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

EXTRAKCE INFORMACÍ Z WIKIPEDIE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE TOMÁŠ KRIŠTOF
AUTHOR

BRNO 2013

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

EXTRAKCE INFORMACÍ Z WIKIPEDIE

INFORMATION EXTRACTION FROM WIKIPEDIA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE TOMÁŠ KRIŠTOF
AUTHOR

VEDOUCÍ PRÁCE Doc. RNDr. PAVEL SMRŽ, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Tato bakalářská práce se zabývá problematikou extrakce informací z nestrukturovaného textu. V první části jsou čtenáři přiblíženy jednotlivé metody, které se v informačních technologiích používají k extrakci informací. Následně je popsán návrh a realizace systému, který získává požadovaná data ze zadaných článků internetového serveru Wikipedie. V poslední části práce je provedena analýza dosažených výsledků.

Abstract

This bachelor's thesis describes the issue of information extraction from unstructured text. The first part contains summary of basic techniques used for information extracting. Thereafter, concept and realization of the system for information extraction from Wikipedia is described. In the last part of thesis, results, coming from experiments, are analysed.

Klíčová slova

Extrakce informací, pravidlový systém, Wikipedie, Python

Keywords

Information extraction, rule-based system, Wikipedia, Python

Citace

Tomáš Krištof: Extrakce informací z Wikipedie, bakalářská práce, Brno, FIT VUT v Brně, 2013

Extrakce informací z Wikipedie

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana docenta Pavla Smrže. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Tomáš Krištof
14. května 2013

Poděkování

Na tomto místě chci poděkovat doc. RNDr. Pavlu Smržovi, Ph.D. za odborné vedení práce a za veškerou pomoc, kterou mi při jejím řešení poskytl.

© Tomáš Krištof, 2013.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **2**

2 Extrakce informací **3**
2.1 Vymezení pojmů . . . 3
2.2 Manuální metody extrakce entit . . . 5
2.3 Metody extrakce entit založené na pravidlech . . . 5
2.3.1 Tvar pravidla . . . 5
2.3.2 Druhy pravidel . . . 5
2.3.3 Kolekce pravidel . . . 7
2.3.4 Učící se algoritmy . . . 7
2.4 Statistické metody extrakce entit . . . 8
2.4.1 Modely na úrovni tokenů . . . 8
2.4.2 Modely na úrovni segmentů . . . 10
2.4.3 Modely na úrovni gramatiky . . . 10
2.5 Extrakce vztahů . . . 11
2.6 Užití v praxi . . . 12

3 Návrh a realizace systému pro extrakci informací **14**
3.1 Návrh systému . . . 14
3.2 Implementace . . . 15
3.2.1 Použité nástroje . . . 16
3.2.2 Ovládání aplikace . . . 16
3.2.3 Sada pravidel . . . 17
3.2.4 Náhrada ukazovacích zájmen . . . 18
3.2.5 Odhad jména . . . 18
3.2.6 Odhad data . . . 19

4 Experimentální výsledky **20**
4.1 Návrh testovací sady . . . 20
4.2 Analýza dosažených výsledků . . . 20
4.2.1 Správnost úprav zdrojových vět . . . 22

5 Závěr **23**

A Obsah DVD **25**

Kapitola 1

Úvod

V posledních letech se enormně zvýšil objem informací obsažených v dokumentech, které jsou dostupné převážně v elektronické podobě. Tyto zdroje nestrukturovaných informací výrazně převyšují množství informací dostupných ve formě strukturované (databáze atd.). Aby tyto informace mohli být automaticky zpracovány, musejí se nejprve převést do vhodnější podoby. A právě to je hlavním úkolem IE[1].

S prvními zmínkami o extrakci informací, jakožto vědním oborem, se můžeme setkat v záznamech z MUC[2] z konce 80. let minulého století. Konference byly sponzorovány americkou vládou za účelem rozvoje této vědní disciplíny. Zúčastňovaly se jich řady účastníků z různých prostředí, především z akademického a výzkumného. Účelem konferencí bylo získat kvantitativní výsledky systémů pro extrakci informací, které vytvořily jednotlivé skupiny účastníků. Následně bylo prokázáno, že plně automatizované systémy pro extrakci, vytvořené na základě nejmodernějších technologií, mohou být pro některé vybrané úlohy stejně úspěšné jako při zpracování lidským expertem [1].

Metody užívané k extrakci informací prošly znatelným vývojem. První systémy byly založeny na pravidlech, která bylo nutné ručně formulovat. Jelikož ale byla tvorba pravidel příliš časově náročná a výkonnost systému byla značně ovlivněna schopnostmi autora pravidel, začaly se vyvíjet algoritmy pro automatické učení pravidel ze vzorků dat. Pravidla ale nebyla dostatečně pružná a nedokázala spolehlivě pokrýt vysoké množství různorodých nestrukturovaných zdrojů dat. Přišla éra statistického učení. Vývoj se rozdělil do dvou větví – první část metod byla založena na Skrytých Markovských modelech, druhá naopak na principu maximální entropie. Nejnovější metody zkoumají komplexní strukturu dokumentu a využívají poznatky z oblasti konstrukční gramatiky.

Tato práce si předkládá za cíl poskytnout čtenáři alespoň částečný náhled do problematiky extrakce informací. Jsou zde vysvětleny základní metody extrakce, následně je popsán návrh a realizace systému pro extrakci informací z Wikipedie. Závěrečná část je zaměřena na analýzu dosažených výsledků.

[1] Z angl. information extraction - extrakce informací
[2] Message Understanding Conferences

Kapitola 2

Extrakce informací

Pod pojmem extrakce informací si můžeme představit takový proces, který z velkého množství nestrukturovaného textu získává relevantní informace, odpovídající uživatelovu zájmu. Konečný výstup procesu může být různý. V každém případě by ale mělo platit, že získaná data lze transformovat tak, aby je bylo možné uložit buď do databází, nebo do šablon. Odtut jsou poté data přístupná k dalšímu zpracování.

Extrakce informací je často zaměňována s disciplínou nazvanou shromažďování informací (IR[1]). Ovšem cílem IR je vyhledávat relevantní dokumenty napříč rozsáhlými kolekcemi dat. Uživatel poté musí sám prohedat získanou množinu dokumentů, aby nalezl požadované informace.

Na proces extrakce informací můžeme pohlížet jako na soustavu zřetězených dílčích úkolů, které jsou na sobě vzájemně závislé. Každá část procesu extrakce informací na svém vstupu očekává výstup z předchozího modulu systému. Vstupem pro první modul je samotný nestrukturovaný zdrojový text a výstupem posledního modulu jsou strukturovaná data. Obecná architektura systému je znázorněna na obrázku[2] 2.1.

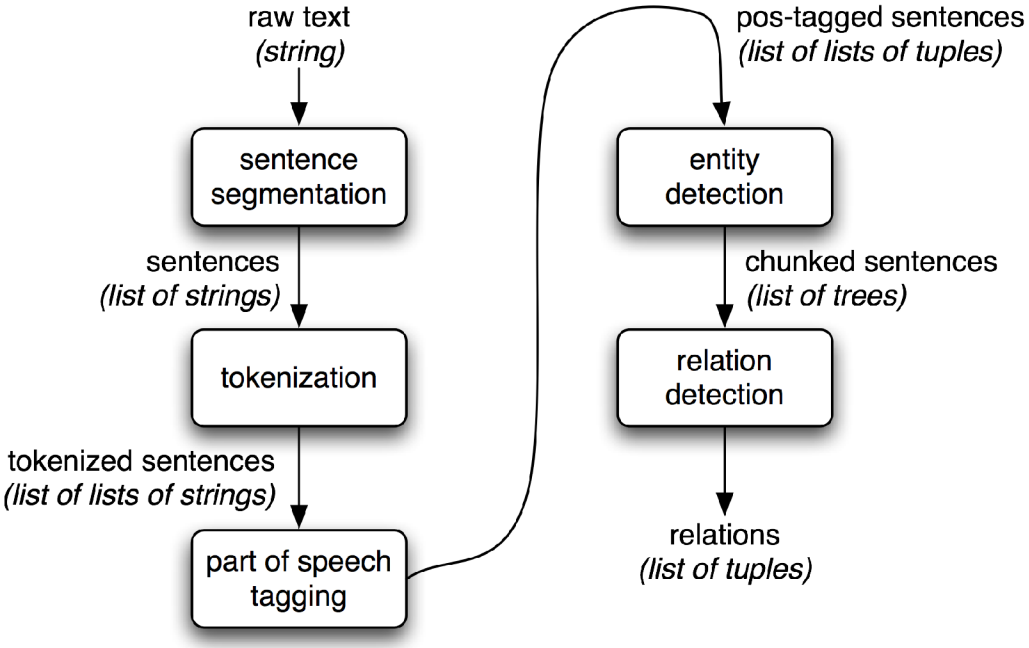
2.1 Vymezení pojmů

Pro správně porozumnění dané látky je nutné vysvětlit význam určitých pojmů, se kterými se čtenář může v této práci dále setkat:

- **strukturovaný text** – Textová informace uložená ve předem definovaném a neměnném formátu. Se strukturovaným textem se můžeme setkat například u databází.
- **token** – Nepřerušovaný řetězec znaků, slovo. Jako token může být označena interpunkce, případně i znak nového řádku.
- **entita** – Slovní spojení, skládající se typicky z jednoho či dvou tokenů. Nejvyhledávanější entity jsou tzv. *pojmenované entity*
- **pojmenovaná entita** – Typicky podstatná jména a vlastní jména, která v textu jasně identifikují určité objekty, jako jsou například osoby, geografické lokace, organizace, peněžní či časové údaje. Znázornění pojmenovaných entit ve větě je možné vidět na obr. 2.2.

[1] Information retrieval

[2] Obrázek byl převzat ze serveru nltk, http://nltk.org/



Obrázek 2.1: Obecná architektura systému pro extrakci informací

```
<PERSON>Bill Gates</PERSON> founded <ORGANIZATION>Microsoft</ORGANIZATION> in
<DATE>1975</DATE>.
```

Obrázek 2.2: Vyznačení pojmenovaných entit ve větě

- **relace** – Udává vztah, který je definován mezi dvěmi entitami. Například mezi osobou a organizací bychom mohli najít vztah *je zaměstnancem*.
- **klasifikátor** – Sada pravidel, umožňující zařadit zpracovávaný objekt do některé z tříd, pokud splňuje její podmínky.
- **korpus** – Soubor textů, opatřených metaznaky, které uchovávají informace jak o samotném textu, tak i o jednotlivých slovech. Typickým druhem informací obsažených v korpusu je například příslušnost daného slova k jednotlivým kategoriím slovních druhů.
- **ortografie** – Konvenční hláskovací systém jazyka.
- **pokrytí** – Poměr správně extrahovaných informací. Pokrytí je definováno následovně:

$$pokrytí = \frac{počet\ správných\ odpovědí}{počet\ všech\ možných\ odpovědí}$$

- **přesnost** – Poměr získanách informací, které byly systémem extrahovány ve správném formátu a mají požadované vlastnosti. Přesnost je definována následovně:

$$přesnost = \frac{počet\ správných\ odpovědí}{počet\ produkovaných\ odpovědí}$$

2.2 Manuální metody extrakce entit

Systémy pro extrakci informací, které využívají manuálních metod, potřebují ke své práci lidského experta, který definuje regulární výrazy a pravidla pro provádění extrakce. Tato osoba musí mít dostatek zkušeností z oblasti extrakce informací, programování a lingvistiky, aby mohla vytvářet dostatečně robustní pravidla pro extrakci [6].

Přímočarým přístupem je vytvořit sadu pravidel ve formě regulárních výrazů, které budou určitým způsobem detekovat rozdílné kategorie entit. Regulární výrazy mohou využívat sady prefixů a sufixů, které napomáhají k identifikaci entit. U detekce jmen by to mohl být například výčet titulů.

Systémy s manuálními metodami často využívají rozsáhlé seznamy osob, lokací, organizací a dalších typů pojmenovaných entit. Mnoho takových seznamů je v dnešní době dostupných prostřednictvím internetu a mohou být poměrně užitečné. Pro některé druhy entit je jejich použití prakticky nezbytné, jelikož neexistují kvalitní kontextové vzory na jejich identifikaci. Rozsáhlé seznamy mohou obsahovat položky, která jsou sice vlastními jmény, v běžných větách ale častěji zastupují roli obecných výrazů (např. město *Normal* ve státě Illinois) – jejich počet by v seznamech měl být co nejmenší.

Vytvořením sady regulárních výrazů je možné vytvořit poměrně efektivní a rychlý systém pro extrakci entit, je k tomu ovšem potřeba poměrně mnoho zkušeností. Tyto systémy jsou vytvářeny na míru jednotlivým úlohám a jejich velikou nevýhodou je, že jsou upnuty na určitý druh vstupních dat a nelze od nich předpokládat široké pokrytí entit ve smyslu jednotlivých typů.

2.3 Metody extrakce entit založené na pravidlech

Mnoho dnešních úloh pro extrakci informací lze řešit pomocí sbírky pravidel. Tyto pravidla mohou být buď zadána ručně, nebo být naučena ze vhodných příkladů. První systémy pro extraki informací byly založené právě na pravidlech. Tyto systémy jsou také rychlejší, než statistické systémy a mnohem snadněji se optimalizují. Tato podkapitola byla volně převzata z [6].

Typický pravidlový systém pro extrakci informací se skládá ze dvou částí: sady pravidel a politiky, kterád se uplatňuje při možnosti aplikace více pravidel současně.

2.3.1 Tvar pravidla

Základní pravidla se zapisují ve tvaru *Kontextový vzor* → *Akce*. Kontextový vzor se skládá z jednoho či více označených vzorů, zachycujících různé vlastnosti entit a kontext, ve kterém se ve větě vyskytují. Označený vzor je většinou tvořen regulárním výrazem, definovaným nad množinou tokenů.

Akce pravidla se používá k označení operace, která bude následovat. Může jít o přiřazení značky entity k sekvenci tokenů či vložení otevíracího či uzavíracího tagu entity na danou pozici.

2.3.2 Druhy pravidel

Jednotlivých pravidel je nepřeberné množství, obecně je ale můžeme rozdělit do tří kategorií – pravidla pro určení jednoduchých entit, pravidla pro určení složených entit a pravidla pro označení hranic entit.

Rule: TheGazOrganization
Priority: 50
// Matches "The <in list of company names>"
({Part of speech = DT | Part of speech = RB} {DictionaryLookup = organization})
→ Organization

Rule: LocOrganization
Priority: 50
// Matches "London Police"
({DictionaryLookup = location | DictionaryLookup = country} {DictionaryLookup = organization} {DictionaryLookup = organization}?) → Organization

Rule: INOrgXandY
Priority: 200
// Matches "in Bradford & Bingley", or "in Bradford & Bingley Ltd"
({Token string = "in"})
({Part of speech = NNP}+ {Token string = "&"} {Orthography type = upperInitial}+ {DictionaryLookup = organization end}?):orgName → Organization=:orgName

Rule: OrgDept
Priority: 25
// Matches "Department of Pure Mathematics and Physics"
({Token.string = "Department"} {Token.string = "of"} {Orthography type = upperInitial}+ ({Token.string = "and"} {Orthography type = upperInitial}+)?) → Organization

Obrázek 2.3: Soustava pravidel pro určení jména organizace v systému Gate

Pravidla pro určení jednoduchých entit se skládají ze tří typů vzorů: vzoru pro vyhledávání tokenů odpovídajících příslušným entitám a volitelných vzorů pro zachycení kontextu před začátkem či po konci entity. Jednoduché pravidlo pro zachycení letopočtu by mohlo být zapsáno ve tvaru:

```
({Řetězec="před rokem"|Řetězec="v roce"|Řetězec="po roce"})({Ortografický
typ=Číslo}):rok => Letopočet=:rok
```

V tomto pravidle se nacházejí dva vzory: první vzor zachycuje kontext výskytu letopočtu a druhý vzor zachycuje vlastnosti samotného tokenu, který tvoří letopočet.

Pravidla pro určení jednoduchých entit jsou používána například v systému pro extrakcí informací Gate[3]. Na obrázku 2.3.2 je možné vidět soustavu pravidel, určených k identifikaci jména organizace [6].

