VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

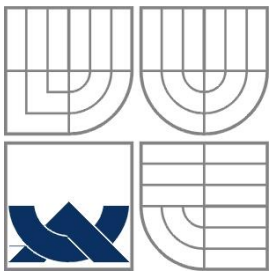
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EXTRAKCE INFORMACÍ Z WIKIPEDIE

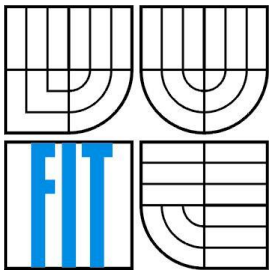
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Miroslav Pospíšil
AUTHOR

BRNO 2015



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 — Miroslav Pospíšil
AUTHOR

VEDOUCÍ PRÁCE — Doc. RNDr. Pavel Smrž, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Tato bakalářská práce se zabývá převodem článků z Wikipedie do vertikálního textu. Zpracování vertikálního textového korpusu pomocí lingvistických nástrojů a převedení do formátu pro indexaci. Vložení článků s přidanými anotacemi do indexu MG4J. Zpracování statistiky počtu výskytů vět obsahujících určité typy relací. Z nejčastějších výskytů vytvořit vyhledávací vzory. Zobecňování vzorů. Navrhnutí systému pro extrakci vybraných relací z článků v indexu. Implementace systému a testování funkčnosti. Spuštění extrakce relací pomocí vzorů a vyhodnocení výsledků extrakce.

Abstract

This bachelor's thesis deals with the transfer of articles from Wikipedia into vertical text. Processing of the vertical text corpus using linguistic tools and converting into a format for indexing. Inserting articles with added annotations to index of MG4J. Processing statistics on the number of occurrences of phrases containing certain types of relations. Creating searching patterns based on the most frequent occurrences. Generalization of patterns. Designing a system for extraction of selected relations from articles in the index. Implementation of the system and testing functionality. Starting extractions based on patterns and evaluation of results of extraction.

Klíčová slova

Extrakce informací, Wikipedie, převod textů, znalostní databáze, vertikální text, indexace, MG4J, anotace jmen a názvů, tvorba a zobecňování vzorů, systém pro extrakci informací, extrakce relací.

Keywords

Information extraction, Wikipedia, text converting, knowledge base, vertical text, indexation, MG4J, annotations of names and titles, creating and generalization of patterns, system of information extraction, extraction of relations.

Citace

Pospíšil Miroslav: Extrakce informací z Wikipedie, bakalářská práce, Brno, FIT VUT v Brně, 2015

Extrakce informací z Wikipedie

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. RNDr. Pavla Smrže, Ph.D.
Další informace mi poskytli Ing. Lubomír Otrusina, Ing. Jan Kouřil, Ing. Jaroslav Dytrych.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Miroslav Pospíšil
20. 5. 2015

Poděkování

Děkuji Doc. RNDr. Pavlu Smržovi, Ph.D. za vedení vypracování mé bakalářské práce. Dále Ing. Janu Kouřilovi za velmi ochotnou a rychlou pomoc a rady, a také za jeho podporu a materiály. Poděkování náleží také Ing. Lubomíru Otrusinovi a Ing. Jaroslavu Dytrychovi za pomoc a zodpovídání mých dotazů.

Obsah

Obsah 1
1 Úvod 3
2 Extrakce informací a relací 4
2.1 Extrakce informací jako pojem 4
2.2 Úvod do extrakce relací 4
2.3 Kontrolované „*supervised*“ metody 5
2.3.1 Metody založené na vlastnostech 6
2.3.2 Metody jádra 7
2.3.3 Shrnutí 10
2.4 Polo-kontrolované „*semi-supervised*“ metody 11
2.4.1 DIPRE (Brin) 12
2.4.2 Snowball (Agichtein & Gravano) 13
2.4.3 KnowItAll (Etzioni) a TextRunner (Banko) 15
2.4.4 Shrnutí 16
2.5 Za oponou binárních relací 16
2.6 Vyhodnocování extrakce relací 18
2.6.1 Vyhodnocování kontrolovaných metod 18
2.6.2 Vyhodnocování polo-kontrolovaných metod 18
3 Zpracování zdrojových dat 19
3.1 Zdrojová data 19
3.2 Možnosti zpracování zdrojových dat 19
3.2.1 Vlastní program 19
3.2.2 Program pro převod zdroje do vertikálního textu 19
3.2.3 Wikipedia Extractor 20
3.3 Zpracování zdrojových dat 20
3.3.1 Přidání odkazů k nadpisům a zájmenům 21
3.4 Zpracování vertikálního textu 21
3.4.1 Vyplnění CONLL sloupců 22
3.4.2 Vyplnění sloupců anotovaných informací 24
4 Indexace 25
4.1 Možnosti indexace 25
4.1.1 ElasticSearch 25
4.1.2 MG4J 26
4.2 Naindexování dat 26
4.3 Spuštění indexu a příprava pro vyhledávání 27
4.4 Vytvoření statistik 27
5 Extrakce relací z Wikipedie 28
5.1 Extrakce relace: „ARTIST – has influenced - ARTIST“ 29
5.1.1 Nalezení výskytů a vytvoření n-tice 29
5.1.2 Vytvoření vzorů 30

5.1.3 Nalezení nových výskytů 30
5.2 Extrakce relace: „LOCATION NAME – HISTORICAL NAME“ 31
5.2.1 Nalezení výskytů a vytvoření n-tic 31
5.2.2 Vytvoření vzorů 32
5.2.3 Nalezení nových výskytů 32
5.3 Extrakce relace: „PERSON – egyptologist“ 33
5.3.1 Nalezení výskytů a vytvoření n-tic 34
5.3.2 Vytvoření vzorů 34
5.3.3 Nalezení nových výskytů 35
5.4 Extrakce relace: „PERSON – was educated at – SCHOOL“ 36
5.4.1 Nalezení výskytů 36
5.4.2 Vytvoření vzorů 37
5.4.3 Nalezení nových výskytů 37
5.5 Extrakce relace: „PERSON – was born – LOCATION“ 37
5.5.1 Nalezení výskytů a vytvoření n-tic 38
5.5.2 Vytvoření vzorů 38
5.5.3 Nalezení nových výskytů 38
5.6 Vyhodnocení extrakce vztahů 40
5.6.1 Vyhodnocení relace: „ARTIST – has influenced - ARTIST“ 41
5.6.2 Vyhodnocení relace: „LOCATION NAME – HISTORICAL NAME“ 42
5.6.3 Vyhodnocení relace: „PERSON – egyptologist“ 42
5.6.4 Vyhodnocení relace: „PERSON – was educated at – SCHOOL“ 43
5.6.5 Vyhodnocení relace: „PERSON – was born – LOCATION“ 43
6 Systém pro extrakci relací 45
7 Závěr 46

1 Úvod

V dnešní době každý vyhledává informace na veřejně dostupných databázích obsahujících články a hlavně informace prakticky o čemkoli. Je proto důležité a zajímavé se více zaměřovat na možnosti extrakce konkrétních informací z těchto rozsáhlých databází pro rychlejší a pohodlnější získání odpovědí.

V této práci se zabývám nejdříve teoretickým rozborem a znalostmi z oboru extrakce relací. Zpracováním zdrojových dat Wikipedie. Vytvoření vertikálního textu z jednotlivých článků a následným zpracováním lingvistickými nástroji pro určování syntaktických a sémantických vlastností slov přirozeného jazyka. Následně převedení výstupu do formátu pro uložení do indexu.

Zpracování statistiky výskytu vět obsahující vybrané relace mezi entitami. Vyzkoušet kompletní proces extrakce relací na základě výsledků navrhnout jednoduchý systému pro extrakci vybraných druhů relací z indexovaných článků. Implementace systému a otestování funkčnosti. Spuštění extrakce relací pomocí vyhledávacích vzorů. Vyhodnocení výsledků extrakce relací.

2 Extrakce informací a relací

2.1 Extrakce informací jako pojem

„Extrakce informací je činnost, spadající pod množinu činností zvaných souborně dolování v textech. Úkolem extrakce předen definovaného typu informací je ve vstupních dokumentech (tvořených ve většině případů nestrukturovaným textem v přirozeném jazyce) vyextrahovat požadované informace (jména, hodnoty, vztahy), které jsou následně uloženy do databáze nebo šablon, kde jsou přístupné pro další zpracování.

Extrakce informací může být použita pro různé druhy textů, například:

- Internetové stránky.
- Klasifikované inzeráty.
- Lékařské zápisky.
- „Newsové“ skupiny na internetu.
- Novinové články.
- Vědecké články.“[1]

2.2 Úvod do extrakce relací

„Existuje obrovské množství nestrukturovaných elektronických textů. Jakým způsobem si může člověk nechat pomoct pro pochopení všech těchto informací? Populárním nápadem je překlenout nestrukturovaný text na strukturovaný pomocí anotování sémantických informací. Nicméně objem a heterogenita dat způsobují nemožnost anotování lidskými zdroji. Ideální by bylo mít počítač, který provede anotování na všechna požadovaná data. Obvykle máme zájem o vztahy mezi subjekty, jako jsou osoby, organizace a místa. Příklady vztahů jsou osoby a jejich příslušnost, ovlivnění osob a podobné. Aktuální nejmodernější rozpoznávače jmenných entit (NER – Named entity recognizer), jako například (Bikel, An algorithm that learns what's in a name) a (Finkel, Incorporating non-local information into information extraction systems by gibbs sampling) dokáží automaticky označit data s vysokou přesností. Celý proces extrakce vztahů není triviálním úkolem. Počítač potřebuje vědět jak rozpoznat části textu, které mají sémantické vlastnosti, pro tvorbu správných anotací. Získání sémantických vztahů mezi subjekty v přirozeném jazyce je důležitým krokem pro aplikace pracující s přirozeným jazykem.

Vztah je definován v podobě n-tice $t = (e_1, e_2, \ldots, e_n)$, kde prvky e_i jsou subjekty v předem definované relaci R, v dokumentu D. Většina systémů pro extrakci se zaměřují na extrahování binárních relací. Příklady binárních relací jsou: „*se nachází v*“ („*located-in*“), „*je otcem …*“ („*father-of*“).

