stávajícím a systém automaticky rozloží zátěž mezi servery. V jedné databázi také umožňuje vytvoření více indexů na vložení odlišných dat. Uložená data v databázi jsou získána jako jednotlivé dokumenty ve formátu JSON, ze kterých je možné získat potřebné informace, které zajímají uživatele. Elasticsearch je postaven na základě vyhledávací knihovny *Lucene*.

JavaScript Object Notation - JSON je univerzální, založený na podmnožině jazyku JavaScript, hojně využívaný formát pro výměnu dat. Jeho výhodou není jen nezávislost na programovacím jazyku, čímž je možné ho využívat ve většině využívaných programovacích jazycích, ale také čitelnost a snadné generování, jak pro člověka, tak pro stroj. JSON může nabývat několika různých datových typů:

- **Řetězec** - představuje posloupnost libovolné délky *Unicode* znaků uzavřených mezi dvojité uvozovky, využívá tzv. únikové sekvence[1].
- **Číslo** - číselná hodnota, může obsahovat desetinnou čárku, případně exponenciální tvar. Nepodporuje oktalový a hexadecimální tvar.
- **Objekt** - jedná se o neuspořádanou množinu párů ve tvaru **{"název":"hodnota"}**. Názvem tvoří řetězec znaků a hodnota nabývá libovolného datového typu.
- **Pole** - je seřazená kolekce hodnot, které jsou od sebe odděleny čárkou. Celé pole je uzavřeno to hranatých závorek a výsledný zápis vypadá například takto: [**hodnota1,hodnota2**].
- **Pravdivostní hodnota** - hodnota datového typu *boolean* v podobě **true** nebo **false**.
- **null** - prázdná hodnota v podobě klíčového slova *null*.

4.3 Specifikace využitého vývojového clusteru

Vývoj této práce, zejména pak její testování na větším objemu dat, bylo nutné provádět s větší výpočetní silou, než jako poskytují dostupné stolní počítače. Pro tento účel bylo možné využít školní výpočetní cluster využívaný *Výzkumnou skupinou znalostních technologií FIT VUT v Brně*. Díky přístupu k výpočetnímu clusteru bylo možné využít vysoký výpočetní výkon, který poskytuje přes třicet dostupných stanic. Pro účely této práce byly nejprve využity všechny servery pro rozložení náročnosti extrakce na více stanic, ověřování existence stránek jednotlivých entit z extrakce bylo prováděno pouze na jednom serveru. Náročnost tohoto systému nepřesáhla takovou hranici, aby cluster nemohl být využíván i ostatními uživateli, kteří mají oprávněný přístup.

[1] Přeloženo z anglického výrazu *escape sequence*

4.4 Problémy navrženého systému

Extrakce informací z textu sebou přináší spoustu problémů, které je potřeba řešit. Obecné problémy byli zmíněny v předchozích kapitolách. Tato část se zaměřuje na problémy, které vznikly při implementaci navrženého systému a bylo nutné najít řešení, případně přijatelnější alternativu.

4.4.1 Přístup k seznamům odkazů pro ověření

Extrahovaný seznam entit obsahuje milióny nálezů, které je nutné porovnat s nalezenými odkazy. Celkově existuje na anglické verzi Wikipedie přes 5 miliónů článků a bylo nalezeno téměř 6 miliónů odkazů, které jsou směrovány na stávající články. Vzniká problém, jak efektivně ověřit milióny nálezů oproti 11 miliónům nalezených odkazů a jejich základních informací.

Jako první způsob bylo zvoleno procházení souboru, který obsahuje extrahované entity, řádek po řádku v jazyce Python a pomocí regulárního výrazů proběhne změna nalezené entity na její odkaz. Tento proces zahrnuje pouze připojení prefixu odkazu k entitě a změnu bílých znaků na podtržítka. Z důvodu rychlosti nebyl procházen soubor s nalezenými odkazy skriptem v jazyce Python, ale byl využit Bash pro rychlejší získání potřebných dat. Velikost dat však velice zpomalovala tento proces. Jedna entita byla takto ověřena řádově za několik vteřin, což není ideální řešení.

Prvním rozšířením ověřování bylo rozprostření ověřování do výpočetního clusteru, kde na každém serveru bylo spuštěno několik vláken. Rychlost ověřování byla redukována z několika dní na několik hodin, ale během ověřování bylo téměř nemožné využívat souborový server, a proto bylo nutné navrhnout lepší řešení.