Pravidla pro určení složených entit sestávají z vícenásobných regulárních výrazů, kde každý regulární výraz reprezentuje odlišnou entitu. Detekce celé složené entity tedy proběhne najednou. Tyto pravidla byly hojně používána v systému WHISK pro těžbu strukturovaných záznamů, jako jsou např. lékařské záznamy a novinové inzeráty.

Pravidla pro označení hranic entity jsou používána pro jejich efektivnost při identifikaci víceslovných entit, jako mohou být například jména knih. V takových případech je vhodnější použít pravidla na označení hranic začátku a konce entity. Sekvence tokenů mezi těmito

[3] http://gate.ac.uk/

hranicemi je poté označena jako entita. Mnoho úspěšných systémů pro extrakci informací, jako STALKER či Rapier, používá právě tyto pravidla.

2.3.3 Kolekce pravidel

Většina pravidlových systémů se skládá z rozsáhlých kolekcí pravidel. Každá aplikace jednotlivých pravidel označí rozsah textu, který bude dále považován za entitu. Může se stávat, že vymezený rozsah rozdílných pravidel se vzájemně překrývá, což může vést ke konfliktům. Je tedy nutné se zabývat organizací pravidel a pořadím, v jakém jsou aplikovány, stejně jako řešením případných konfliktů.

Jedním řešením je zacházet s kolekcí pravidel jako s neuspořádanou množinou – každé pravidlo se aplikuje nezávisle na ostatních. Problém může nastat, pokud dva či více překrývajících se úryvků textu jsou označeny dvěma či více pravidly. Tyto konflikty poté řeší politiky systému, které jsou vytvořeny k právě těmto účelům. Známou politikou je preferování pravidel, která označí větší rozsah textu.

Další často používanou strategií je přiřadit všem pravidlům prioritu a tím definovat pořadí, v jakém se budou aplikovat. Při vícenásobné shodě pravidel je poté použito to pravidlo, které má vyšší prioritou. Výhoda této organizace je, že lze vytvářet pravidla, která budou vyhledávat text již zpracovaný předchozími pravidly.

2.3.4 Učící se algoritmy

Každý systém pro extrakci entit je závislý na obsáhlém souboru jemně laděných pravidel. Pravidla mohou být vytvořena ručně expertem z oboru, nebo mohou být naučena z označených příkladů entit v nestrukturovaném textu. Mějme množinu dokumentů $D = x_1, x_2, \ldots, x_n$, kde jsou všechny výskyty požadovaných entit správně označeny. Taková množina se nazývá *trénovací množina*. Našim cílem je se naučit množinu pravidel $R_1, R_2, \ldots, R_k$. K dosažení co nejvyšší obecnosti, algoritmy pro učení pravidel se snaží definovat co nejmenší množinu pravidel, která pokryje maximální počet trénovaných případů s co nejvyšší přesností. Nalezení optimální množiny parvidel je ale velmi složité. Používá se k tomu matematický algoritmus *Greedy hill climbing*, jehož pseudokód můžeme popsat následovně:

1. R = množina pravidel, na počátku prázdná
2. Dokud existuje entita $x \in D$, pro kterou neexistuje pravidlo v R:
   (a) Vytvoř nové pravidla pro pokrytí x
   (b) Přidej nově vzniklé pravidla do množiny R
3. Odstraň přebytečná pravidla

Hlavní výzvou je poté zjistit, jak vytvořit nové pravidla, která mají co nejvyšší celkové pokrytí, přesnost a nejsou již zastoupeny ve stávající množině pravidel. Na toto téma je vytvořeno mnoho strategií a heuristik, které vesměs spadají do dvou hlavních tříd: formace pravidel metodou zdola-nahoru a formace pravidel metodou shora-dolů.

Při učení se pravidel metodou zdola-nahoru jsou zpočátku definována pravidla velmi specifická, pokrývající pouze specifické výskyty entit. Takové pravidlo má minimální pokrytí, ale 100% přesnost a je zaručeno, že není přebytečné, jelikož vzešlo z entity, která nebyla dosud pokryta žádným pravidlem z naučené množiny. Toto pravidlo je následně zobecněno, čímž se zvýší pokrytí na úkor přesnosti. Jedním z úspěšných algoritmů z této

kategorie je algoritmus $(LP)^2$, vyvinutý speciálně pro učení se pravidel pro extrakci entit. Tento algoritmus používá pravidla pro označení hranic entit, jejich akční část tedy spočívá ve vložení patřičné otevírací či uzavírací značky na příslušné pozice v textu. Pravidla jsou učena nezávisle pro každou akci. Pro každý druh značky Z se opakovaně prochází následující kroky, dokud v trénovací množině dat nejsou nepokryté případy pro daný druh značky:

1. Vytvoření pravidla pro pokrytí daného výskytu
2. Zobecnění vytvořeného pravidla
3. Odstranení všech výskytů entit, které jsou pokryty nově vytvořeným pravidlem

Při metodě shora-dolů počáteční pravidlo dosahuje 100% pokrytí, ale nízké přesnosti. Počáteční pravidlo je poté upraveno tak, aby se získala sada pravidel s vyšší přesností. Každý krok úprav zvyšuje pokrytí výchozího pravidla. Algoritmus $(LP)^2$ má i svoji variantu, využívající metodu učení shora-dolů. Další algoritmus, které využívá této metodyje například FOIL[4].

2.4 Statistické metody extrakce entit

Statistické metody extrakce entit se zabývají převážně dekompozicí nestrukturovaného textu a jeho značením. Nejběžnější formou dekompozice je rozdělení textu na sekvenci tokenů, čehož je dosáhnuto rozštěpením nestrukturovaného textu dle předem definované množiny oddělovačů (bílé znaky či interpunkce). Následně je každý token přiřazen k entitě nebo části entity. Jakmile jsou všechny tokeny řádně označeny, entity jsou tvořeny po sobě jdoucími tokeny se stejnou značkou. Toto jsou tzv. metody na úrovni tokenů.

Další formou dekompozice je rozdělení textu do bloků slov. Při značení textu se značky přiřazují celým blokům (segmentům) slov, oproti jednotlivým tokenům. Tyto metody jsou úspěšné pro přirozeně tvarované věty v přirozeném jazyce. V opačných případech tato metoda selhává (adresy nebo klasifikované inzeráty). Tyto modely jsou založeny na metodách na úrovni segmentů.

Obě ze zmíněných metod dekompozice zdrojového textu dokumentu neumí těžit z vlastností a struktury samotného dokumentu. V určitých případech jsou tedy úspěšnější metody na úrovni gramatiky, využívající bezkontextové gramatiky vedené produkčními pravidly. Tato podkapitola byla volně převzata z [6].

2.4.1 Modely na úrovni tokenů

Modely na úrovni tokenů převládají nad ostatními statistickými metodami. Nestrukturovaný text je rozdělen na posloupnost tokenů a hlavním úkolem je přiřadit každému tokenu značku, reprezentující příslušnou entitu. Mějme množinu tokenů $\mathtt{x} = x_1, x_2, \ldots, x_n$, kde n je počet tokenů ve větě. Každému tokenu x_i je následně přiřazena značka z množiny Υ. Tím je získána posloupnost tagů $\mathtt{y} = y_1, y_2, \ldots, y_n$. Množina Υ se skládá ze značek entit, které chceme extrahovat, a speciální značky *Další*, která je přiřazena všem ostatním tokenům, jež nespadají pod pozornost uživatele. Chtěli-li bychom ve zdrojovém textu vyznačit adresy, množina Υ by byla tvořena následovně: $\Upsilon = \{$*Číslo popisné, Ulice, Město, Poštovní směrovací číslo, Stát, Další*$\}$. Na obrázek č. 2.4 najdeme příklad, jak je věta rozdělena na posloupnost tokenů `x` a následně je získána množina značek `y`.

[4] First Order Induction Learner

Žádost zašlete na adresu Jarní 18, Hodonín, 69504.

i	1	2	3	4	5	6	7	8	9	10
x	Žádost	zašlete	na	adresu	Jarní	18	,	Hodonín	,	69504
y	*Další*	*Další*	*Další*	*Další*	*Ulice*	*Č. p.*	*Další*	*Město*	*Další*	*PSČ*

Obrázek 2.4: Dekompozice věty na sekvenci tokenů a jejich následné označení

Jelikož entity mohou sestávat z více tokenů, je možné každé takové části entity opatřit značkami *Entity Begin*, *Entity Continue* či *Entity End*. Toto se nazývá jako tzv. BCEO[5] značení. Ekvivalentem je též BIO[6] značení.