[1] SEDLÁČEK Petr, *Extrakce informací z textu*, 2006

Začneme s kontrolovanými „*supervised*“ přístupy, které specifikují úkol extrakce relací pouze na binární relace. Dále se zaměříme na metody založené na vlastnostech (Kambhatla, Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations), (Zhao & Grishman, Extracting relations with integrated information using kernel methods), a na metody jádra, dohlížející na extrakce relací. Hlavní výhodou metod jádra je, že nabízejí efektivní řešení, která nám umožní prozkoumat velký (často exponenciální nebo v některých případech nekonečný) funkční prostor v polynomu výpočetního času bez potřeby explicitně definovat funkce. Budeme porovnávat jádra jako je stromové jádro „*tree - kernel*“ (Zelenko, Kernel methods for relation extraction), subsekvenční jádro „*subsequent - kernel*“ (Bunescu & Mooney, Subsequence kernels for relation extraction) a jádro stromu závislostí „*dependency – tree - kernel*“ (Bunescu & Mooney, A shortest path dependency kernel for relation extraction).

Nedávné polo-kontrolované „*semi-supervised*“ a „*bootstrapping*“ přístupy si vyžádaly zvláštní pozornost. V další části si ukážeme DIPRE (Brin, Extracting patterns and relations from the world wide web) a Snowball (Agichten & Gravano, Snowball: Extracting relations from large plain-text collections) systémy, které vyžadují pouze malou sadu označených instancí textů nebo několik ručně vytvořených extrakčních vzorů pro každou relaci, k zahájení procesu trénování. Všechny tyto polo-kontrolované systémy používají stejný přístup jako v Yarowskyho algoritmu, ve smyslu disambiguace významů (Yarowsky, Unsupervised word sense disambiguation rivaling supervised methods). Také „*KnowItAll*“ (Etzioni, Unsupervised Named-Entity Extraction from the Web) nebo „*TextRunner*“ (Banko, Open information extraction from the web) jsou navrhnuté jako systémy pro extrakci relací, které mají své vlastní trénovací klasifikátory binárních relací.

Další částí jsou způsoby extrahování relací (McDonald, Simple algorithms for complex relation extraction with applications to biomedical ie) vyšších řádů. Novinka (McDonald) rozkládá komplexní složité vztahy do binárních relací, které jsou zastoupeny ve formě grafu a algoritmus rekonstruuje složité vztahy tak, že vybírá n-tice z vybraných maximálních hodnot v grafu. Hlavní výhodou jejich přístupu je, že algoritmus umožňuje použití klasifikátorů binárních relací, které byly dobře prostudovány, a také jsou velmi často přesné.“[2]

2.3 Kontrolované „*supervised*“ metody

„Nejdříve budeme považovat úkol extrakce relací jako klasifikační úkol. Pro jednoduchost a přehlednost se omezíme na binární relace, tedy relace mezi dvěma entitami. N-ární extrakce relací budou popsány v dalších částech. Pro danou větu $S = w_1, w_2, \ldots, e_1 \ldots, w_j, \ldots e_2, \ldots, w_n$, kde e_1 a e_2 jsou entity, je mapovací funkce $f(.)$ dána:

$$f_R(T(S)) = \begin{cases} +1 & \text{Je } e_1 \text{ a } e_2 \text{ ve spojení podle relace } R \\ -1 & \text{Jinak} \end{cases} \quad (1)$$

Kde T(S) jsou vlastnosti, které jsou extrahovány z S. V podstatě mapovací funkce $f(.)$ rozhoduje, zda entity ve větě jsou nebo nejsou v relaci. Jinak řečeno úkol extrakce relací subjektů se

[2] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007

stává detekcí vztahů mezi entitami. Jsou-li k dispozici soubory označených pozitivních a negativních příkladů relací pro trénování, funkce $f(.)$ může být zkonstruována jako diskriminační klasifikátor, například jako perceptron. Tyto klasifikátory mohou být trénovány používáním souborů vlastností vybraných po provedení textové analýzy (například POS značení, rozbor závislostí, atd.) označených vět. Na druhé straně vstup klasifikátorů může často mít podobu bohatých strukturálních reprezentací jako například derivační stromy. V závislosti na formě vstupu klasifikátoru pro trénování, kontrolované přístupy pro extrakci relací jsou dále děleny na metody založené na vlastnostech a metodách jádra.“[3]

2.3.1 Metody založené na vlastnostech

„Máme soubory pozitivních a negativních příkladů relací. Syntaktické a sémantické vlastnosti mohou být extrahovány z textu. Tyto získané údaje slouží jako vodítka pro rozhodování o tom, které entity ve větě jsou v relaci nebo ne.

Syntaktická data extrahovaná z věty zahrnují:

- Entity samotné.
- Typy obou prvků.
- Sekvenci slov mezi subjekty.
- Počet slov mezi entitami.
- Cestu v derivačním stromu obsahujících oba objekty zkoumání.

Sémantické informace obsahují cestu mezi prvky v rozboru závislostí. Extrahované syntaktické i sémantické vlastnosti jsou prezentovány v klasifikátoru formou vektoru vlastností, pro trénování nebo klasifikaci. (Kambhatla) trénuje log-lineární model použitím vlastností popsaných pro klasifikaci entit. Na druhou stranu, (Zhao & Grishman) a (GuoDong, Exploring various knowledge in relation extraction) používají „*SVM*” trénované na tyto vlastnosti, užitím polynomického a lineárního jádra, respektive pro klasifikování různých typů relací entit. Některé vlastnosti jsou dobrými indikátory relací entit, zatímco jiné nemusí být. Je proto důležité vybrat pouze ty vlastnosti, které jsou relevantní pro konkrétní úkol. Metody založené na vlastnostech mají heuristické volby a vlastnosti musí být vybírány metodou pokus – omyl, jako základ pro maximální výkon. Vzhledem k tomu, NLP aplikace obecně a extrakce informací zahrnují zejména strukturované reprezentace vstupních dat, může být obtížné dospět k optimální podmnožině příslušných vlastností. K nápravě problému výběru vhodného souboru vlastností, specializovaná jádra jsou navrhnutá pro extrakci informací za účelem využití bohaté reprezentace vstupních dat, jako například nízko-úrovňové derivační stromy. Tato jádra zkoumají vstupní reprezentaci obtížným způsobem implicitně ve vyšším dimenzionálním prostoru.“[4]

[3] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction,* 2007
[4] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction,* 2007

2.3.2 Metody jádra

„Jádra použité pro extrakce relací (nebo detekci relací) jsou založené na jádrech řetězců v (Lodhi, Text classification using string kernels). Jádra řetězce byly popsány v (Lodhi), v kontextu textové klasifikace. Máme dva řetězce *x* a *y*. Jádra řetězců jsou porovnány na podobnost, postavenou na počtu sub-sekvencí, které jsou podobné pro oba řetězce. Čím více společných sub-sekvencí, tím větší je podobnost mezi dvěma řetězci. Každý řetězec může být mapován vyšším dimenzionálním prostorem, kde každá dimenze koresponduje s přítomností nebo absencí konkrétní sub-sekvence.

Pro příklad, řetězec *x = cat* může být vyjádřen ve vyšším dimenzionálním prostoru jako sub-sekvence:

$$\begin{aligned} \phi(x = cat) &= [\phi_a(x) .. \phi_c(x) .. \phi_t(x) .. \phi_{at}(x) .. \phi_{ca}(x) .. \phi_{ct}(x) .. \phi_{cat}(x) ..] \\ &= [\ \lambda \quad .. \ \lambda \quad .. \ \lambda \quad .. \ \lambda^2 \quad .. \ \lambda^2 \quad .. \ \lambda^2 \quad .. \ \lambda^3 \quad ..] \end{aligned} \tag{2}$$

Kde $\lambda \in (0, 1]$ je faktor útlumu – kde jsou delší a nesouvislé sub-sekvence penalizovány. Část $\phi_{ct}(cat)$ je penalizována více (λ^3) než $\phi_{ca}(cat)$ (λ^2) a $\phi_{at}(cat)$ (λ^2) od chvíle, kdy se „*ct*“ vyskytuje nesouvisle v „*cat*“. Obecněji, řekněme, že *u* je sub-sekvence přítomnosti na indexech $i = i_1, i_2, ..., i_{|u|}$ ($i_1 \leq i_2 \leq ... \leq i_{|u|}$) uvnitř řetězce *x* ($u = x[i]$) a řekněme, že $l(i) = i_{|u|} - i_1 + 1$ je délka řetězce *u*. Od chvíle, kdy *u* může být přítomno uvnitř *x* více než jednou, souřadnice řetězce *x* korespondují s *u* ve vyšším dimenzionálním prostoru daným:

$$\phi_u(x) = \sum_{i:u=x[i]} \lambda^{l(i)} \tag{3}$$

Pokud *U* je soubor všech možných seřazených sub-sekvencí přítomných v řetězcích *x* a *y*, jádro podobnosti mezi *x* a *y* je dáno:

$$\begin{aligned} K(x, y) &= \phi(x)^T \phi(y) \\ &= \sum_{u \in U} \phi_u(x)^T \phi_u(y) \end{aligned} \tag{4}$$

Tento přímočarý výpočet (4) použitím předchozího vzorce (3) zahrnuje exponenciální složitost. V praxi se však výpočet (4) implicitně provádí účinným způsobem za použití techniky dynamického programování popsané v (Lodhi). Složitost výpočtu jádra řetězce v (4), je dána: $O(|x||y|^2)$ $(|x| \geq |y|)$.