Konečným řešením je uložení nalezených odkazů na články do Elastic databáze a následné dotazování na vložená data při ověřování entit. Z důvodu zahlcení serveru, na kterém běží databázový systém, je dotazování prováděno pouze z jednoho serveru ve výpočetním clusteru. Pro srovnání rychlosti je možné takto ověřit zhruba 300 extrahovaných entit za 10 vteřin.

4.4.2 Extrahované entity bez relevantních informací

Při extrakci entit jsou ukládány také věty, ve kterých se entita vyskytuje v textu, odkaz na stránku článku, ze kterého byla extrahována a základní informaci o článku, která určuje, o čem článek pojednává. Získané věty jsou využívány pro vyhodnocení shody extrahované entity s článkem, který prezentuje informace o stejnojmenné entitě. V takovém případě jsou využity informace uložené v Elastic databázi u daného článku a jsou hledány ve větách, ve kterých se vyskytuje extrahovaná entita. Tento způsob přestává fungovat v případě, že extrahované věty neobsahují žádné relevantní informace, které by jakkoli mohli entitu identifikovat.

Pro částečné řešení tohoto problému jsou využity informace o článku, ze kterého byla entita vyextrahována. Pokud například extrahujeme entitu ze stránky basketbalového klubu, ale ani jedna věta, ve které se vyskytuje, neobsahuje relevantní informace, nemůžeme ověřit, zda článek o entitě stejného jména prezentuje data o hráči basketbalu. V takovém případě se pravděpodobně jedná o stejné entity, systém tyto informace vyhodnotí a odstraní extrahovanou entitu z výsledného seznamu. V některých případech není možné extrahované informace porovnávat přímo, je tedy nutné rozhodnout pouze na základě shody extrahované entity a odkazu na článek, nesoucí stejný název jako extrahovaná entita.

4.4.3 Nekonzistentní vstupní korpus

V podkapitole 3.2 bylo řečeno, že navržený systém využívá již anotovaný korpus dostupný na školních serverech. Během testování vytvořeného systému byla odhalena nekonzistence v podobě anotace některých entit. Mezi možné anotace se řadí dva druhy odkazů. Jedním z nich je odkaz na článek, na který je uživatel přesměrován v případě kliknutí na entitu. Druhým odkazem je v podstatě stejný odkaz na jiné položce, který ovšem nikam nepřesměrovává, ale má pouze informativní charakter. Právě tento odkaz, který v online verzi korpusu není využíván, způsobuje problémy při extrakci entit bez vlastních článků. V některých případech takto anotované entity nastává případ, kdy připojený odkaz referuje o odlišné entitě se stejném jménem. Řešením tohoto problému může být vylepšení přiřazování těchto odkazů při anotaci textu, případě zapojení systému pro kontrolu konzistence entit a odkazů přímo při extrakci.

4.4.4 Zpracování seznamů

Stávající zpracovaný korpus obsahuje všechny textové části anotovaných článků. Problém nastává při vyhledání informací, které jsou uloženy v článcích v podobě tabulek. Tyto části jsou během anotace odstraněny z důvodu syntaxe tabulky, kterou anotační systém ignoruje. V tabulkách není uložen takový druh textu, který by bylo nutné zpracovávat a dolovat z něho informace, ale obsahují již strukturovaná data. Pro porozumění těmto datům je nutné získat ze článku, ve kterém se tabulka vyskytuje, informaci o druhu dat, která jsou v tabulce uloženy.

V některých případech výskytu takto interpretovaných dat není problém tabulku nezpracovávat, ale co se týče stránek, které obsahují informace v podobě seznamů, je pro získání informací jediným způsobem zpracovat tabulky. Stránky obsahující tyto seznamy je nutné extrahovat z původního korpusu Wikipedie ve formátu XML. Po získání jednotlivých článků společně s tabulkami nastává problém při jejich zpracování. Syntaxe jednotlivých tabulek může být odlišná a zpracování více různých typů tabulek je problémové. Další potíží při tomto zpracování je identifikace té části tabulky, které obsahuje potřebná a užitečná data. Aktuální stav vytvořeného systému neumožňuje automatické zpracování různých druhů tabulek.