Typickým úkolem extrakce tohoto druhu je zachytit vlastnosti a kontext jednotlivých tokenů pomocí různorodého souboru záchytných bodů a vodítek. K tomu může být použita mapovací funkce $f : (\mathtt{x}, y, i) \mapsto R$, která jako argumenty přebírá množinu tokenů $\mathtt{x}$, tvořící zpracovávanou větu, index i, udávající pozici požadovaného tokenu a značku y, kterou chceme přiřadit tokenu x_i. Návratová hodnota poté udává vlastnosti i-tého tokenu a také tokenů v jeho okolí, je-li jim přiřazena značka y. Získané vlastnosti můžeme rozdělit do tří kategorií:

- **Vlastnosti slova**: Tvar samotného slova je často sám o sobě silným vodítkem k určení značky, která mu má být přiřazena. Příkladem nechť je mapovací funkce

  $$f_1(\mathtt{x}, y, i) = [x_i \textit{ shodné s „Picasso“}] \cdot [y = \textit{Osoba}],$$

  která přiřadí značku *Osoba* všem tokenům, které jsou shodne s řetězcem *Picasso*.

- **Ortografické vlastnosti**: Mnoho cenných informací může být odvozeno z různých ortografických vlastností slova, jako je například kapitalizace nebo přítomnost speciálních znaků. Příkladem funkce, využívající ortografické vlastnosti tokenu, může být funkce

  $$f_2(\mathtt{x}, y, i) = [x_i x_{i+1} \textit{ obsahují } \text{Spec_znak} + \text{Vlastní_jméno}] \cdot [y = \textit{Osoba}],$$

  jejíž úlohou je přiřadit značku *Osoba* takové dvojici tokenů, kde první z nich obsahuje speciální znak jako je interpunkční znaménko, a druhý je vlastním jménem. Tyto vlastnosti by byly nalezeny například u páru tokenů *Dr. Novák*.

- **Vlastnosti vyhledávacích slovníků**: Jak bylo zmíněno v sekci 2.2 na straně 5, často je možné využít databázi již zpracovaných entit, dostupnou v čase zpracování. vyhledávat shody zpracovávaných tokenů v těchto slovnících je mocnou pomůckou při extrakci entit. Funkce

  $$f_3(\mathtt{x}, y, i) = [x_i \textit{ obsaženo v } \text{Seznam_jmen}] \cdot [y = \textit{Osoba}]$$

  by za osobu označila všechny tokeny, které by byly obsaženy v příslušném seznamu jmen.

Metoda CRF[7] poskytuje účinný a flexibilní mechanismus pro predikci značek tokenům, s ohledem na jejich okolí a kontext. CRF je typ bezsměrového grafického pravděpodobnostního modelu, kde je každý uzel popsán náhodnou proměnnou [4].

[5] Begin, Continue, End, Other
[6] Begin, Inside, Other
[7] Conditional Random Fields

2.4.2 Modely na úrovni segmentů

Výstupem těchto modelů je posloupnost segmentů, kde každý segment definuje celou entitu. Oproti tomu modely na úrovni tokenů, popsané v kapitole 2.4.1, označovaly jednotlivé tokeny, přičemž entita mohla být tvořena více tokeny. Formálněji řečeno, segmentace $\mathbf{s}$ vstupu o délce n je posloupnost segmentů $s_1, s_2, \ldots, s_p$, takových, kde první segment začíná na pozici 1 a poslední segment končí na pozici n. Pro dva po sobě jdoucí segmenty s_j, s_{j+1} platí, že segment s_{j+1} začíná ihned za ukončením segmentu s_j. Každý segment s_j je tvořen počáteční pozicí l_j, koncovou pozicí u_j a značkou $y_j \in \Upsilon$. Segmentaci vzorové věty lze vidět na obrázku č. 2.5. Zpracovávané vlastnosti jsou definovány nad segmenty, množinou tokenů tvořící entity. To umožňuje zkoumat jemnější rysy oproti tokenům, jelikož segmenty nyní sdružují souhrn vlastností všech všech tokenů, které je tvoří.

Pablo Picasso se narodil v Málaze, Španělsku

i	1	2	3	4	5	6	7	8
$\mathbf{x}$	Pablo	Picasso	se	narodil	v	Málaze	,	Španělsku
l_j, u_j, y_j	*1, 2, Osoba*		*3, 5, Další*			*6, 8, Lokace*		

Obrázek 2.5: Dekompozice věty na segmenty a jejich následné označení

Podobně jako u vlastností tokenů, označení segmentů je závislé na značce předchozího segmentu a vlastnostech aktuálního segmentu. Vlastnost segmentu bude mapována funkcí ve tvaru $f(\mathbf{x}, y_j, y_{j-1}, l_j, u_j)$. Vlastnosti segmentů lze rozdělit do dvou kategorií:

- **Podobnost k entitě v databázi**: Většina entit je tvořena množinou tokenů. Oproti modelům na úrovni tokenů, kde není možné definovat vlastnost, která měří podobnost entity k celé entitě obsažené v patřičné databázi, modely na úrovni segmentů to umožňují. Objeví-li se navíc jeden z tokenů tvořících segment například v databázi organizací, je velice pravděpodobné, že celý segment bude tvořit název organizace a můžeme ho tedy opatřit příslušnou značkou. Tento případ by mapovala funkce

  $$f_1(\mathbf{x}, y_j, y_{j-1}, l, u) = [x_l || \ldots || x_u \ \textit{obsaženo v} \ \text{Seznam_organizací}] \cdot [y_j = \textit{Organizace}],$$

  která by testovala výskyt jednoho z tokenů na pozici 5 až 7 v seznamu organizací.

- **Délka segmentu**: Další užitečnou vlastností segmentu je jeho délka, jelikož některé druhy entit ji mají relativně pevně stanovenou. Jako příklad nechť nám poslouží například seznam titulů uváděných za jménem.

2.4.3 Modely na úrovni gramatiky

Některé extrakční systémy požadují kvalitnější interpretaci struktury zdroje, než jakých lze dosáhnout segmentací, případně tokenizací textu. Modely na úrovni gramatiky používají sadu přepisových pravidel, podobně jako bezkontextové gramatiky programovacích jazyků. Pomocí těchto pravidel je vyjádřena globální struktura entity. Přepisy jsou definovány nad terminály, v tomto případě tokeny, a neterminály, zahrnující značky entit a další dotatečné metaznaky. Výstupem tohoto procesu je derivační strom, respektive množina derivačních stromů, jelikož pro sekvenci tokenů existuje více validních derivačních stromů. Každý výstupní strom je ohodnocen body, které se rozloží mezi jednotlivé přepisy. Dřívější metody k výpočtu ohodnocení byly založeny na generativních modelech, později se začaly používat

diskriminační modely. I přesto, že tyto metody jsou velice účinné, v praxi se s nimi moc nesetkáme, jelikož hledání optimálního derivačního stromu pro danou sekvenci tokenů je příliš náročnou operací.

2.5 Extrakce vztahů

Detekce a klasifikace relací jako dílčí úkol extrakce informací byla poprvé představena na MUC č. 7 v roce 1998. Zde byly pokryty tři druhy relací spadající k entitám typu organizace: relace *je_produktem, je_zaměstnancem* a *nachází_se_v*. Od roku 2002, součástí ACE[8], byly relace podrobeny rozsáhlým studiím [2].

Relace je definována jako predikát nad dvěma argumenty, kde argumenty reprezentují objekty, pojmy nebo osoby, a vztah popisuje druh asociace či interakce mezi objekty, které jsou zastoupeny již zmíněnými argumenty [3]. Zmínka o relaci je úsek textu, vyjadřující spojení R mezi dvěmi zmínkami o entitách m_1 a m_2. Jestliže m_1 se odkazuje k entitě e_1 a m_2 k entitě e_2, pak se R zmiňuje o relaci mezi entitami e_1 a e_2. Najít v textu úseky textu, ve kterých je zmínka o hledané relaci, je ale složité.