Obecnější výklad rovnice je, že pokud *x* a *y* jsou dva objekty, $K(x, y)$ počítá strukturální společné části mezi nimi, tedy počítá jejich podobnost. Prvky *x* a *y* mohou být objekty, jako jsou řetězce, sekvence slov, derivační stromy atd. V extrakci relací, pokud x^+ a x^- jsou objekty, reprezentující pozitivní a negativní příklady relací mezi entitami navzájem, a pokud y je testovací příklad, z $K(x^+, y) > K(x^-, y)$ vyplývá, že *y* obsahuje relaci. V praxi je však $K(x, y)$ podobnost funkce,

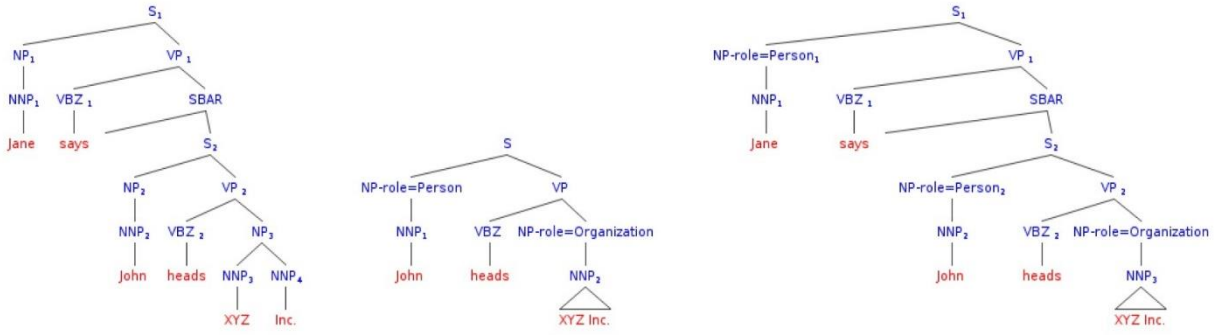
použitá v klasifikátorech jako „*SVMs*“, „*Voted Perceptron*“ a podobně. Pro úkol extrakce relací, objekty x^+, x^- a y mohou být reprezentovány jako sekvence slov okolo subjektů nebo derivační stromy obsahujících entity.

Jádra souborů vlastností

Vezměme si větu „*Centrála společnosti Google se nachází v Mountain View*“. Vzhledem k tomu, že „*Google*“ a „*Mountain View*“ jsou pojmenované entity, slova „*nachází*“ a „*centrála*“ indikují, že se jedná o vztah organizace – lokace. Můžeme tedy učinit závěr, že kontext subjektů může být použit pro rozhodnutí, zda jsou v relaci nebo ne. (Bunescu & Mooney, b) používají slovní kontext pojmenovaných prvků pro extrakci interakcí proteinů z medicínských abstraktů. Věta $s = w_1, \ldots, e_1, \ldots, w_i, \ldots, e_2, \ldots, w_n$ obsahující relační entity e_1 a e_2, mohou být popsány jako $s = sb\ e_1\ sm\ e_2\ sa$. Zde *sb*, *sm* a *sa* jsou části slovního kontextu „*before*“, „*middle*“ a „*after*“ entit relací. Nyní daná testovací věta t obsahuje entity e_1' a e_2'. Podobnost jejich „*before*“, „*middle*“ a „*after*“ částí s těmi z věty *s*, je počítána použitím sub-sekvenčních jader. Jádro počítá podobnosti mezi dvěma sekvencemi na úrovni slova spíše, než na úrovni písmen (jako v jádrech řetězců). V (Bunescu & Mooney, b) jsou definovány tři sub-jádra pro každou „*before*“, „*middle*“ a „*after*“ část kontextu entit, a ty jsou pak kombinovány do jádra jednoduchou sumarizací všech sub-jader. Užíváním sub-sekvenčních jader v konjunkci s „*SVMs*“ zlepšuje přesnost a efektivitu úkolu extrakce relací.

Stromová jádra

Ve srovnání s předchozím přístupem, (Zelenko) nahrazuje řetězce v rovnici strukturovanými nízko-úrovňovými derivačními stromy sestavenými z vět. Používají se stromová jádra a připojují se k „*SVM*“ a „*Voted Perceptron*“ pro úkoly extrakce relací typu: osoba - přičlenění a organizace – lokace, z textu. To znamená, že jádro je navrženo tak, aby počítalo podobnosti mezi dvěma subjekty rozšířenými o struktury nízko-úrovňových derivačních stromů. Vzhledem k větě, nízko-úrovňový rozbor je konstruován nejdříve. Důvodem pro použití mělkých derivačních stromů namísto plných derivačních stromů je, že tyto stromy mají tendenci být více robustní a spolehlivé. Nyní, vzhledem k mělkému rozboru, pozitivní příklady jsou konstruovány podle výpočtu nejnižšího podstromu, který pokrývá obě entity ve vzájemném vztahu. Pokud podstrom nezahrnuje entity v relaci, je mu přiřazeno negativní označení.



(a) Derivační strom (b) Pozitivní příklad relace (c) Negativní příklad relace

Obr. 1: Příklady derivačních stromů [5]

[5] Dostupné z: http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf

Na příkladu (a) je derivační strom pro větu „*Jane says John heads XYZ Inc.*“. Kořen stromu S_2 zahrnuje obě entity v relaci („*John*“ a „*XYZ Inc.*“) a tudíž dostane označení „pozitivní“, jak je znázorněno na příkladu (b). Na druhé straně příklad (c) je negativním příkladem, protože zde není nejnižší podstrom pokrývající oba subjekty a entita „*Jane*“ není v relaci s ostatními objekty. Každý uzel T je pozitivním nebo negativním příkladem stromu a má tři atributy: role entity (T.role = {osoba, organizace}), typ bloku (T.chunk = {NP, VP}) a text, který obaluje (T.text). Vzhledem k souboru pozitivních a negativních příkladů a funkci vhodného jádra, lze trénování provést užitím klasifikátorů jako „*SVM*“, „*Voted Perceptron*“ a tak dále.

Funkce jádra popsaná v (Zelenko) je modifikací jádra specifikovaného v rovnici (4). Vzhledem ke dvěma mělkým derivačním stromům, jádro vypočítává stejné části mezi stromy. To je analogické k jádrům řetězců, které počítají podobnosti dvou řetězců, pokud jde o počet znakových sub-sekvencí, které mají společné. Jinými slovy, stromové jádro počítá vážený součet podstromů, které jsou společné mezi dvěma mělkými derivačními stromy. Podobnost jader je vypočítávána rekurzivně tímto způsobem:

Pro dva podstromy s kořeny T_1 a T_2, $K(T_1, T_2)$ je dáno:

1. Najdi shodu atributů uzlu T_1 a T_2. Pokud se atributy neshodují, vrať hodnotu podobnosti: 0.
2. Pokud se atributy shodují, přidej hodnotu skóre: 1 a porovnej sekvence potomků T_1 a T_2. Pokud *potomek* (T_1) a *potomek* (T_2) jsou navzájem sekvence potomků T_1 a T_2, jejich podobnost je vypočítána z hlediska počtu společných sub-sekvencí potomků. Proces výpočtu podobnosti sekvencí potomků je analogická s výpočty jader řetězců v rovnici (4).

V případě, že m a n jsou počty uzlů v mělkých derivačních stromech, složitost výpočtu jádra je $O(mn^3)$. Jádra použitá v (Culotta & Sorensen, Dependency tree kernels for relation extraction) jsou velmi podobná jádrům (Zelenko). Unikátní vlastnost jejich práce je, že se užívají stromy závislostí, namísto mělkých derivačních stromů. Každý uzel stromu závislostí obsahuje více informací jako například slovní druh, POS značku, větný člen, typ entity, a tak dále. Argumentem (Culotta & Sorensen) je, že používají bohatší strukturované vyjádření, které vede k významnému zvýšení výkonu ve srovnání s jádrem souborů vlastností.

Shrnuto, (Culotta & Sorensen) a (Zelenko) používají bohaté strukturální informace ve formě stromů seřazených pro dosáhnutí decentního výkonu v úkolu extrakce relací. V kontrastu s (Bunescu & Mooney, a) vytváří zajímavý postřeh, že nejkratší cesta mezi dvěma entitami v rozboru závislostí obsahuje dostatek informací pro extrakci relace mezi těmito prvky. Pokud e_1 a e_2 jsou dvě entity ve větě a p je jejich predikátem, pak nejkratší cesta mezi e_1 a e_2 prochází skrz p. To proto, že e_1 a e_2 jsou argumenty p. Máme větu S. Nejprve extrahujeme strom závislostí. Až poté jsou nejkratší cesty mezi páry entit vypočítány. Řekněme, že jedna cesta může být $P = e_1 \rightarrow w_1 \rightarrow \ldots \rightarrow w_i \leftarrow \ldots \leftarrow \ldots w_n \leftarrow e_2$. Zde w_i jsou slova v nejkratší cestě a šipky indikují směr závislostí extrahovaných ze stromu. Užitím P samotného jako vlastnosti, může vést ke špatnému výkonu kvůli datové řídkosti.

Protože slovní třídy jako větné členy „*POS (part of speech)“* jsou extrahovány z každého slova a jsou vlastnosti vektorů dané kartézským součinem jako:

$$x = [^{e_1}_{C(e_1)}] \times [\rightarrow] \times [^{w_1}_{C(w_1)}] .. \times [\rightarrow] \times [^{w_i}_{C(w_i)}] \times [\leftarrow] .. \times [\leftarrow] \times [^{w_n}_{C(w_n)}] \times [\leftarrow] \times [^{e_2}_{C(e_2)}]$$

Kde $C(w_i)$ jsou třídy slov w_i, které mohou být větným členem, zobecněným větným členem, typem entity a podobně. Jádro nad tímto prostorem vlastností je definováno jako:

$$K(x,y) = \begin{cases} 0 & \text{Pokud } |x| \neq |y| \\ \prod_{i=1}^{|x|} f(x_i, y_i) & \text{Jinak} \end{cases} \quad (5)$$

Kde $f(x_i, y_i)$ je číslo slovní třídy společné pro x_i a y_i. Výhoda tohoto jádra nad jinými navrženými (Zelenko) a (Culotta & Sorensen) je, že jeho výpočet vyžaduje lineární čas ve vlastním zjednodušeném prostoru vlastností. Zatímco v rámci stejného času, výkon extrakce relací se ukazuje jako lepší než Cullotův stromově založený přístup popsán výše. Faktem je, že opětovným zavoláním nejkratší cesty závislostí, je jádro znatelně lepší než Cullotovo stromově založené jádro, pokud máme srovnatelnou přesnost.“ [6]

2.3.3 Shrnutí

„Popsané kontrolované metody extrakce relací jsou hlavně orientované na metody jader. Začínali jsme s definicí jádra řetězce, kde jsme popsali metody založené na variacích jader řetězců pro extrakci relací. Popis jader je založen na vzrůstajícím pořadí dle bohatosti vstupů a korespondujícím vzrůstu komplexnosti jádra. Zatímco jádro souboru vlastností je relativně jednoduché, stromová jádra (Zelenko) a (Culotta & Sorensen) vyžadují bohatší reprezentaci vstupu ve formě mělkého derivačního stromu, stromu závislostí a podobně.