Kapitola 5

Zhodnocení výsledků a experimentování se systémem

Poslední kapitola této práce, jak již bylo zmíněno v předchozích kapitolách, se zabývá experimentováním s vytvořeným systémem, což zahrnuje zejména zhodnocení správnosti extrahovaných dat. V podkapitole 3.1 již bylo zmíněno, že jako vstupní korpus byla využita anglická verze internetové encyklopedie Wikipedie. Využit byl již anotovaný korpus, který byl vytvořen v říjnu roku 2015 a rozprostřen mezi 32 školních serverů. Tato verze obsahuje přes více než 5 miliónů článků. Mohou však nastat jisté odlišnosti s online verzí Wikipedie v podobě nových článků přidaných po říjnu 2015 nebo aktualizací dat ve stávajících článcích. Je proto nutné počítat s mírnými odlišnostmi od online verze. Experimenty byly prováděny pouze na 31 serverech, na zbývajícím serveru nebylo možné spustit skript pro extrakci stejným způsobem jako na ostatních. Korpus i přes tento problém je dostatečně velký pro testování vytvořeného systému.

5.1 Počet nálezů a doba běhu jednotlivých částí systému

Systém se skládá z několika částí, které jsou postupně spouštěny na základě aktuální úlohy systému. Tyto části provádí jednotlivé úkony, které je potřeba provést k dosažení výsledků, čímž jsou extrahované informace o entitách bez vlastní stránky na Wikipedii. Každá část zpracovává odlišné informace, některé pracují na více strojích a jiné zase využívají dotazování na data v databázi. Doba běhu jednotlivých částí je proto velmi odlišná.

5.1.1 Extrakce entit a odkazů

Samotná extrakce entit a odkazů na články je spuštěna na dostupných strojích výpočetního clusteru. Korpus je na těchto serverech rovnoměrně rozdělen podle velikosti jednotlivých částí, které jsou tvořeny samotnými články. Články tedy nejsou rozloženy rovnoměrně podle počtu, ale podle velikosti, kterou zabírají. Tabulka 5.1 zobrazuje počet článků na jednotlivých serverech, dobu zpracování a počet nalezených entit.

Server	Počet článků	Počet nálezů	Doba zpracování (m:s)
1.	160 205	178 383	16:12
2.	176 116	209 229	20:40
3.	177 754	228 598	21:03
4.	171 550	212 614	21:14
5.	114 661	210 780	21:35
6.	49 376	168 626	21:41
7.	164 693	240 521	22:02
8.	186 375	230 256	23:08
9.	201 072	262 947	23:36
10.	200 241	247 840	23:48
11.	201 796	243 406	23:53
12.	186 247	253 283	23:58
13.	207 262	253 412	24:17
14.	203 073	259 904	25:14
15.	89 921	183 420	25:54
16.	238 707	283 244	27:22
17.	169 452	273 944	28:32
18.	64 838	178 790	28:40
19.	99 885	236 809	28:55
20.	188 272	299 002	28:59
21.	137 330	246 843	29:10
22.	182 790	297 644	29:14
23.	79 485	222 608	29:18
24.	103 289	233 680	29:19
25.	115 240	233 680	29:48
26.	153 699	266 912	31:19
27.	137 255	270 938	31:58
28.	197 395	260 841	32:43
29.	156 325	280 929	32:49
30.	117 706	258 323	33:35
31.	108 967	250 533	34:06
Celkem	4 740 977	7 481 416	34:06
Průměr	152 935	241 336	27:22

Tabulka 5.1: Přehled statistik extrakce entit.

Výjimku tvoří extrakce přesměrovaných odkazů. Tato extrakce je totiž spouštěna pouze na jednom serveru, protože vstupní korpus pro tento krok je jediný soubor ve formátu XML. Využitý korpus obsahuje **5 664 283 odkazů**, které jsou přesměrovány na určitý článek. Extrakce těchto odkazů zabrala v průměru **55 minut**.

5.1.2 Kontrola existence článků

Filtrování entit na základě existence článku, který nese název extrahované entity, je krok spouštěný pouze na jednom stroji. Není možné ho spustit na více serverech z důvodu zahlcení serveru, kde je databáze nainstalována. Hledání dat, která jsou uložena v Elastic

databázi, je mnohem rychlejší, než vyhledávání v souboru, ale vzhledem k pouze jednomu přístupu k databázi a rozsahu extrahovaných dat je rychlost ověřování celkem pomalá. Systém dokáže za minutu ověřit v průměru 2 000 extrahovaných entit s úspěšností, která závisí na smysluplnosti vět, ve kterých byla nalezena. Úspěšnost této části je popsána v podkapitole 5.2.2. Kontrola všech nalezených entit (7 481 416) zabrala **4 516 minut a 52 vteřin** (zhruba 75 hodin). Výstupem byl seznam 6 505 443 entit a jejich vět. Bylo odfiltrováno 975 973 entit, což je zhruba 13 % z výstupu extrakce. Doba provedení této části je vysoká, výhodou ovšem je, že nezahltí výpočetní cluster, který může být využíván ostatními uživateli. Tento krok je třeba provést pouze v případě změny metody extrakce entit nebo v případě změny vstupního korpusu. V případě budoucího využití tohoto systému, je uvažováno provedení tohoto kroku pouze při změně vstupního korpusu.