Mnoho konkrétních výskytů relací může být identifikováno podle typů zúčastněných entit a druhu slov, které se mezi entitami nacházejí. Jedním z přístupů k extrakci entit je tedy manuální tvorba vzorů, podle kterých se relace budou vyhledávat. Chtěli-li bychom zjistit, jaký je vztah mezi osobou a místem, mohli bychom vytvořit vzory, které by zachycovaly situace:

- *osoba* žije v *místě*
- *osoba* žila v *místě*
- *osoba* bydlí v *místě*
- *osoba* přebývá na *místě*
- *osoba* vlastní dům v *místě*
- ...

Jak je zřejmé, vytvořit komplexní sadu pravidel pro zachycení vztahu mezi osobou a místem by bylo velice časově náročné, nemluvě o velikosti sady těchto pravidel, jelikož by se musela vytvořit varianta pro veškeré syntaktické, morfologické a lexikální variace. Lepšího pokrytí dosáhneme za použití určitých syntaktických zobecnění. Ve vzoru bude například zmíněn pouze základní tvar slovesa, reprezentujícího relaci, tak, aby pravidlo zachytilo i ostatní, odvozené tvary. Pro anglický jazyk by pak pravidlo

$$\textit{person}\ \langle \text{verb* base=move} \rangle\ \textit{location}$$

vyhledalo všechny tvary, které se pojí se slovesem *move* – *moved*, *had moved*, *will move*, atd.

Namísto vylepšování a vytváření stále nových pravidel, systém pro extrakci relací může být postaven na základě anotovaného korpusu. K tomu je potřeba korpus, který byl anotován jak pro entity, tak pro relace. K manipulaci s jediným druhem relace se klasifikátor vycvičí tak, aby vždy ve stejné větě klasifikoval každý pár zmíněných entit, i pokud mezi

[8] Automatic Content Extraction, http://projects.ldc.upenn.edu/ace/

nimi není požadovaný druh relace. K manipulaci s n druhy relace je potřeba vytvořit $n+1$-třídní klasifikátor. V praxi je poté nutné nejdříve ve zdrojovém textu označit entity a až následně vyhledávat relace mezi právě dvěmi entitami. U každého páru entit je navíc nutné rozhodnout, zda spolu jednotlivé entity vůbec souvisejí (což zajišťuje binární klasifikace) a pokud ano, tak jaký je mezi nimi vztah. Při detekci druhu relace se využívá rozličných vlastností entit, jmenujme například druh entit (lokace, organizace), vzdálenost entit, závislost entit, výskyt určité sekvence slov mezi entitami či výskyt jednotlivých slov mezi entitami.

Tento přístup ale není úplně vždy užitečný. Někdy je jednodušší pokusit se v korpusu nalézt takový výskyt, který je určitým způsobem podobný námi detekované relaci a přiřadit mu takové označení, které nese originál v korpusu. K tomu můžeme vytvořit tzv. jádrovou funkci[9], která ohodnotí podobnost mezi dvěmi objekty [5].

Samotná příprava anotovaného korpusu pro extrakci relací je z principu mnohem náročnější, než označování entit. Zmíněná entita musí být nejdříve označena, případně zkontrolována a opravena, byla-li označena automaticky, a až poté musí být anotována řada rozličných relací. Nastává tedy otázka, zda-li by systém nešel vycvičit i bez závislosti na korpusu.

Zpočátku se vezme skupina párů entit, nesoucí specifický vztah. Chtěli-li bychom vyhledávat například relace mezi autorem a dílem, počáteční data by byla tvořena entitami *[The Lost Symbol, Dan Brown], [Animal Farm, George Orwell]*. Následně je v textu vyhledáván kontext, ve kterém se v něm tyto páry entit vyskytují – v našem příkladu bychom mohli získat například relace *kniha od, napsal* atd. Na základě tohoto kontextu by byly vyhledány i nové entity a celý proces by se opakoval. Tento přístup je založen na předpokladu, že entity jsou spojeny jedinenčým vztahem. Pokud mezi více páry entit existuje také více druhů ne zcela jedinečných asociací, bude docházet k postupnému rozšiřování druhů vzorových entit a tím pádem i k učení se nesprávných druhů relací. Tento jev se nazývá sémantický drift [2].

2.6 Užití v praxi

Užití systémů pro extrakce informací je užitečné napříč širokým spektrem aplikací, jak už v komerčních systémech, tak i ve volně dostupných a každodenně používaných množstvím uživatelů.

Jedním z nejčastějších typů aplikací, kde se s těmito systémy můžeme setkat, jsou **webové aplikace**. Zde se s nimi můžeme setkat ve velké míře například u *citačních databází*, které byly vytvořeny za základě propracovaných úkonů vedoucích k naplnění potřebných šablon údaji nezbytnými ke korektnímu citování. Mezi tyto systémy patří například Google Scholar[10]. Tvorba takových databází vyžaduje extrakci informací na mnoha rozdílných úrovních, jmenujme například vyhledání umístění patřičných dokumentů, extrakci individuálních zápisků z HTML stránek, získávání názvů děl, jejich autorů a referencí z dokumentů ve formátu PDF.

V poslední době se také můžeme s *reklamami*, které jsou na internetových stránkách umístěny v těsné blízkosti textu, který dané téma, na které je reklama zaměřena, zmiňuje. I toho je dosaženo extrakcí informací, které v tomto případě má za úkol zjistit, jaký druh reklamy k článku umístit.

[9] Kernel function

[10] http://www.scholar.google.com

Další velkou a majoritní výzvou je *strukturované vyhledávání*. Vyhledávání pomocí klíčových slov nám poskytne informace o entitách, reprezentovanými převážně podstatnými jmény a jmennými frázemi. Tato metoda ale selže, jakmile se snažíme získat informace o vztazích mezi entitami. Chceme-li získat dokumenty, obsahující frázi ve tvaru „Společnost X získala společnost Y“, samotná klíčová slova jsou značně nedostatečná, jelikož jediným vhodným je sloveso „získala“. Po jeho použití ale získáme až nepřiměřeně obsáhlou množinu výsledků, kterou je nutné dále zpracovat. Proto je u podobných dotazů vhodnější použít strukturované vyhledávání. Čím dál více internetových vyhledávačů vyvíjí funkce, které to umožňují[6].

Dalším prostředím, kde jsou systémy pro extrakci informací hojně užívány, jsou **firemní aplikace**, užívané například ke *sledování zpráv*. Jedním z prvních systémů pro extrakci informací, užívaných v komerční sféře, byl použit agenturou Reuters a sloužil k automatické tvorbě multimediálních zpráv integrováním videa a obrázků, které zachycovaly dané entity a události zmíněné v článcích.

Mnoho velkých institucí, jako jsou banky, telefonní společnosti a univerzity, shromažďují a uchovávají obrovské množství dat, včetně adres, ve strukturované formě. Často se ale stává, že například údaje pro jednu osobu jsou umístěny ve více databázích odděleně. Při budování datových skladů je proto nezbytné, aby všechny adresy byly převedeny do jednotné formy, jednotlivé položky byly správně identifikovány a duplicity odstraněny. Takto uložené záznamy poté poskytují vyšší účinnost při vyhledávání.

Ve **vědeckých aplikací** se nejčastěji můžeme setkat se systémy na poli bioinformatiky. Prudký rozvoj této oblasti rozšířil rozsah vyhledávaných entit také o biologické objekty, jako jsou proteiny a geny. Cílem těchto aplikací je získávat z volně přístupných databází informace o názvech proteinů a jejich vzájemné působení. Vzhledem k tomu, že tvar jména proteinů a genů je naprosto odlišný od klasických pojmenovaných entit, pomohlo toto odvětví k rozvoji metod a technik používaných pro extrakci informací.

Kapitola 3

Návrh a realizace systému pro extrakci informací

Tato kapitola se zabývá návrhem a implementací systému pro extrakci informací z Wikipedie[1]. Cílem praktické části této práce bylo vytvořit systém, jež bude na vstupu očekávat články z Wikipedie a bude z nich extrahovat informace typu *Kdo je jak spjat s jakým místem*, tzn. že bude vyhledávat relace mezi entitami typu *Osoba* a *Místo*.