V kontrastu metod jádra, metody založené na vlastnostech popsané v (Kambhatla) a (Zhao & Grishman) používají soubor bezpečně extrahovaných vlastností (heuristik) pro úkol extrakce relací. V problémech vztahujících se k NLP, je často obtížnější generovat optimální reprezentace vlastností ze vstupních dat. Jádra nabízejí vyhovující řešení díky objevováním vstupní implicitnosti v mnohem vyšším dimenzionálním prostoru. Takže používáním jader přesunujeme problém od inženýrství vlastností „*feature engineering*“ k navrhování jader „*kernel design*“. Když tři jádra (Zelenko) a (Culotta & Sorensen) používají strukturovaný vstup a výstupní metody založené na vlastnostech, jsou tyto jádra výpočetně tíživá. Na druhé straně, jádra nejkratší cesty závislostí popsané v (Bunescu & Mooney, a), nabízejí výhodu v lineárním časovém výpočtu podobnosti jádra a relativně lepšího výkonu než (Culotta & Sorensen). S výjimkou (Bunescu & Mooney, a), nikdo

[6] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007

další nevytvořil porovnání výkonu se svou konkurencí. Jádra cesty závislostí se stávají vítězi nad všemi metodami jádra.

Obecně, kontrolované metody mají několik omezení:

- Tyto metody jsou obtížné pro získání nového typu relace z označených dat.
- Výskyty vyšších seřazených relací prvků jsou také složité.
- Metody jsou relativně výpočetně tíživé a nepodporují škálovatelnost s narůstajícím objemem vstupních dat.
- Většina popsaných metod vyžaduje předpřipravená vstupní data ve formě derivačního stromu, stromu závislostí a podobně. Fáze předpřipravení je náchylná k chybám a mohou bránit dobrému výkonu systému.“ [7]

2.4 Polo-kontrolované „*semi-supervised*“ metody

„Viděli jsme, že zde jsou výhody extrakce relací použitím označených dat. Jak tedy polo-kontrolované / „*bootstrapping*“ metody extrakce relací pracují? Polo-kontrolované učení začíná být důležitým tématem počítačové lingvistiky. Mnoho úkolů zpracování přirozeného jazyka zahrnuje extrakce relací. Přirozený jazyk skýtá bohatství v neoznačených datech. Většinou označená data chybí, a je výpočetně příliš drahé vytvářet označená data ve velkých kvantech. Vytváření „*bootstrapping*“ technik je tedy velmi žádoucí.

Zaměříme se na (Yarowsky) a (Blum & Mitchell, Combining labeled and unlabeled data with co-training) algoritmy. Budeme zkoumat použití polo-kontrolovaných metod extrakce relací. Hlavní myšlenkou obou algoritmů je použít výstup slabých nástrojů jako trénovací data pro další iterace. Opakované trénování (Blum & Mitchell) je slabě kontrolované paradigma, které se učí úlohu z malé sady označených dat a z velké množiny neoznačených dat použitím rozdělení, ale s redundantními pohledy (tj. pomocí nesouvislých pod-setů vlastností reprezentující data). Chceme-li zajistit prokazatelné výkonnostní záruky, algoritmus opakovaného trénování předpokládá jako vstup sadu pohledů, které splňují dvě poměrně přísné podmínky. První, že každý pohled musí být dostačující pro učení určitého konceptu. Druhá, že pohledy musí být podmíněně nezávislé na každé jiné zadané třídě.

Obecný rámec algoritmu Yarowsky je uveden na další straně. Yarowsky používá tento algoritmus pro úlohu slovní disambiguace. Yarowskyho algoritmus je ve skutečnosti speciální případ algoritmu opakovaného trénování. (Abney, Understanding the yarowsky algorithm.) prezentuje teoretickou analýzu Yarowskyho algoritmu. Budeme zkoumat rodinu těchto technik, a pokládat si dvě hlavní otázky. 1) Jak získat semínko automaticky? A 2) Co je to dobré semínko?“ [8]

[7] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction,* 2007
[8] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction,* 2007

Algorithm 1 Yarowsky's algorithm in general form (McDonald, 2004)

Input: a set of unlabeled data D and a set of seed examples S
repeat
  Train a classifier C on S
  Label D using C
  N = top n labels that C was highly confident
  $S = S \cup N$
  $D = D \backslash N$
until convergence criteria is reached

Obr. 2: Obecný rámec Yarowskyho algoritmu [9]

2.4.1 DIPRE (Brin)

„DIPRE (Dual Iterative Pattern Relation Expansion) je systém pro extrakci relací navržen v (Brin). Zajímá nás vztah (autor, kniha) na internetu. DIPRE začíná s malým souborem dvojic (autor, kniha), které jsou také semínky. Předpokládejme, že naše první sada semínek obsahuje pouze jedno semínko (Arthur Conan Doyle, The Adventures of Sherlock Holmes). Systém prochází internet a vyhledává stránky obsahující obě instance semínka. Pro učení vzorů DIPRE používá n-tice šesti elementů (pořadí, autor, kniha, prefix, sufix, střed), kde pořadí je 1 pokud se řetězec autora vyskytuje před řetězcem knihy. Jinak je 0. Prefix a sufix jsou řetězce obsahující deseti-znakový výskyt levé nebo pravé strany, střed je řetězec vyskytující se mezi autorem a knihou.

Pro příklad, pokud vyhledávač „*web crawler*“ nalezne stránky, které obsahují:

„*Read The Adventures of Sherlock Holmes by Arthur Conan Doyle online or in your email*“
„*know that Sir Arthur Conan Doyle wrote The Adventures of Sherlock Holmes, in 1982*“
„*When Sir Arthur Conan Doyle wrote the adventures of Sherlock Holmes in 1892 he was high*“

Pak extrahované n-tice jsou:

(0, Arthur Conan Doyle, The Adventures of Sherlock Holmes, Read, online or, by)
(1, Arthur Conan Doyle, The Adventures of Sherlock Holmes, now that Sir, in 1892, wrote)
(1, Arthur Conan Doyle, The Adventures of Sherlock Holmes, When Sir, in 1892 he, wrote)

[9] Dostupné z: http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf

Po extrahování všech n-tic, systém shlukuje n-tice porovnáváním části *pořadí* a *střed*. Pro každou skupinu n-tic, nejdelší společné prefixy nebo sufixy řetězců jsou extrahovány, tedy každá skupina indukuje vzor ve formátu:

(nejdelší společný sufix, nebo prefix řetězců, autor, střed, kniha, nejdelší společný prefix nebo sufix řetězců)

Pro příklad vzor bude vypadat takto (Sir, Arthur Conan Doyle, wrote, The Adventures of Sherlock Holmes, in 1892). Dalším krokem je zobecnění vzoru pomocí výrazu „divoké karty" a vzor nyní bude vypadat (Sir, .*?, wrote, .*?, in 1892). DIPRE použije tento vzor pro prohledání internetu znovu a extrahuje relace. Z instancí extrahuje novou relaci (Arthur Conan Doyle, The Speckled Band). DIPRE přidá novou relaci do sady semínek a zopakuje proceduru znovu, dokud nejsou splněna ukončující kritéria.

Algorithm 2 DIPRE (Brin, 1998)

1. Use the seed examples to label some data
2. Induce patterns from the labeled examples
3. Apply the patterns to data, to get a new set of author/title pairs
4. Return to step 2, and iterate until convergence criteria is reached

Obr. 3: Obecný rámec DIPRE [10]

Jak je DIPRE podobný Yarowskyho algoritmu? Oba algoritmy začínají se sadou semínek příkladů. Klasifikátor použitý v DIPRE porovnává vzory, které jsou trénovány iterativně díky extrahování vzorů z relací semínek. Dostáváme řetězec. Pokud odpovídá jednomu vzoru, pak je řetězec klasifikován jako pozitivní, a je použit pro extrakci nových relací. V opačném případě je negativní. Nové vztahy jsou přidány do sady semínek a jsou znovu použity klasifikátorem, což znamená extrahování více vzorů v DIPRE. DIPRE je instancí aplikace Yarowskyho algoritmu pro extrakci vztahů." [11]

2.4.2 Snowball (Agichtein & Gravano)

„Snowball má podobnou architekturu jako DIPRE. Úkolem je identifikovat vztah (organizace, lokace) na běžný text. Snowball také začíná se sadou semínek relací a připojuje k nim důvěru s hodnotou 1. Klasifikátor ve Snowballu je systém pro porovnávání vzorů jako v DIPRE, avšak Snowball nepoužívá přesnou shodu. Snowball reprezentuje každou n-tici jako vektor a používá funkci podobnosti pro skupinu n-tic. Snowball extrahuje n-tice ve formě (prefix, organizace, střed, lokace, sufix). Prefix, sufix a střed jsou vlastnosti vektorů označených termínů vyskytujících se

[10] Dostupné z: http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf
[11] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007

v páru. Pro příklad, pokud (CMU, Pittsburgh) je známý pár, pak pro řetězec „*... go to CMU campus in Pittsburgh to meet ...*“ systém vyextrahuje:

((*w*1, go), (*w*2, to), ORG, (*w*1, campus), (*w*2, in), LOC, (*w*1, to), (*w*2, meet))

Limit na prefixovou a sufixovou vlastnost vektorů je v tomto případě 2. Každý w_i, je váhový termín, který je počítán díky normalizované frekvenci termínu v daných pozicích.