5.1.3 Extrakce informací o entitách

Extrakce informací, které je možné získat z vět získaných při extrakci entit, je opět operace rozdělena na více dostupných serverů. Doba běhu této části je relativně nízká, proto by bylo možné provádět tento krok pouze na jedné stanici, ale pro urychlení práce a nízké výpočetní zátěži tohoto kroku je přijatelnější využít výhody výpočetního clusteru. Navržený extraktor zpracovává pouze věty, ve kterých lze nalézt potřebné části pro tvorbu užitečné informace. Ve větě tedy musí být obsažena samotná entita (podmět) a minimálně jedno slovo přibližující význam entity (předmět). V ideálních případech pracuje také s větným přísudkem, ale není pro účely této části nezbytný. Věty, které obsahují pouze výčet entit, případně se jedná o citaci atp., neobsahují pro systém dostatek informací pro extrakci.

Zpracování výsledků extrakce po ověření existence článků, zabírá průměrně **4 minuty a 39 vteřin**, včetně rozdělení souboru obsahující všechny entity na menší soubory pro jednotlivé servery a následné složení získaných informací do jednoho celku. Ze získaných **6 505 443** entit bylo extrahováno **435 473** informací o různých entitách. Všechny tyto informace bohužel nemají ideální formu a hodnotu pro další využití, ale většinu je možné jednoduše strojově zpracovat. Více o úspěšnosti této části je možné nalézt v podkapitole 5.2.3.

5.2 Úspěšnost jednotlivých částí

Pro určení úspěšnosti jednotlivých částí bylo využito metrik pro hodnocení úspěšnosti systémů pro extrakci informací, které byly popsány v podkapitole 2.4. Každá část systému má svou vlastní úspěšnost. Část pro extrakci přesměrovaných odkazů a extrakce odkazů na články má úspěšnost 100 %, protože se jedná o získání přesných dat, které stačí zkopírovat z textu.

5.2.1 Extrakce entit

Hodnocení extraktoru entit znázorňuje kontingenční tabulka 5.2. Pro vyhodnocení úspěšnosti byl vybrán článek o basketbalovém týmu *Alaska Aces*. Článek byl ručně zpracován a výsledky byly porovnány s daty extrakce[1]. Zpracovávaná verze článku obsahuje 209 entit, které odpovídají třídě *person*, z toho je pouze 42 entit, které nemají anotovaný žádný odkaz, tudíž jsou jasnou volbou pro extrakci a získání informací. Dalších 32 entit má anotovaný odkaz na jinou entitu, než která se nachází v textu. Těchto 32 entit tedy není bráno jako

[1] Extrahovanou stránku a potřebná data pro vyhodnocení úspěšnosti je možné nalézt mezi přílohami.

špatný výsledek v případě, že systém danou entitu nenalezne. Více o tomto problému je popsáno v podkapitole 3.4 a 4.4.3.

Přesnost navrženého extraktoru podle kontingenční tabulky 5.2 vyšla 58,62 %, pokrytí vyšlo 65,38 %.

	Relevantní	Nerelevantní
Získané	34	24
Nezískané	18	133

Tabulka 5.2: Tabulka zachycující úspěšnost extraktoru entit.

Navzdory relativně nízké úspěšnosti extraktoru, většina nejčastěji se vyskytujících tokenů v entitách napovídá tomu, že se jedná o entitu třídy *person*. Prvních několik nejčastěji se vyskytujících tokenů zachycuje tabulka 5.3. Tokeny označující hodnost, případně společenskou vrstvu (Dr., Captain, Lord), byly při extrakci ponechány z důvodu přesnějšího označení entity.

Pořadí	**Token**	**Výskyt**	**Pořadí**	**Token**	**Výskyt**
1.	de	101 306	11.	Robert	30485
2.	John	73 191	12.	U.S.	29 910
3.	Dr.	58 730	13.	Thomas	29 632
4.	William	38 823	14.	Michael	28 678
5.	J.	36 097	15.	George	28 434
6.	James	35 189	16.	Lord	27 302
7.	David	34 414	17.	Captain	27 241
8.	La	33 952	18.	M.	26 081
9.	Mary	33 557	19.	Paul	25 341
10.	A.	30 661	20.	Peter	25 319

Tabulka 5.3: Tabulka zobrazující nejčastěji se vyskytující tokeny v extrahovaných entitách.