3.1 Návrh systému

Jelikož zadání jasně nedefinuje, na základě jakých metod má být systém navržen, pro jejich snažší, ale efektivnější implementaci byly vybrány pravidlové metody. Systém tedy bude obsahovat sadu pravidel, v tomto případě ručně vytvořených.

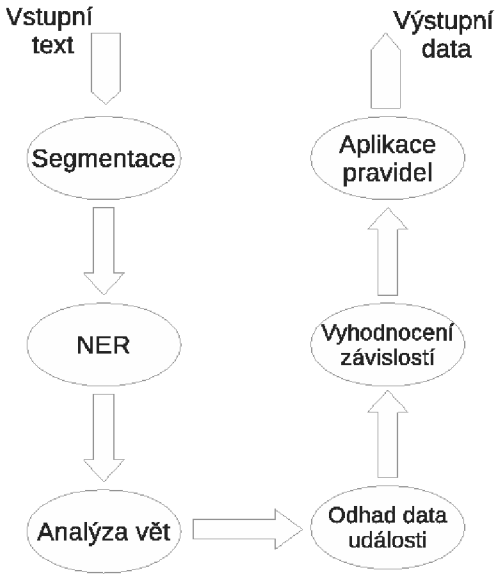
Pro přesnou identifikaci pojmenovaných entit budou použity klasifikátory pojmenovaných entit, pro efektivnější vyhodnocení relací bude navržena sada pravidel, využívající lexikální parsery na úrovni vyhodnocování závislostí jednotlivých tokenů. Pravidla budou dostatečně obecná, aby dosahovaly co nejvyššího pokrytí.

Systém se bude muset umět vypořádat s určitými překážkami, jako jsou například věty s nevyjádřeným podmětem, případně osobou zastoupenou ve větě prostřednictvím ukazovacího zájmena. Systém by měl uživateli poskytnout co nejvíce relevantních informací a v případě zájmu mu poskytnout i hrubý časový odhad, kdy se událost, zmiňovaná ve zdrojovém článku, odehrála. V takovém případě by mělo být uživateli poskytnuto i procentuální ohodnocení úspěšnosti odhadu.

Vstupní data budou tvořena jednotlivými články z Wikipedie v textovém formátu. Články budou převzaty z volně dostupnách záloh serveru Wikipedie[2].

Návrh tohoto systému proběhne ve skriptovacím jazyce Python pro jeho jednoduchost a převážně efektivnost a přirozenost při práci s řetězci a regulárními výrazy. Bude kladen důraz na jednoduchost, rychlost a přesnost. Návrh architektury systému pro extrakci informací je zobrazena na obrázku č. 3.1.

[1] http://www.wikipedia.org/
[2] http://dumps.wikimedia.org/



Obrázek 3.1: Architektura navrženého systému pro extrakci informací

3.2 Implementace

Jelikož zálohy serveru Wikipedia jsou tvořeny soubory ve formátu XML, je nutné z nich nejdříve získat samotný prostý text článku, zbavený veškerého formátování. K tomuto účelu je použit volně dostupný nástroj `WikiExtractor.py`[3], který byl vytvořen Leonardem Souzou z Univerzity v Pise.

Systém pro extrakci informací byl vyvíjen a testován v linuxovém prostředí, jakožto konzolová aplikace. Po jeho spuštění jsou vstupní data předána k vyhledání a označení pojmenovaných entit. K tomuto účelů je použit volně dostupný Stanford NER[4], implementovaný v jazyce Java. V průběhu vývoje systému byla vyzkoušena řada klasifikátorů, určených k vyhledávání pojmenovaných entit. Za zmínku stojí například klasifikátor knihovny `NLTK`[5], případně `decipher_ner`[6]. Od jejich použití bylo ale upuštěno, jelikož nedosahovali tak kvalitních výsledků, jako Stanfordský klasifikátor.

V další části systému dochází k detailní analýze jednotlivých vět. Zde dochází například k nahrazování osobních ukazovacích zájmen patřičného rodu jménem osoby, ke které se vztahují. Zároveň dochází také k uložení letopočtu, pokud ho daná věta obsahuje. Tento údaj je následně použit při odhadu data, kdy se událost odehrála. Pokud datum události není ve větě přímo zmíněno, je použito datum s nejkratší vzdáleností od zpracovávané události.

Z důvodů časových úspor jsou k dalšímu zpracování v podobě vyhodnocení koreferencí připuštěny pouze takové věty, které obsahují pojmenované entyty typu *Osoba* a *Lokace*. Závislosti jsou následně vyhodnoceny volně dostupným Stanfordským parserem[7] a je na ně aplikována sada pravidel, která má za účel získat z dané věty vhodnou relaci, která je

[3] Dostupný na adrese http://medialab.di.unipi.it/wiki/Wikipedia_Extractor
[4] http://nlp.stanford.edu/software/CRF-NER.shtml
[5] http://nltk.org/
[6] https://merlin.fit.vutbr.cz/nlp-wiki/index.php/Decipher_ner
[7] http://nlp.stanford.edu/software/lex-parser.shtml

držena mezi pojmenovanými entitami.

Jelikož systém má získávat informace o osobách, jméno klíčové osoby musí být uvedeno v názvu vstupního souboru. Toto opatření bylo zavedeno z toho důvodu, že v objemnějších článcích se vyskytuje mnoho cizích osob a informací, které poté zkreslují výstup. Klíčová osoba ze jména článku je tedy považována jako referenční a všechny ostatní jména osob, které se neshodují s referenčním jménem, jsou ignorována.

3.2.1 Použité nástroje

Pro správnou funkci vyžaduje systém tyto prostředky:

- interpret jazyka *Python v. 2.5* nebo vyšší
- Stanford Lexicalized Parser v2.0.4 – 2012-11-12
- Stanford NER v1.2.8 – 2013-04-04

3.2.2 Ovládání aplikace

Běh systému je ovlivňován přepínači, které jsou systému předány při jeho spuštění. Jediným povinným parametrem je název vstupního souboru, všechny ostatní parametry jsou nepovinné. Jednotlivé parametry je možné zadávat buď v klasickém, nebo zkráceném tvaru.

Přepínač	**Argument**	**Popis**
-h, --help		Tisk návopědy
-i, --input	FILE	Nastavení vstupního souboru
-o, --output	FILE	Nastavení výstupního souboru, implicitně se tiskne na standardní výstup
-s, --simple		Ze souvětí jsou zpracovávány pouze jednotlivé věty obsahující požadované entity, čímž dochází k navýšení rychlosti běhu systému
-d, –approx-date		Odhad data, kdy se konala událost, viz. sekce 3.2.6
-n, –names-estimation		Odhad jména v určitých formátech vět, viz. sekce 3.2.5

Tabulka 3.1: Ovládání aplikace

Příklad spuštění aplikace:

```
./ie.py -i Jeffrey_Smart.dat -n
> Jeffrey Smart studied in Paris
> In 1963 Jeffrey Smart moved to Italy
> Jeffrey Smart was born in Adelaide in 1921
> In 1948 Jeffrey Smart traveled around Europe
> In 1948 Jeffrey Smart studied in Paris
> In 1950 Jeffrey Smart lived on the island of Ischia in the bay of Naples
> Jeffrey Smart moved to Sydney
> In 1965 Jeffrey Smart returned to Italy
```

3.2.3 Sada pravidel

Veškerá pravidla jsou aplikována na výstup Stanfordského parseru pro vyhodnocení koreferencí. To znamená, že pravidla budou aplikována spíše na větné členy, než slovní druhy. Způsob vyhodnocení koreferencí je možné vidět na obrázku č. 3.2. Předpokládá se, že osoba plní ve větě roli podmětu.

Seznam hledaných větných členů:

- `subj` – Přísudek a podmět. Jestliže podmětem ve větě je hledaná osoba, přísudek bude s největší pravděpodobností tvořit požadovanou relaci.
  - `auxpass` – Vyhledávání pomocného slovesa, je-li přísudek v minulém čase. Příkladem pomocného slovesa tvořící minulý čas je například sloveso *was*.
  - `cop` – Vyhledávání spony, jestliže je ve větě detekován přísudek jmenný se sponou.
- `dobj` – Nalezení předmětu ve větě, souvisí-li s již nalezeným přísudkem.
- `prep_*` – Ověření, zda spolu osoba a lokace ve větě souvisí. Pokud ano, určí se předložka, náležící k lokaci. Pokud se předložka nevztahuje k lokaci, ověří se, nevztahuje-li se předložka k datumu, respektive letopočtu.
- `conj_*` – Určení spojky v souvětí nebo ve výčtu.
- `tmod` – Nalezení časového modifikátoru vztahujícího se k přísudku. Příkladem časového modifikátoru může být například jméno měsíce.
- `num` – Získání číslovky, vztahující se k časovému modifikátoru. V tomto případě by tedy číslovka určovala kalendářní číslo dne.