Pro příklad, váhový termín tokenu „*meet*“ v sufixu je:

$$\text{váha } (meet, \text{suffix}) = \frac{\text{frekvence } meet \text{ v suffixu}}{\text{počet všech slov v suffixu}}$$

Tyto váhové termíny jsou aktualizovány v postupných iteracích čím více n-tic je přidáno. Shrnout n-tice, které mají stejnou (organizace, lokace) reprezentaci, ale odlišný prefix, sufix a střed, Snowball zavádí funkci podobnosti:

$Match(tuple_i, tuple_j) = (prefix_i \,.\, prefix_j) + (sufix_i \,.\, sufix_j) + (middle_i, middle_j)$

Po shlukování n-tic do tříd, Snowball zavádí jeden n-ticový vzor P pro každou třídu, který je těžištěm vektoru n-tice.

Každému vzoru P je přiřazena hodnota důvěry, která měří kvalitu nově navrhovaného vzoru.

$$Confidence(P) = \frac{P_{positive}}{P_{positive} + P_{negative}}$$

Kde $P_{positive}$ je počet, kdy nový vzor objevil pár (organizace, lokace), který byl v předchozí iteraci trénování; a $P_{negative}$ je počet objevených párů, kde organizace byla s odlišnou lokací než v předchozí iteraci.

Pro označení nových dat, Snowball nejdříve spustí rozpoznávání jmenných entit nad daty, pro identifikování všech entit lokací a organizací. V rámci věty, pro každou relaci (organizace, lokace) systém vytvoří n-tici. Proto mají populární páry sadu n-tic asociovaných s původními vzory. Nové vzory jsou indukovány. Systém porovnává každého kandidáta relace se všemi vzory a pouze ponechává kandidáty, kteří mají hodnotu podobnosti větší než je prahová hodnota. Dále Snowball přiřazuje vysokou hodnotu spolehlivosti pro kandidáty vztahů, kde se shodují s více n-ticemi, které

mají vysokou podobnost se vzory, ke kterým má systém důvěru ve formě hodnoty spolehlivosti. Nakonec nový vztah je přidán do souboru semínek a proces je opakován iterativně.

V porovnání s DIPRE, Snowball má flexibilní porovnávací systém. Namísto porovnávání přesných textových řetězců, metriky ve Snowball povolují nepatrné variace v znacích nebo interpunkci.

Co je to dobré semínko? DIPRE tuto otázku zvládá tím, že používá dlouhé vzory, čímž vytváří možnost shody méně pravděpodobnou. Snowball používá nejprve odstranění vzorů, které indukují malý počet n-tic (neproduktivní vzory). Za druhé hodnotu spolehlivosti vzorů. A za třetí hodnotu spolehlivosti na každém semínku.“ [12]

2.4.3 KnowItAll (Etzioni) a TextRunner (Banko)

„Oproti DIPRE a Snowball, KnowItAll (Etzioni) je systém pro zpracování rozsáhlých a rozšiřitelných extrakcí informací z internetu, který označuje své vlastní trénovací příklady použitím malého souboru vzorů z doménově nezávislé extrakce. Pokud je spuštěn pro částečné relace, tyto obecné vzory nesou relačně specifická extrakční pravidla, která jsou dále použita pro učení doménově specifických extrakčních pravidel. Pravidla jsou aplikována na webové stránky identifikované díky vyhledávačům a jejich dotazům. Výsledným extrakcím jsou připojeny pravděpodobnosti použitím bodových vzájemných informací (PMI – Pointwise Mutual Information) odvozených z výsledků vyhledávačů. Pro příklad KnowItAll učí sadu vzorů pro relačně specifické extrakce, například „capital of <country>“, který vede pro extrakci dalších měst.

DIPRE, Snowball a KnowItAll jsou všechny relačně specifické systémy. Soubor zaměřených relací musí být předem pojmenovaný lidskými zdroji. TextRunner (Banko) je navrhnutý pro předcházení tomuto problému. Namísto vyžadování vstupu specifických relací TextRunner učí relace, třídy a entity z textového korpusu ve své kontrolované podobě.

TextRunner definuje relace ve formě n-tice $t = (e_1, r, e_2)$, kde e_1 a e_2 jsou řetězce denotovaných entit nebo podstatných jmen a r je řetězec denotovaných vztahů mezi e_1 a e_2. Systém obsahuje tři komponenty, které jsou 1) vlastní kontrolovaný nástroj pro učení; 2) „single-pass“ extraktor; 3) nadbytečně založený program pro rozhodování.

Nástroj pro učení nejdříve automaticky označí svá vlastní data jako pozitivní nebo negativní, pak použije tato označená data pro trénování binárního klasifikátoru, který je dále použit v extraktoru. Extraktor vygeneruje jeden nebo více kandidátů relací z každé věty. Následně se spustí klasifikátor, který zachovává relace, které již byly označené jako důvěryhodné. Nástroj pro rozhodování přiřadí pravděpodobnost ke každé udržované n-tici založené na pravděpodobnostním modelu redundance popsaný v (Downey, A probabilistic model of redundancy in information extraction).

Nástroje pro rozbory jsou použity pouze jednou, a to ve chvíli, kdy nástroj pro učení provádí své vlastní trénovací procesy pro klasifikátor. Klíčovou myšlenkou systému je, že extraktor extrahuje relace z velkého množství textu bez spuštění procesu rozboru závislostí. Pro příklad, chceme extrahovat relace z „*English Giga*“ slovního korpusu. Namísto spuštění rozborů pro celý korpus,

[12] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007

pouze musíme provést proces rozboru pro jeden milion slov a použít označená data pro trénování binárního klasifikátoru.“ [13]

2.4.4 Shrnutí

„Hlavní nevýhodou DIPRE je jeho systém pro striktní porovnávání vzorů. Pro příklad, dva vzory v DIPRE jsou odlišné, pokud mají rozdíl v pouhé jedné interpunkci. Snowball má jemnější systém pro porovnávání vzorů, to však závisí na programu pro rozpoznávání jmenných entit. V tuto chvíli nejmodernější NER systémy jsou kontrolované systémy (Bikel) a (Finkel) a vyžadují mnoho anotovaných dat pro trénování. To nemusí být problém pro standardní entity jako osoba, organizace nebo lokace, avšak Snowball bude mít problém, pokud budeme chtít rozšíření o nové typy entit nebo vztahů. TextRunner má stejný problém, protože tento systém velmi závisí na rozboru závislostí pro anotování svých vlastních dat pro trénování. V DIPRE může být přidán nový jazyk pro extrakci relací od doby, kdy DIPRE nezávisí na nástrojích pro zpracování přirozeného jazyka, jako jsou nástroje pro rozbory, separátor, NER a další, které jsou trénované pro specifické jazyky.

Dále Snowball, KnowItAll a TextRunner závisí na velkém počtu vstupních parametrů. Definice parametrů poskytují volby pro uživatele pro balancování jejich požadavků od systému. Žádný z těchto systému však nevysvětluje, jakým způsobem můžeme vybírat optimální parametry.“ [14]

2.5 Za oponou binárních relací

„Hlavní podobnost všech zde popsaných systémů je, že se všechny zaměřují primárně na binární relace. Polo-kontrolované systémy jako například TextRunner tvrdí, že jejich systémy dokáží pracovat s n-árními relacemi, ale nejsme si tím příliš jistí kvůli změnám algoritmu, které jsou vyžadovány. Nedávno (McDonald) byl navrhnut framework pro extrakci komplexních relací (n-tic) mezi entitami v textu. Jejich experimenty jsou na extrahování 4-árních relací z bio medického abstraktního textu. V instanci je relace jako seznam entit ($e_1, e_2, \ldots, e_n$), kde e_i je typ entity. Pro příklad nás bude zajímat ternární relace (organizátor, konference, lokace), která závisí na organizátorech konferencí v konkrétní lokaci. Pro větu „ACL-2010 will be hosted by CMU in Pittsburgh“, systém může extrahovat (CMU, ACL-2010, Pittsburgh).

Možná cesta pro extrakci komplexních zaměřených relací může být nejprve seznam všech možných n-tic. Použitím všech n-tic pro trénování binárního klasifikátoru rozlišuje validní instance z nevalidních. Avšak problémem je zde množství možných kandidátů, které roste exponenciálně. Pro příklad, typy relací s n entitami, kde má každý element m možných cest, pak zde je $O(m^n)$ možných komplexních kandidátů relací.

[13] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007
[14] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007

Namísto zkoušení klasifikovat všechny možné relační instance, hlavními myšlenkami (McDonald) jsou:

- Začít s rozeznáváním binárních instancí relací, pokud systém má argumenty požadovaných relací.
- Extrahované binární relace mohou být považovány za hrany grafu entit jako uzly.
- Rekonstrukce komplexních relací díky vytváření n-tic z vybraných maximálních kliků v grafu.

Pro trénování klasifikátoru, se musí nejdříve vytvořit sada všech pozitivních a negativních párů v datech. Pozitivní instance jsou všechny páry, které se vyskytují společně ve validní komplexní relaci v anotovaném korpusu. Negativní příklady jsou instance, které mají prvky, které se nevyskytují společně ve validní relaci. To vede k velké sadě pozitivních a negativních binárních relací. Binární klasifikátor byl trénovaný použitím standardních metod jako maximální entropie a podmíněná náhodná pole.

Po identifikování všech binárních relací je dalším krokem vytvoření grafu entit, kde dvě entity v grafu mají společnou hranu, pokud binární klasifikátor věří, že jsou navzájem v relaci. Jsou konstruovány komplexní relační instance hledáním maximálních kliků. Pro přiřazení hodnoty věrohodnosti pro každou možnou komplexní relaci přiřazují váhu *w(e)* hraně, která je rovna pravděpodobnosti, že dvě entity v *e* jsou v relaci podle klasifikátoru. Váha kliku *w(C)* je definována jako významová váha hrany kliku. Od doby, kdy váhy hran reprezentují pravděpodobnosti, můžeme používat geometrický průměr:

$$w(C) = \left(\prod_{e \in E_C} w(e)\right)^{1/|E_C|}$$

Pokud $w(C) \geq 0{,}5$ pak klik *C* je validní relace. Systém vyhledává všechny maximální kliky o kterých se domnívá, že budou mít $w(C) \geq 0{,}5$. Číslo 0,5 je zvoleno na základě proběhlých experimentů. Problémem jsou exponenciální kliky, které dostaneme, jakmile je graf plně propojen. Používá se ořezávací strategie, která odstraňuje všechny kliky, které mají $w(C) \leq 0{,}5$.