5.2.2 Kontrola existence článků

Úspěšnost kontroly existence článků byla provedena na výstupních hodnotách extraktoru. Bylo vybráno několik, na sobě nezávislých článků, ze kterých bylo dohromady získáno 112 entit. V této části již netřeba hledět na úspěšnost extrakce, a proto není nutné řešit správnost extrahovaných entit, ale hlavně správnost vyfiltrování entit s existující stránkou. Kontingenční tabulka 5.4 znázorňuje výsledky získané systémem a výsledky získané ruční kontrolou extrahovaných entit. Přesnost této části systému na testovacích článcích byla vypočítána na 88,24 % a pokrytí 96,15 %.

	Relevantní	Nerelevantní
Získané	75	10
Nezískané	3	22

Tabulka 5.4: Tabulka zachycující úspěšnost kontroly existence článků extrahovaných entit.

5.2.3 Relevantnost získaných dat

Přínos informací, které systém extrahuje z jednotlivých vět nalezené entity, byl rovněž hodnocen pomocí stejného článku jako v podkapitole 5.2.1. Úspěšnost dat je hodnocena na základě počtu informací, které bylo možné z nalezených vět získat o jednotlivých extrahovaných entitách a také smysluplnosti získaných informací. Může nastat takový případ, kdy systém špatně vyhodnotí posloupnost tokenů a jejích návaznost, což vede ke zkreslené, případně nesmyslné informaci. Problém také tvoří entity, jejichž věty neobsahují dostatečné množství relevantních informací. V takových případech není možné podstatné informace získat pouze z dostupné věty, případně získané informace jsou bez okolního kontextu k ničemu. Tento problém se většinou týká entit, které nebyly správně extrahovány.

Při zpracování testovacího článku bylo ručně nalezeno 46 informací o 36 entitách v takovém formátu, který by mohl být nadále zpracováván a nepotřeboval další kontext. Zbylých 13 entit neobsahovalo takové informace, které by bylo možné bez kontextu okolních vět využít. V kontingenční tabulce 5.5 je možné vidět získané hodnoty systémem při extrakci informací z testovacího článku. Informace zachycují hlavní myšlenku věty, která samozřejmě navazuje na danou entitu. Přesnost i pokrytí pro tento článek vychází stejně a to 74,46 %.

	Relevantní	Nerelevantní
Získané	35	12
Nezískané	12	5

Tabulka 5.5: Tabulka zachycující úspěšnost při získávání informací o extrahovaných entitách.

Tato úspěšnost bohužel není u všech extrahovaných entit. Jak již bylo zmíněno dříve, zisk informací o entitě je podmíněn smysluplností věty, ve které byla entita nalezena. Nelze proto přesně říci, zda u některých entit nebyla informace nalezena z důvodu chybovosti algoritmu nebo nedostatečného množství potřebných slovních druhů pro získání informací. Vzhledem k počtu nálezů při extrakci informací (7 % z počtu extrahovaných entit) je nutné zdůraznit, že celková úspěšnost extrakce informací je velmi odlišná od ověřené úspěšnosti na testovacím článku. Data v tabulce 5.6 se vztahují k ručně ověřenému článku.

5.3 Zhodnocení systému jako celku

Navržený systém je sestrojen tak, aby fungoval jako celek bez nutného zásahu člověka. Všechny části jsou plně automatické. Systém automaticky vytváří potřebné adresáře a soubory v umístění, které je možné nastavit před spuštěním. Každý krok systému dostává na vstup jiná data, než krok předchozí, proto není nutné zbytečně zahlcovat úložný prostor soubory, které nadále nebudou využity, a systém je automaticky maže. Automaticky je také řízeno nahrávání dat do databáze, tuto možnost však může uživatel vypnout.

V tabulce 5.6 je možné zhlédnout získané hodnoty z jednotlivých částí. Největší časovou přítěží je kontrola existence článků. Bohužel, se současným rozsahem dat nebylo možné zredukovat čas, potřebný pro ověření všech nálezů, do takové míry, aby se podobal době trvání ostatních částí. Řešením problému dlouhého běhu systému je možnost vypustit ověřování existence článků extrahovaných entit a po extrakci entit přejít rovnou na extrakci informací.