In 1963 Jeffrey Smart moved to Italy and in 1971 bought the house where he still resides.

```
prep_in(moved-5, 1963-2)
nn(Smart-4, Jeffrey-3)
nsubj(moved-5, Smart-4)
nsubj(bought-11, Smart-4)
root(ROOT-0, moved-5)
prep_to(moved-5, Italy-7)
prep_in(bought-11, 1971-10)
conj_and(moved-5, bought-11)
det(house-13, the-12)
dobj(bought-11, house-13)
advmod(resides-18, where-14)
nn(Smart-16, Jeffrey-15)
nsubj(resides-18, Smart-16)
advmod(resides-18, still-17)
rcmod(house-13, resides-18)
```

Obrázek 3.2: Ukázka způsobu vyhodnocení koreferencí Stanford parserem

3.2.4 Náhrada ukazovacích zájmen

Při implementaci bylo nutné řešit určité problémy, se kterými se bude muset umět systém vypořádat. Řada takových problémů byla již nastíněna v sekci 3.1.

První závažnější překážkou, kterou bylo potřeba řešit, byl příliš vysoký výskyt ukazovacích zájmen, které ve větě zastupovaly klíčovou osobu. Rozhodl jsem se, že se bude provádět náhrada ukazovacích zájmen vlastními jmény. K této činnosti je nejdříve nutné zjistit rod zájmen, která mají být nahrazena – zda-li mužská nebo ženská. To je provedeno prostým spočítáním výskytu obou rodů zájmen. Nahrazují se ty zájmena, která se v článku nachází ve vyšším počtu. Nelze ale nahradit každý výskyt daného zájmena. Předpokládá se, že osobní ukazovací zájmeno se vztahuje k nejblíže uvedené osobě, nacházející se ve větě *před* zájmenem. Pokud před zájmenem není uvedena žádná osoba, považuje se, že zájmeno odkazuje na klíčovou osobu. Pokud bychom měli článek, pojednávající o umělci jménem Jeffrey Smart, věta

He later studied in Paris with Fernand Léger.

by byla změněna na

Jeffrey Smart later studied in Paris with Fernand Léger.

Ovšem zájmeno ve větě ze stejného článku

Fernand Léger moved to Italy where he visited Jeffrey Smart.

by už nahrazeno nebylo, jelikož se nevztahuje ke klíčové osobě.

3.2.5 Odhad jména

Další překážkou byl určitý tvar vět, v nichž je podmět nevyjádřený. Jelikož tyto věty následně selhávaly při vyhodnocování koreferencí, rozhodl jsem se implementovat možnost predikce jména v tomto druhu vět. Zde už situace není natolik triviální, jako u nahrazování zájmen, a je potřeba potřeba navrhnout širší sadu podmínek, které věta musí splňovat, aby do ní mohlo být jméno doplněno. Těchto podmínek je celkem 5 a jmenovitě to jsou:

- Jméno klíčové osoby musí být zmíněno v některé z jednotlivých částí souvětí
- Pojmenovaná entita typu *Osoba* se nesmí v té větě ze souvětí, do které má být jméno doplněno, vyskytovat
- Pokud je před větou, určenou k doplnění jména, zmíněna nějaká osoba, smí se jednat jen a pouze o osobu klíčovou
- Věta ze souvětí, do které má být jméno doplněno, musí obsahovat entitu typu *Lokace*
- Věta ze souvětí, do které má být jméno doplněno, musí začínat slovesem

Tyto pravidla byla zavedena převážne kvůli poměrně častě se objevujícím frázím s nevyjádřeným podmětem, jako je například věta ve tvaru na obrázku 3.3. Jak již bylo zmíněno v sekci 3.2.3, nedílnou částí věty, kontrolovanou vytvořenou sadou pravidel, je podmět.

Aby z exemplární věty bylo možné korektně získat požadované informace, je nutné ji upravit do tvaru

Born in Adelaide in 1921, Smart started drawing at an early age on anything he could find.

```
partmod(started-8, Born-1)
prep_in(Born-1, Adelaide-3)
prep_in(Adelaide-3, 1921-5)
nsubj(started-8, Smart-7)
root(ROOT-0, started-8)
csubj(find-19, drawing-9)
det(age-13, an-11)
amod(age-13, early-12)
prep_at(drawing-9, age-13)
nn(Smart-17, anything-15)
nn(Smart-17, Jeffrey-16)
prep_on(age-13, Smart-17)
aux(find-19, could-18)
ccomp(started-8, find-19)
```

Obrázek 3.3: Vyhodnocení koreferencí u věty obsahující nevyjádřený podmět

Jeffrey Smart was born in Adelaide in 1921, Smart started drawing at an early age on anything he could find.

který už není problém zpracovat. Pro uživatelsky přívětivý výstup je počáteční sloveso věty kontrolováno a dle jeho druhu je rozhodnuto, má-li se mezi vlastní jméno a sloveso přidat ještě pomocné sloveso například pro správnou tvorbu minulého času, v našem případě *was*.

3.2.6 Odhad data

Další částí systému, stojící za zmínku, je odhad data, kdy byla událost vykonána. V každé větě ve zdrojovém textu je vyhledáván token nesoucí údaj o letopočtu. Jeho pozice v textu je zapamatována. Projde-li následně některá z dalších vět sadou pravidel, určených k vyhodnocení relace, a tato věta sama o sobě nenese časový údaj vztahující se k relaci, vyhodnotí se nejblíže se vyskytující datum vzhledem k relaci. Dle pozice data a relace je následně určeno, stala-li se událost *před, po* nebo *v* daném roce. V závislosti na vzdálenosti data a relace je také vypočítán procentuální ohodnocení provedeného odhadu data.

```
Jeffrey Smart studied in Paris (after 1942-1947 ~87%)
```

Kapitola 4

Experimentální výsledky

Jako vstupní data, potřebná k experimentálnímu testování systému, byly použity články ze záložního souboru Wikipedie, konkrétně `enwiki-20130204-pages-articles10.xml--p000925001p001325000`. Veškerá data, platná ke dni 4. 2. 2013, byla řádně předzpracována nástrojem `WikiExtractor`. Obecně je ale možné pro testování systému použít jakákoliv nestrukturovaná textová data, která neobsahují žádné specifické formátování.

4.1 Návrh testovací sady

Jako testovací vzorek bylo použito 100 náhodně vybraných článků, které posloužily jako vstupní data pro navržený systém. Zkoumáním 500 náhodných článků bylo zjištěno, že průměrná velikost textových dat jednoho článku se pohybuje okolo 5 kB. Toto zjištění bylo reflektováno i na mnou náhodně vybranou množinu testovacích dat, jejichž velikost byla rovnoměrně rozložena kolem zjištěné průměrné hodnoty. Články byly vybrány z rozdílných kategorií, aby bylo testování co nejobecnější. Všechny testovací články ale pojednávaly o osobách. Veškerá získaná data byla následně ručně zkontrolována.

Při analýze výsledků jsem pozornost zaměřil především na rozdíly v získaných informacích, které byly způsobeny zadáním rozdílných přepínačů při spuštění systému. Hlavním cílem experimentálního testování bylo zjistit, jaký je poměr mezi správnými, špatnými a chybějícími odpovědmi. Vedlejším cílem testování bylo zjistit, na čem a v jaké míře je závislá doba odezvy systému a jak se systém bude chovat při zpracovávání velkého množství vstupních dat.

4.2 Analýza dosažených výsledků

Testovací množina dat obsahovala celkem 517 možných správných odpovědí, které by měl systém se 100% úspěšností produkovat.