Zde jsou dvě hlavní výhody členění komplexních relací do binárních relací. Za prvé, povoluje použití pro téměř všechny binární relační klasifikátory, které byly studovány a jsou většinou přesné. Za druhé, číslo možných binárních relací je o hodně menší než číslo možných komplexních relací.

Podobně jako Snowball, (McDonald) závisí na rozpoznávači jmenných entit, protože předpokládají, že entity a jejich typy jsou známy. Další podmínkou je, že také arita komplexních relací musí být známa předem. Použité binární klasifikátory jsou založené na vlastnostech. Představili jsme si metody jádra, které jsou velmi výkonné pro klasifikování binárních relací, takže je velmi zajímavé provádět experimenty kombinování metod jádra s touto metodou.“ [15]

[15] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007

2.6 Vyhodnocování extrakce relací

„Vyhodnocování extrakce prvků a relací záleží na typu aplikované metody (kontrolované nebo nekontrolované) a typu použitých dat.

2.6.1 Vyhodnocování kontrolovaných metod

V nastavení kontrolovaných metod jsou extrakce relací vyjádřeny jako klasifikační úkol, a proto jsou metriky „Precision“, „Recall“ a „F-Measure“ použity pro vyhodnocení výkonu. Tyto metriky jsou definovány jako:

$$\text{Precision } P = \frac{\text{Počet správně extrahovaných relací entit}}{\text{Celkový počet extrahovaných relací entit}} \quad (6)$$

$$\text{Recall } R = \frac{\text{Počet správně extrahovaných relací entit}}{\text{Aktuální počet extrahovaných relací entit}} \quad (7)$$

$$\text{F-Measure } F1 = \frac{2PR}{P+R} \quad (8)$$

2.6.2 Vyhodnocování polo-kontrolovaných metod

Pokud chybí označená testovací data, vyhodnocování polo-kontrolovaných metod je mírně odlišnou procedurou ačkoli základní metriky zůstávají stejné („Precision“, „Recall“ a „F-Measure“). Polo-kontrolované metody extrakce relací jsou typicky používané na obrovské množství dat a také výsledky často s objeveným velkým množstvím nových vzorů a relací. Proto je získání přesných měření metrik „Precision“ a „Recall“ obtížné. Malý vzorek získaný náhodně z výstupu je považován za reprezentanta výstupu a manuálně zkontrolován pro aktuální relaci. Dále průměrný odhad přesnosti je vypočítán definicí (6). Tato procedura pro odhadování byla použita v (Brin) a (Banko). Od chvíle, kdy je obtížné získat relace entit z velkého množství dat, je také těžké spočítat metriku „Recall“ pro vyhodnocování polo-kontrolovaných metod.“ [16]

[16] BACK, Nguyen a Sameer BADASKAR. *A Review of Relation Extraction*, 2007

3 Zpracování zdrojových dat

3.1 Zdrojová data

V mé práci byl výběr zdrojových dat striktně omezen na anglickou verzi Wikipedie. Samozřejmě jsou k dispozici další velikostně i obsahově podobné databáze článků a informací jako například DBpedia, Freebase, GeoNames, Getty a podobné. Výhodou je, že databáze Wikipedie není nijak zaměřená a obsahuje články a informace jakýchkoli typů.

Wikipedie je mnohojazyčná webová encyklopedie s otevřeným obsahem. Jejím cílem je volné šíření encyklopedických informací. Wikipedie existuje ve více než 270 jazykových verzích různého rozsahu. [17]

Největší zastoupení článků a příspěvků je samozřejmě v anglickém jazyce. Jedná se tedy o nestrukturované texty, které je třeba zpracovat a použít rozpoznávač jmenných entit pro získání strukturovaných označených testovacích dat.

Archiv databází Wikipedie nabízel několik desítek verzí. Nejdříve byla použita verze ze září 2014. Později byla použita jedna z posledních verzí Wikipedie z března 2015. Databáze je strukturovaná do XML formátu obvykle po jednotlivých článcích. Samotný zdrojový soubor s kompletní Wikipedií měl velikost přibližně 50 GB a obsahoval více než 15,3 miliónů článků.

3.2 Možnosti zpracování zdrojových dat

Pro potřeby dalšího zpracování jsem potřeboval zdrojová data zpracovat. Důležitým zamyšlením bylo jakým způsobem zpracovat data. Zdrojový soubor obsahoval značky XML, nadbytečné wiki formátování, nedůležité informace nebo také určité značky a entity jazyka HTML. Z těchto všech informací bylo úkolem vyextrahovat pouze ty relevantní informace, a to ve formě čistého textu. V ideálním případě navíc ve formátu, se kterým by se nadále dobře pracovalo.

3.2.1 Vlastní program

První možností, která mne ze začátku zaujala nejvíce, neboť mne nenapadaly lepší metody, byl návrh, že si na to napíšu skript, kde si mohu sám definovat, jaké konkrétní části textu chci profiltrovat a získat tak čistý textový formát.

3.2.2 Program pro převod zdroje do vertikálního textu

V rámci skupiny projektů „Decipher wikipedia“ patřící výzkumné skupině znalostních technologií vedené Doc. RNDr. Pavlem Smržem, Ph.D. již byl vytvořen program, který také dokáže filtrovat nežádoucí prvky, hromadně zpracovat kompletní zdrojovou databázi Wikipedie a produkovat články ve formátu vertikálního textu. Kromě toho také zastává program rozpoznávání jmenných entit. Formát v prvním sloupci vždy obsahoval jediné slovo, interpunkční znaménko nebo speciální

[17] Wikipedie. *Wikipedie: Otevřená encyklopedie*, 2001

formátovací značky pro oddělení jednotlivých článků, odstavců vět a podobně. Ve druhém sloupci pak jsou u jmenných entit nebo jejich částí URI odkazy na konkrétní článek na Wikipedii, který se k dané jmenné entitě vztahuje. Jednotlivé sloupce jsou od sebe odděleny tabulátorem – jedná se tedy o datový formát TSV (Tab-Separated Values).

Skript očekává na standardním vstupu XML soubor s Wikipedií. Na chybový výstup vypisuje průběžné hlášení o průběhu zpracování programu. Na STDOUT pak po řádcích vypisuje už extrahovaná data. Výstupy jsem vhodně přesměroval pro získání jediného souboru, který obsahuje výstup programu a průběžně zobrazoval, jak program pokročil. Program předem zjišťuje počet článků a průběžná hlášení doplňuje o údaj pravděpodobného času dokončení.

3.2.3 Wikipedia Extractor

Tento program generuje ze zdrojových dat Wikipedie obyčejný text ve formátu: titulek a text článku. Každý článek ohraničený formátovacími značkami pro rozlišení, kde článek končí a začíná další. Výstupem programu je defaultně jediný soubor obsahující všechny extrahované texty. Program je původně určen pro italskou verzi Wikipedie a autoři nikde neuvádějí, zda program funguje stejně dobře i na zdrojová data Wikipedie v anglickém jazyce. Program také umí vytvářet a komprimovat výstupní soubory o zadané velikosti a spoustu dalších funkcionalit. Dostupná je také verze pro vícevláknové zpracování. [18]

3.3 Zpracování zdrojových dat

Vydal jsem se cestou vytvoření vlastního programu v přesvědčení, že je to pro mne nejoptimálnější. Nejdříve jsem celý vstup rozdělil pomocí programu CSPLIT na 26 320 částí, kde každá část měla 30 000 řádků. Napsal jsem jednoduchý program v jazyce Python 3, který převáděl text na použitelnou textovou formu a odstranil prakticky všechno nepotřebné a nadbytečné a takto ze zdrojového souboru vyextrahoval 8 000 článků v jednotlivě členěných souborech. Skript přijímá dva povinné a jeden nepovinný parametr. První „`--input`“ specifikuje cestu ke zdrojovým souborům. Druhý parametr „`--number`“ udává počet zpracovaných článků. Poslední argument je nepovinný „`--type`“, který zadáním slova „*treetagger*“ výstupní soubory upraví do formátu pro zpracování programem TreeTagger. Skript na svém vstupu vezme soubor a extrahuje z něho první článek ohraničený XML značkami „`<page>`“ a „`</page>`“. Tento článek je následně přefiltrován. Výstup se ukládá do složky „*output*“. V případě přítomnosti třetího parametru do složky „*treetagger*“. Každý tento soubor obsahoval na prvním řádku titulek, a pod ním samotnou textovou část. Měl jsem pocit, že počet získaných článků bude dostačující jakožto reprezentování vzorku dat.

V té době jsem však ještě nevěděl kolik článků Wikipedie obsahuje a tudíž jsem si neuvědomoval, že osm tisíc těžko může jakkoli reprezentovat „vzorek dat“ 15,3 miliónů článků. Brzy jsem však zjistil, že tento stav není ideální. Zvláště když jsem následně použil lingvistické nástroje, tudíž jsem měl brzy 4 x 8 000 různých výstupů a v každém byly jiné informace.

Začal jsem znovu a tentokrát zkusil program pro převod zdroje Wikipedie do vertikálního textu. Program běžel více než 36 hodin, ale zpracoval všechny články. Výstupní soubor také měl

[18] BAUGH, Wesley. *Wikipedia Extractor*, 2015

o 20 GB méně než původní XML soubor Wikipedie. Na první pohled se zdálo, že je výstupní soubor obsahově v pořádku. Bohužel jsem zjistil, že výstup programu pro převod do vertikálu není příliš kvalitní a obsahuje spoustu chyb způsobených převodem. Příkladem byly neodstraněné HTML direktivy a tagy, části řetězců pro zápis odkazů v HTML („ahref=http://...cz/“). Nebo také celé úseky vět či odstavců v jednom řádku, kde slova byla oddělena podtržítky. Tyto chyby se vyskytovaly v podstatě v celém výstupním souboru. Chyby se však špatně hledaly vzhledem k rozsáhlosti textu. Pokud jsem na chybu zrovna nenarazil náhodou, nejčastěji jsem tyto chybové elementy hledal pomocí náhodných filtrování příkazu GREP.