Získané informace o extrahovaných entitách je možné využit pro kontrolu konzistence

#	Název kroku	Čas (m:s)	Počet výstupů	Přesnost	Pokrytí
1.	Extrakce entit	34:06	7 481 416	58,62 %	65,38 %
2.	Extrakce odkazů	55:10	5 664 283	100,00 %	100,00 %
3.	Kontrola existence článků	4 516:52	6 505 443	88,24 %	96,15 %
4.	Extrakce informací	4:39	435 473	74,46 %	74,46 %

Tabulka 5.6: Přehled všech kroků systému.

článků, které obsahují seznamy entit. Pro příklad můžeme využít opět entity ze článku Alaska Aces PBA. Mezi existujícími seznamy na Wikipedii je možné nalézt seznam hráčů basketbalu z PBA[2]. Entity z tohoto seznamu porovnáme s entitami extrahovanými ze článku Alaska Aces PBA, které se řadí mezi hráče tohoto týmu. V případě, že v seznamu chybí některá z extrahovaných entit, je tento seznam doplněn extrahovanou entitou. Mezi nevýhody tohoto kroku patří různorodá syntaxe seznamů ve vstupním korpusu.

5.4 Možnosti budoucího vývoje a využití vytvořeného systému

Předchozí podkapitoly ukazují úspěšnost jednotlivých částí. Ze získaných hodnot je patrné, že ve stávajícím stavu je co vylepšovat. Další vývoj se může ubírat hned několika směry, které vylepší funkčnost jednotlivých částí a přidají další možnosti využití. Jednou takovou cestou může být využití externích nástrojů pro extrakci entit nebo validace nálezu podle již extrahovaných entit, které obsahuje například DBpedia. V principu by entity byly ověřovány pomocí NER systému se staženou znalostní databází existujících jmen. Bylo by možné vyfiltrovat takové nálezy, které obsahují podstatné jméno, které se tváří jako část entity, případně přesněji rozpoznat entity, které se v textu vyskytují hned vedle sebe. Validace nálezů podle již extrahovaných entit by vedla k odfiltrování částí takových entit, které při extrakci přibraly také sousedící entitu. Z jedné nevalidní entity by tak bylo možné získat dvě validní entity.

Další možností je vylepšení anotace korpusu do takové míry, aby každá entita s vlastním článkem obsahovala mezi anotacemi jeho odkaz. Tento krok by mohl být realizován provedení ověřování entit přímo během anotace textu, kde po každé anotované větě by systém zkusil vyhledat odkazy pro všechny entity označené ve větě. Anotace textu by tímto trvala mnohem déle, ale následné zpracování anotovaného korpusu by bylo nejen jednodušší, ale i rychlejší při opakované extrakci bez nutnosti ověřování odkazů.

Při extrakci entit hraje velkou roli koreference. V některých případech extrahovaných vět nelze získat správnou informaci z důvodu nedostatečného zpracování koreference. V takových případech zůstane ve větě pouze zájmeno, na které extraktor informací nemá jak zareagovat kvůli nemožnosti přístupu k předchozí větě. Vylepšené zpracování koreference by umožnilo nahrazovat zájmena rovnou při extrakci entit. Každá věta, která nejdříve obsahovala koreferenci, by nově obsahovala předmět, který by umožnil extrahovat další informaci.

Ověřování existence článků o nalezené entitě lze také vylepšit. Pro vhodnější porovnání je možné zpracovat nejen entity v textu, které nemají odkaz, ale také entity, kterých se týká zpracovávaný článek. Výsledkem by bylo získání přesnější informace nebo informací o článku. Následné porovnání s extrahovanou entitou by mělo hlubší kontext v případě

[2]Philippine Basketball Association - basketbalová liga na Filipínách.

shody jména a bylo by možné správné ověření ve více případech, než nabízí aktuální verze ověřování pouze na základě krátké informace o čem článek referuje.

Vytvořený systém aktuálně extrahuje pouze entity, které se při rozpoznávání řadí do třídy *person*. Extrakci je možné rozšířit na všechny typy entit. Toto vylepšení by však vyžadovalo mnohem přesnější rozeznávání entit, než aktuální systém poskytuje. Společně s použitím postupů rozeznávání entit, které byly zmíněny na začátku této podkapitoly, by mohl mít výsledný extraktor velmi dobrou úspěšnost při extrakci entit pro účel podobné práce.