Pokud od systému nebyly vyžadovány **žádné korekce** vstupních vět, produkoval výsledky zobrazené v následující tabulce tabulce:

Správných odpovědí	348
Chybných odpovědí	37
Neprodukovaných odpovědí	169
Neúplně vyhodnocených relací	7
Průměrná doba běhu systému	25 s

Bylo-li od systému požadováno, aby **zpracovával pouze ty části souvětí**, ve kterých se nacházely hledané entity, dosáhlo se daleko lepších výsledků. Počet správných odpovědí vzrostl, za současného poklesu počtu odpovědí chybných a neprodukovaných:

Správných odpovědí	405
Chybných odpovědí	23
Neprodukovaných odpovědí	112
Neúplně vyhodnocených relací	7
Průměrná doba běhu systému	18 s

Tohoto jevu bylo dosaženo zřejmě proto, že rozsáhlá a rozvinutá souvětí byla příliš náročná na vyhodnocení koreferencí. Úpravou takového souvětí na větu jednoduchou, obsahující požadované pojmenované entity, se docílilo i snížení náročnosti na vyhodnocení koreferencí, na úkor ztráty občasných dodatečných informací, jako například časové údaje. Další pozorovanou změnou je snížená doba běhu systému.

Dle dostupných metrik, popsaných v sekci 2.1, je možné určit přesnost a pokrytí systému následovně:

$$P_1 = \frac{348}{385} = 0,903896104 \qquad R_1 = \frac{348}{517} = 0,67311412$$

$$P_2 = \frac{405}{428} = 0,946261682 \qquad R_2 = \frac{405}{517} = 0,783365571$$

Přesnost P_1 a pokrytí R_1 se vztahují k systému, jež neprovádí ve zdrojových datech žádné změny. Pro takový systém budeme nadále používat označení S_1. Přesnost P_2 a pokrytí R_2 se vztahují naopak k systému, který ze souvětí přebírá pouze takové věty, které obsahují hledané pojmenované entity. Takový systém bude nadále označen jako S_2.

Zajímavé je také srovnání závislosti velikosti vstupních dat na přesnosti a pokrytí systému:

S_1	$< 3kB$	$> 20kB$
Přesnost	0,95846	0,81351
Pokrytí	0,84187	0,59462

S_2	$< 3kB$	$> 20kB$
Přesnost	0,96153	0,83874
Pokrytí	0,89285	0,61701

Lze vypozorovat, že přesnost a pokrytí jsou nepřímo úměrné velikosti zpracovávaných dat. Tato statistika byla prováděna na menším vzorku dat, jednotlivé výsledky tudíž mohou být zkreslené. Z pozorování ovšem vyplynulo, že pro menší vzorky dat poskytoval systém lepší výsledky. To je patrně způsobeno tím, že v rozsáhlých článcích se často vyskytují věty daleko rozvinutější, než je tomu u méně obsáhlých článků, a nesou daleko více informací. Korektně vyhodnotit koreference v souvětích s několikanásobným přísudkem není zcela triviální záležitostí a navrhnutá sada pravidel na to není dostatečně optimalizovaná.

Počet výskytů zájmen	1328
Počet náhrad zájmen	921

Tabulka 4.1: Poměr mezi výskytem osobních ukazovacích zájmen a jejich nahrazením vlastním jménem

4.2.1 Správnost úprav zdrojových vět

Pro zajištění vyššího pokrytí systému byla implementována řada metod, které upravovaly zdrojový text, s vidinou dosažení lepších výsledků při získávání informací. Nejčastější a nejúčinnější bylo nahrazování zájmen, popsané v sekci 3.2.4. Chyby se při aplikaci těchto úprav objevovaly pouze ojediněle a byly způsobeny tím, že nebyl znán hlubší kontext výskytu zájmen ve větě. Typickým případem, kdy byly věty chybně upraveny, byly například věty obsahující zájmena v přímé či nepřímé řečí. Na výstup systému to ale nemělo veliký vliv, jelikož se nestávalo často, aby v přímé nebo nepřímé řeči byla zároveň zmíněna i pojmenovaná entita typu *Lokace*. Z tabulky 4.2.1 vyplívá, že v testovacím vzorku dat bylo nahrazeno zhruba 70% zájmen.

Další úpravou, která měla vliv na množství získaných informací, bylo odhadnutí jména při výskytu věty s nevyjádřeným podmětem. Analýza přirozeného jazyka je velice složitá vědní disciplína a není tedy divu, že tato metoda poskytuje rozporuplné výsledky. Úspěšnost odhadnutí správného podmětu ve větě se pohybovala okolo 50%. Proto tato metoda není v systému užívaná implicitně, ale záleží na uživateli, bude-li ji chtít při zpracovávání dat aplikovat. Další velkou nevýhodou této metody je, že znatelně prodlužuje dobu běhu systému, někdy až dvojnásobně. Děje se tak z důvodu volání externích klasifikátorů pro vyhodnocení podmínek, vyjmenovanými v sekci 3.2.5.

Kapitola 5

Závěr

Cílem bakalářské práce bylo seznámit se s metodami užívanými při extrakci informací. Jelikož je tato oblast značně obsáhlá, práce je zaměřena především na metody související s extrakcí entin a relací. Další částí bakalářské práce bylo návrhnout systému pro extrakci informací z Wikipedie. Výsledný systém ke svému běhu používá klasifikátory vytvořené na Stanfordské univerzitě a umožňuje uživateli získat z textu informace ve smyslu *kdo je jak spjat s jakým místem*. Navržený systém je založený na pravidlových metodách. Z dosažený experimentálních výsledků byla následně náhodně vybrána množina 100 článků, u kterých byla provedena manuální kontrola extrahovaných informací Na jejich základě byla poté vyhodnocena přesnost a pokrytí systému.

Tato práce nepřichází s žádnou novou metodou extrakce informací. Přínos této práce spočívá v aplikaci již známých metod a ve vytvoření aplikace, která v textu vyhledává požadované informace. Největší nedostatek této aplikace tkví v použití ne zcela 100% úspěšných klasifikátorů pojmenovaných entit, a to i přes snahu vybrat takový klasifikátor, který dosahuje nejlepších výsledků. Při extrakci relací dosahoval navržený systém velice slušných výsledků. Jelikož jsou data extrahována z prostého textu, je možné tento systém použít i na jiné články, než jsou články získané ze serveru Wikipedie.

Systém je otevřen mnoha úpravám a vylepšením. Je možné například implementovat vyhledávání informací mezi více druhy pojmenovaných entit, než je *osoba* a *lokace*. Teoreticky by bylo možné navrhnout taková vylepšení, při kterých by uživatel explicitně zadal druhy pojmenovaných entit, mezi kterými mají být v textu relace vyhledávány. Výsledný systém by poté nebyl tolik omezený určitou kategorií článků, pro které by byl použitelný. Další možností je změnit používané klasifikátory pojmenovaných entit. Jelikož klasifikátory nejsou interní součástí systému, změna by to nebyla příliš razantní. Bylo by také možné pokusit se implementovat druhou variantu systému, která by byla založena na jiných metodách, než pravidlových. Následně by se porovnala účinnost nových metod s předchozí variantou, založenou na pravidlových metodách.

Literatura

[1] Eikvil, L.: Information extraction from world wide web-a survey. 1999.

[2] Grishman, R.: Information Extraction: Capabilities and Challenges. *Notes prepared for the 2012 International Winter School in Language and Speech Technologies*, 2012.

[3] Hachey, B.; Grover, C.; Tobin, R.: Datasets for generic relation extraction. *Natural Language Engineering*, ročník 18, č. 01, 2012: s. 21–59.

[4] Lafferty, J.; McCallum, A.; Pereira, F. C. N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., 2001, ISBN 1-55860-778-1, s. 282–289.

[5] Moncecchi, G.; Minel, J.-L.; Wonsever, D.; aj.: A survey of kernel methods for relation extraction. In *Workshop on NLP and Web-based technologies*, 2010.

[6] Sarawagi, S.: Information extraction. *Foundations and trends in databases*, ročník 1, č. 3, 2008: s. 261–377.

Příloha A

Obsah DVD

Adresářová struktura DVD:

`/zprava`	*technická zpráva*
`/doc`	*zdrojové soubory k technické zprávě*
`zprava.pdf`	*technická zpráva ve formátu pdf*
`/scripts`	*vlastní skripty*
`/utils`	*použité nástroje*
`README`	*popis spuštění systému*