Takto nekvalitní výstupní materiál samozřejmě nemělo smysl dále zpracovávat. Na vyzkoušení posledního zmíněného nástroje jsem neměl dostatek času. Bylo třeba chybné řádky pokud možno maximálně odfiltrovat a získat co nejlepší možnou verzi výstupu. Pro filtrování jsem naprogramoval drobný program v Pythonu 3, který pomocí sady regulárních výrazů nalezené chyby odstraňoval do nového výstupního souboru. Samotný běh programu vzhledem k velikosti vstupního souboru opět trval 2 – 3 dny což se negativně podepsalo na celém plánu zpracování mé bakalářské práce. Bohužel se také občas stávalo, že se zpracování programu přerušilo například ve dvou třetinách. Až později jsem začal používat program SCREEN, který problém náhlého přerušení odstranil. Nejméně třikrát jsem nalezl v již odfiltrované verzi další chyby. Program jsem pokaždé obohatil o filtrování nově nalezených chyb a spustil filtraci znovu. Takto jsem strávil nad tímto krokem více než 14 dní jen proto, abych opravil chybný výstup použitého programu pro převod. Nakonec jsem měl výstupní soubor, který také obsahoval ještě nějaké chyby, ale můj skript pro filtraci dokázal odfiltrovat bezmála další 10 GB chybných dat.

3.3.1 Přidání odkazů k nadpisům a zájmenům

Program, který převedl zdrojová data do vertikálního textu, také automaticky do druhého sloupce ke jmenným entitám doplnil URI odkaz na konkrétní články na Wikipedii. Odkazy na články bylo třeba také doplnit k titulkům, tedy nadpisům, článků ve vertikálním textu. Tento drobný úkol jsem se rozhodl vyřešit jednoduchým skriptem, který vyhledává strukturní značky označující hlavičku. Mezi těmito značkami se vyskytovaly nadpisy článků. K nim do druhého sloupce vertikálního formátu skript přidá odkazy vytvořené z nadpisu, kde mezery nahradí podtržítky. Program zpracovával takové kvantum informací opět další 2 – 3 dny.

V článcích o osobách se vždy používala přivlastňovací zájmena, která ve většině případů správně identifikovala předmět článku. Tedy například, pokud byl článek o umělci a v textu článku byla použita nějaká varianta přivlastňovacího zájmena „*her, his, ...*“, pak se jednalo o toho samého umělce. Bylo třeba doplnit ke všem tvarům těchto zájmen také odkaz na článek, pro provázání významů zájmen a anotací ze znalostní databáze. To platilo také k zájmenům, pomocí kterých se označovalo místo. Například „*this, there, here*“, která v článku o nějakém místě označovala právě toto místo.

3.4 Zpracování vertikálního textu

Po získání slušné podoby vertikálního textu s odkazy na články Wikipedie bylo ještě třeba na tento datový objekt použít programy pro získání lingvistických informací a rozpoznávání jmenných entit

a s tím spojené doplnění anotací k jménům a názvům napříč celým vertikálním textem. Výstup těchto programů je nutné formovat do tvaru, který bude snadno použitelný pro indexaci.

Formát obsahuje tyto sloupce informací:

Position, Token, Tag, Lemma, Parpos, Function, Parword, Parlemma, Paroffset,
Nerid, Nertag,
Param0, Param1, Param2, Param3, Param4, Param5, Param6, Param7, Param8, Param9

Kde prvních devět jsou informace CONLL formátu. Tedy lingvistické údaje o jednotlivých slovech, indexy, na kterých se slovo nachází, slovní druh, větný člen a podobné informace. Následují dva speciální sloupce, kde první je id entity ze znalostní databáze a druhý parametr je typ entity (osoba, lokace, umělec, událost nebo umělecké dílo). Posledních devět parametrů je vyplněno informacemi ze znalostní databáze a to dle typu entity:

- Osoba: URL, obrázek, jméno, pohlaví, místo narození, datum narození, místo úmrtí, datum úmrtí, profese, národnost.
- Umělec: URL, obrázek, jméno, pohlaví, místo narození, datum narození, místo úmrtí, datum úmrtí, role, národnost.
- Místo: URL, obrázek, název, země.
- Umělecké dílo: URL, obrázek, jméno, forma, datum začátku vytváření díla, datum ukončení vytváření díla, umělecký směr, žánr.
- Událost: URL, obrázek, název, datum začátku konání, datum ukončení konání, místo.

3.4.1 Vyplnění CONLL sloupců

Pro získání dat ke každému slovu je třeba spustit lingvistické nástroje, které analyzují slovo a doplní k němu požadované informace – jazykové údaje o slově. Pro tyto analýzy slov v anglickém jazyce existují dostupné programy, které dokážou automaticky zpracovat jakékoliv slovo.

TreeTagger

„TreeTagger je nástroj nezávislý na jazyce pro anotování textu informacemi jako větný člen a slovní druh. Tento nástroj byl naprogramován Helmutem Schmidem v rámci TC projektu na Ústavu pro počítačovou lingvistiku na univerzitě ve Stuttgartu. TreeTagger byl úspěšně použitý pro označení textu v jazycích: němčina, angličtina, francouzština, italština, holandština, španělština, bulharština, ruština a další. Program je snadno přizpůsobitelný do dalších jazyků, pokud je k dispozici slovník a manuálně označený trénovací korpus.“ [19]

MDParser

[19] SCHMID, Helmut. TreeTagger: - a language independent part-of-speech tagger.

„MDParser zastává program vícejazyčného rozboru závislostí a daty řízený systém, který může být použit k analýze textů libovolného jazyka, pro který jsou k dispozici trénovací data. Rozbor umožňuje vytváření obou – neoznačených i označených struktur závislostí. Počet možných typů relací závisí na rozmanitosti tréninkových dat.

Modely systému jsou založené na mnoha vlastnostech, které jsou extrahovány ze slov ve větě zahrnujících slovní druhy a větné členy. Proto, abychom mohli zpracovat nejdříve neoznačený text MDParser navíc obsahuje některé komponenty předzpracování:

- Rozdělovač vět, aby rozbor mohl vytvořit struktury závislostí pro individuální věty.
- Program pro rozdělení vět na slova, aby rozpoznal prvky mezi relačními závislostmi.
- Program pro označování pro zjištění lingvistických informací k danému slovu.

MDParser je obzvláště rychlý systém a proto je zvláště vhodný pro zpracování velkého množství dat. Také může být použit jako součást pro větší aplikace, které potřebují tuto funkcionalitu.

MDParser byl již testován na několika jazycích, včetně němčiny a angličtiny. V současné době je možné dosáhnout velmi skvělých výsledků, když uvážíme, že je program založen na rychlém lineárním klasifikačním přístupu a deterministické syntaktické analýze.“ [20]

Zpracování

Vstupem TreeTaggeru musí být vertikální text, který obsahuje pouze jedno slovo na řádek. Já měl však k dispozici datový objekt, který byl dvousloupcový a u některých entit právě ve druhém sloupci obsahoval URI adresu. Kromě těchto informací se na samostatných řádcích také vyskytovaly strukturální značky „<doc>, <head>, <p>, <s>, <g/>“, které označovaly začátky a konce článků, odstavců, vět a podobně. Od Ing. Kouřila jsem dostal k dispozici programy pro zpracování vertikálního textu právě programy TreeTagger a MDParser. Oba programy už počítaly s výskytem strukturálních značek, a proto bylo velmi vhodné je použít. První program si ukládá pozice značek, slovo pošle do programu TreeTagger na označení a k výstupu TreeTaggeru připojí značky do speciálních sloupců. Program pro MDParser funguje na podobném principu. Do programů bylo třeba doplnit také ukládání URI adres z druhého sloupce vertikálu. URI jsem jen prostě schoval do jednoho ze speciálních sloupců.

Dosud uvedené postupy v této kapitole však byly velmi problémové. Často vznikaly problémy z důvodu špatné funkčnosti získaných programů a skriptů. Také vznikaly problémy kvůli době zpracování, která se stále prodlužovala. Zvláště, pokud třídenní zpracování skončilo nezdarem. To mne dostalo do velké časové tísně, proto jsme se s Doc. Smržem dohodli, že skončím se zpracováním dat přibližně v těchto místech. Zbylé zpracování dat a samotnou indexaci provedl Ing. Kouřil a já se mohl zaměřit na samotný proces extrakce relací. V několika následujících kapitolách teoreticky popíšu, jakým způsobem bych postupoval ve zpracování dat a indexaci a jaké bych měl možnosti, kdyby dosud uvedené postupy fungovaly podle plánu.

[20] VOLOKH, Alexander a Günter NEUMANN. *MDParser*, 2014

3.4.2 Vyplnění sloupců anotovaných informací

Vyplnění zbývajících sloupců bych zajistil pomocí programu SECAPI a znalostní databáze. Všechny tyto produkty rovněž pochází z projektů výzkumné skupiny znalostních technologií. Znalostní databáze obsahuje anotované jmenné entity extrahované v rámci projektu Decipher NER. SECAPI poskytuje aplikační rozhraní pro komunikaci s touto databází a je dostupné na většině fakultních serverů. Konkrétně rozhraní nabízí službu „`get_entity_by_uri`“, která dokáže ze znalostní databáze vytáhnout anotované informace dle zadané URI, které mám ke jmenným entitám uložené. Výsledkem by byl krátký skript, který by procházel řádky a nacházel URI u jmen a názvů. URI by pak posílal do SECAPI a namísto této adresy by do tohoto a dalších sloupců dosadil hodnoty vrácené SECAPI. Toto by byl poslední krok pro vytvoření výše uvedeného více než dvacetisloupcového formátu, vhodného pro další zpracování například indexaci.

4 Indexace

Protože jsem věděl, že budu potřebovat mít někde data uchována a ideálně v takové podobě, abych v nich mohl rychle a jednoduše vyhledávat, bylo vhodné vybrat nějaký způsob naindexování dat. Vyhledávání v obyčejném textovém souboru by bylo náročné a zdlouhavé. Nejlepší by tedy bylo mít data uložená v indexu a mít k dispozici již implementovaný vyhledávací full-textový nástroj pro vytvoření ideálních podmínek pro vytváření extrakčních vzorů a také pro systém pro extrakci relací, který bude popsán níže.