Získané výsledky vytvořeného systému mohou být využity hned pro několik účelů. Jedním z nich může být využití získaných dat v QA[3] systémech operujícími nad daty Wikipedie. Hledání odpovědí na otázky o entitách, které mají vlastní článek, je možné provádět jednoduše[4]. Problém nastává při dotazu na entitu, která nemá vlastní článek a je nutné ji hledat v celém korpusu. Tento problém by eliminoval přístup takového QA systému do databáze nalezených entit, včetně jejich informací. QA systém by poté mohl uživatele informovat například o místě výskytu entity, počtu stejných entit a hlavně by mohl zodpovědět položenou otázku, pokud by v databázi byla odpověď. Reakce systému by byla mnohem rychlejší, než při prohledávání celého korpusu. Dalším účel pro využití získaných dat může být již zmíněná kontrola konzistence dat například v článcích na Wikipedii, které obsahují jisté druhy seznamů s entitami a následnou opravou chyb, ale také kontrolu konzistence dat získaných při extrakci vztahů mezi entitami.

Mezi poslední možnost vývoje, která bude zmíněna v této práci, patří rozšíření možností pro vstupní korpus. K aktuální možnosti vstupu, do kterého patří Wikipedie, by mohlo být přidáno několik dalších rozsáhlých korpusů, případně rozšířit záběr na extrakci dat z webových stránek. V těchto případech by však bylo nutné ke stávajícímu systému připojit část pro rozbor a anotování textu, který by dokázal zpracovat jak spisovnou formu jazyka, tak i jeho nespisovnou formu, která se může vyskytovat na webových stránkách různých druhů.

[3] QA - Question answering, systémy pro automatické zodpovídání otázek.

[4] Částečně je tato možnost vytvořena ve stávajícím systému, ale není zapojena do provozu.

Kapitola 6

Závěr

Smyslem této práce bylo pochopení a přiblížení metod, které se využívají pro extrakci informací a kontrolu konzistence dat, návrh a implementace systému s funkcionalitou zaměřenou na získávání informací a jejich kontrolu a v neposlední řadě prezentace získaných výsledků a přínosů této práce.

K současnému výsledku práce bylo na začátku nutné pochopit principy extrakce informací, rozpoznávání jmenných entit a anotace textu z dostupné literatury. Tyto znalosti byly poté využity při konstrukci výsledného systému pro extrakci informací o entitách bez vlastního článku na Wikipedii. Cesta k současnému výsledku však nebyla pouze získání znalostí a tvorba jednoduchého systému, ale zahrnovala úpravu a tvorbu vlastních metod, které byly nutné pro správnou funkčnost jednotlivých modulů výsledného systému. Tyto metody bylo potřeba vytvořit na základě získaných znalostí a následně je upravit do nejobecnější podoby s největším možným rozsahem pro zisk potřebných dat, aby bylo možné navržené postupy využít nejen v této práci, ale případě také v budoucích pracích s podobným zaměřením.

Ladění vytvořených metod a systému jako celku byla náročná část práce, která zahrnovala nejen řešení vzniklých problémů, ale také přijímání kompromisů ve funkcionalitě. Bylo nutné postupně procházet výsledky jednotlivých částí, kontrolovat jejich správnost a následně upravovat jednotlivé části systému pro získání lepších výsledků.

Přínos této práce nespočívá pouze v získaných datech, které lze dál využívat, ale hlavně v přínosu nových znalostí o problematice řešené po celém světě, která by jednoho dne mohla být vyvinuta do takové podoby, kdy bude uměle vytvořený systém konkurovat lidským možnostem v oblasti myšlení.

Literatura

[1] Introducing JSON. http://www.json.org/, [Online], [cit. 2016-05-16].

[2] GNU Bash. https://www.gnu.org/software/bash/, 2014, [Online], [cit. 2016-05-16], [rev. 2016-04-29].

[3] Extensible Markup Language (XML). https://www.w3.org/XML/, ©2013–2015, [Online], [cit. 2016-04-25].

[4] ACE. https://www.ldc.upenn.edu/collaborations/past-projects/ace, ©1992-2016, [online]. [cit. 2016-13-04].

[5] ADAMČEK, A.: *Metody extrakce informací*. diplomová práce, FIT VUT v Brně, Brno, 2015.

[6] BANKO, M.; CAFARELLA, M. J.; SODERLAND, S.; aj.: Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, s. 2670–2676.
URL http://dl.acm.org/citation.cfm?id=1625275.1625705

[7] BOLDI, P.; VIGNA, S.: MG4J at TREC 2005. In *The Fourteenth Text Retrieval Conference (TREC 2005) Proceedings*, editace E. M. Voorhees; L. P. Buckland, číslo SP 500-266 in Special Publications, NIST, 2005, http://mg4j.di.unimi.it/.

[8] CAFARELLA, M. J.; DOWNEY, D.; SODERLAND, S.; aj.: KnowItNow: Fast, scalable information extraction from the web. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2005, s. 563–570.

[9] Chacon, S.; Straub, B.: *Pro Git*, ročník 1. New York, NY, USA: Apress, 2009, ISBN 978-1-4302-1833-3.

[10] CVRČEK, V.: Český národní korpus: Příručka pojmů - Korpus. http://wiki.korpus.cz/doku.php/pojmy:korpus, [Online]. [cit. 2016-13-04].

[11] CVRČEK, V.: Český národní korpus: Příručka pojmů - Lemma. https://wiki.korpus.cz/doku.php/pojmy:lemma, [Online], [cit. 2016-13-04].

[12] ETZIONI, O.; CAFARELLA, M.; DOWNEY, D.; aj.: Web-scale Information Extraction in Knowitall: (Preliminary Results). In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, New York, NY, USA: ACM, 2004, ISBN 1-58113-844-X, s. 100–110.
URL http://doi.acm.org/10.1145/988672.988687

[13] GRISHMAN, R.; SUNDHEIM, B.: Message Understanding Conference - 6: A Brief History. In *Proceedings of the 16th conference on Computational linguistics*, ročník 1, PA, USA: Association for Computational Linguistics Stroudsburg, 1996, s. 466–471.

[14] GUODONG, Z.; JIAN, S.; JIE, Z.; aj.: Exploring Various Knowledge in Relation Extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, s. 427–434.
URL http://dx.doi.org/10.3115/1219840.1219893

[15] JI, H.; GRISHMAN, R.: Knowledge base population: successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ročník 1, Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, ISBN 9781932432879, s. 1148–1158, [Online], [cit. 2016-13-04].

[16] JIANG, J.: Information Extraction from Text. In *Mining Text Datas*, ročník 1, Springer US, 2012, ISBN 9781461432234, s. 11–41, [Online], [cit. 2016-23-04].

[17] JORDAN, M. I.; RUMELHART, D. E.: Forward models: Supervised learning with a distal teacher. *Cognitive Science*, ročník 16, 1992: s. 307–354.

[18] LEHMANN, J.; ISELE, R.; JAKOB, M.; aj.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, ročník 6, ©2015: s. 167–195.

[19] MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H.: An Introduction to Information Retrieval. http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf, ©2008, [Online], [rev. 2009-03-01], [cit. 2016-13-04].

[20] MANNING, C. D.; SCHÜTZE, H.: *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999, ISBN 0262133601.

[21] PALÁTOVÁ, H.; GRÁC, M.: Segmentace textu na věty. Hradec Králové: Gaudeamus, 2012, ISBN 978-80-7435-243-0.
URL http://www.ujc.cas.cz/miranda2/export/sitesavcr/data.avcr.cz/humansci/ujc/zakladni-informace/oddeleni/oddeleni-gramatiky/konferencni-sbornik/PalatovaHelena_GracMarek.pdf

[22] REMEŠ, R.: *Programujeme v jazyku Python*. České Budějovice: Jihočeská univerzita v Českých Budějovicích, Ekonomická fakulta, 2008, ISBN 9788073941284.

[23] SANTORINI, B.: Part-of-Speech Tagging Guidelines for the Penn Treebank Project. 1995.

[24] SEDLÁČEK, P.: *Extrakce Informací z textu*. diplomová práce, Masarykova Univerzita v Brně, Brno, 2006.

[25] The MathWorks, I.: Hidden Markov Model. http://www.mathworks.com/help/stats/hidden-markov-models-hmm.html?requestedDomain=www.mathworks.com, ©2016, [Online], [cit. 2016-04-25].

Přílohy

Seznam příloh

Příloha A

Obsah CD

- Zdrojové kódy v adresáři **/src/**
  - Zdrojové kódy vytvořeného systému v adresáři `/src/system/`
  - Zdrojové kódy vytvořených pomocných nástrojů v adresáři `/src/utility/`
- Ukázkové soubory vstupu a jednotlivých výstupů v adresáři `/data/sample/`
- Soubory demonstrující úspěšnost systému v adresáři `/data/succes/`
- Tato technická zpráva ve formátu PDF v adresáři `/thesis/`
- Zdrojové kódy technické zprávy ve formátu LaTeXv adresáři `/thesis/latex/`
- Plakát vytvořené práce v kořenovém adresáři jako `poster.pdf`