Samotný pojem indexace se dá definovat jako: „Proces vyjádření obsahu dokumentu pomocí prvků selekčního jazyka, obvykle s cílem umožnit zpětné vyhledávání.“ [21]

4.1 Možnosti indexace

Možnosti indexování se omezily jen na systémy, které jsou s volnou licencí pro použití a ideálně již běží implementované na některém ze školních serverů. Hlavní výběr se tedy týkal jen systémů ElasticSearch a MG4J, které již jsou na školních serverech používány pro podobné účely.

4.1.1 ElasticSearch

„Základním stavebním kamenem systému ElasticSearch je knihovna Lucene. „Lucene je výkonná open source knihovna pro fulltextové vyhledávání implementovaná v Javě. Mezi vývojáři je ceněná především pro svoji vyspělost a širokou komerční i komunitní podporu. Je to „*low-level*“ knihovna, přímé použití není pro každého. Nabízí velmi detailní, nízkoúrovňové API, ale řadu problémů spojených s implementací rozsáhlejších systémů neřeší.

Fulltextové vyhledávače ukládají data do invertovaných indexů. Aby Lucene mohl vytvořit invertovaný index, potřebuje vstupní text rozdělit na jednotlivá slova a případně tato slova dále upravit (například převést na základní tvar). Tento proces se nazývá analýza a obsahuje:

- Analyzér – sestává z „*char filtrů*“, jednoho tokenizéru a několika „*token filtrů*“.
- „*Char filtr*“ – může měnit, přidávat nebo odebírat jednotlivé znaky vstupního textu. Jeho vstupem i výstupem je proud znaků.
- Tokenizér – je zodpovědný za rozdělení vstupního textu (proudu znaků) na jednotlivá slova. V Lucene se pro „*slovo*“ užívá označení „*token*“ nebo „*term*“ – v závislosti na kontextu. Výstupem tokenizéru proud tokenů.
- „*Token filtry*“ – přicházejí na řadu jako poslední. Každý vstupní token je postoupen jednotlivým token filtrům v předem definovaném pořadí. Token filtry mohou vstupní token nějak zpracovat či modifikovat. Výstupem token filtru může být žádný, jeden či více tokenů.“ [22]

[21] Indexace. *KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV)*, 2014
[22] VLČEK, Lukáš. *Elasticsearch: Vyhledáváme hezky česky*, 2013

4.1.2 MG4J

„MG4J (Managing Gigabytes for Java) je volný full-textový vyhledávač pro kolekce obrovských dokumentů naprogramovaný v Javě. MG4J je vysoce přizpůsobitelný, výkonný, plnohodnotný vyhledávač poskytující nejmodernější funkce (jako například BM25 / BM25F hodnocení) a nejnovější vyzkoumané algoritmy.

Hlavními body MG4J jsou:

- Propracované indexování. Podpora kolekcí dokumentů a továrniček umožňuje analyzovat, indexovat a dotazovat konzistentně velký počet kolekcí dokumentů. Poskytuje jednoduché rozhraní, které vysvítí důležité pasáže ve vyhledaných dokumentech.
- Efektivita. MG4J může indexovat bez úsilí TREC GOV2 kolekce (dokumentové továrničky jsou poskytovány pro tento účel) a zvětšují se do stovek miliónů dokumentů.
- Multi-indexový sémantický interval. Když zadáte dotaz, MG4J vrátí pro každý index seznam intervalů odpovídající dotazu.
- Výrazové operátory. MG4J jde mnohem dál než je model „balík slov“ a poskytuje efektivní implementaci frázových dotazů, přibližovací omezení, seřazené konjunkce a kombinované mnohonásobné dotazy. Každý operátor je reprezentován vnitřně jako abstraktní objekt, takže můžete jednoduše použít vaši nejoblíbenější syntaxi.
- Flexibilita. Můžete vytvářet velmi malé záznamy zadáním pozic termů nebo počtu termů. Je to jen na vás. Různé odlišné typy kódů mohou být vybrány pro balancování efektivnosti a velikosti indexu.“ [23]

4.2 Naindexování dat

Nejdříve jsem uvažoval, že data budu indexovat ve službě ElasticSearch. Neboť jsem měl s tímto vyhledávačem už předchozí zkušenosti. Později, jsme se však s Doc. Smržem domluvili, že bude použitý MG4J. Pro indexaci dat v MG4J bych nejdříve aktuální soubor obsahující sloupec s jednotlivými slovy a další sloupce s připojenými informacemi z TreeTaggeru, MDParseru a znalostní databáze, upravil do podoby indexovatelné systémem MG4J. Následně bych se domluvil s Ing. Kouřilem, kde bych si ve spuštěném systému MG4J na fakultním serveru mohl založit vlastní index a spustit ideálně paralelní indexování. Použity by pravděpodobně byly nástroje Ing. Kouřila, který s tímto již má zkušenosti a vlastní implementované nástroje. Pokud by takové nástroje nebyly dostupné, nezbývalo by mi nic jiného než si nastudovat proces indexace do MG4J a postarat se o naindexování článků Wikipedie vlastními silami.

[23] SEBASTIANO, Vigna. *MG4J: Managing Gigabytes for Java*

4.3 Spuštění indexu a příprava pro vyhledávání

MG4J jako služba běží na většině školních serverů. Kolekce naindexovaných dokumentů jsou také rozděleny na různých serverech. Ing. Kouřil data naindexoval a poskytl mi přístup ke grafickému uživatelskému rozhraní běžící na portu 8081 na serveru athena2. Také mi dal k dispozici textové rozhraní pro dotazování indexu z terminálu. Dotazování na první pohled fungovalo výborně. Později jsem však zjistil, že například dotazování konkrétního jména mi systém napoprvé najde výskytů 5. Při druhém dotazování jen 4. Třetí našel už pouze jeden a na počtvrté žádný. Zjistil jsem, že index zkrátka po nějaké chvíli používání z neznámého důvodu padá. Konzultoval jsem tento problém s Ing. Kouřilem, který mi pokaždé službu restartoval, ale také neměl žádné tušení, proč systém přestává fungovat. Vždy jsem vyhledával jen jména nebo jiná obyčejná slova. Protože s vyhledávačem pracovali i jiní lidé, nechtěl mi Ing. Kouřil poskytnout přístupy pro restartování služby z mé strany.

Domluvili jsme se na duplikování služby, která poběží na jiném portu a bude přístupná pouze mě. Budu si tedy moci službu libovolně restartovat a spouštět, kdykoliv bude vyhledávač MG4J nedostupný. Ing. Kouřil tedy pro mne upravil vyhledávač i indexátor, aby pro mne bylo co nejjednodušší vyhledávač spouštět. Nebylo z mé strany nutné nic editovat ani překládat. Obdržel jsem dva skripty, které automaticky službu spustí / zastaví. Vyhledávač přístupný tedy výhradně mě běží na serveru athena2 na portu 12 001. Pro přístup jsem měl opět webové rozhraní na portu 8080 a textové rozhraní.

4.4 Vytvoření statistik

Jedním z drobných podúkolů této práce bylo vytvoření statistik počtu výskytů vět obsahujících vyhledávané výrazy vybraných typů vztahů. Tyto statistiky by pak byly použity pro porovnání s výsledky extrakcí. Naprogramoval jsem ještě před zjištěním situace s vyhledávačem, skript v Pythonu 3, kterému se zadávají dva parametry. První „`--relation`“ přebírá vstupní soubor, který obsahuje dvojice platných vztahů. Na každém řádku je, pro příklad relace autor ovlivnil jiného autora, jméno a příjmení prvního autora a bílým znakem oddělené jméno a příjmení druhého autora. Druhý parametr „`--output`“ specifikuje výstupní soubor se statistikami. Program po řádcích prochází entity v relaci a pomocí textového rozhraní pro dotazování MG4J nechá dvojici vyhledat. Výsledek pak sečte a vypíše do výstupního souboru.

Bohužel po zjištění stavu vyhledávače bylo nemožné získat smysluplné statistiky, neboť jsem nevěděl, zda-li je vrácená hodnota počtu výskytů opravdu správná – vzhledem k výše uvedenému různému počtu výsledků vrácené MG4J při opakovaném dotazování. A dále také s nekontrolovanými pády systému se zpracování skriptu vždy stalo po pádu zbytečným, kdy pak dotazováním MG4J odpovídal, vždy nula výskytů. Dalo by se sice vždy například skript upravit, aby se vždy začalo znovu na místě, kde se začíná poprvé vyskytovat nula. Stále by ale statistiky byly neprůkazné.

5 Extrakce relací z Wikipedie

Ze všech postupů a metod popsaných v kapitole 2. jsem si vybral polo-kontrolovanou metodu postavenou na DIPRE (kap. 2.4.1.). Tedy proces extrakce probíhá tak, že se nejdříve začíná s malým souborem dvojic binárních platných relací - semínek. V indexu článků Wikipedie se vyhledají tyto dvojice a z nalezených výskytů se extrahují n-tice (pořadí, první prvek dvojice, druhý prvek dvojice, prefix, sufix, střed). Prefix a sufix jsou maximálně desetiznakové řetězce před nebo za prvky dvojice. Střed je řetězec mezi subjekty dvojice. Tyto n-tice se shlukují podle pořadí a středu. Každá skupina vyprodukuje vzor ve formátu n-tice (nejdelší společný prefix nebo sufix, první prvek dvojice, střed, druhý prvek dvojice, nejdelší společný prefix nebo sufix). Následuje nahrazení prvků pomocí výrazu divoké karty. Tato n-tice se použije pro vyhledání dalších výskytů dvojic. Tyto dvojice se přidají k sadě semínek a proběhne celý postup v mém případě ještě jednou. To by mělo postačovat k určení úspěšné / neúspěšné extrakce. Vyhledávač automaticky viditelně označuje, která slova úspěšně porovnal s vyhledávanými výrazy. Na příkladech níže jsem tato slova zvýraznil tučným řezem textu.

Pro každou relaci jsem měl k dispozici seznam platných vztahů, ze kterých jsem většinou náhodným způsobem vybíral semínka. Jak jsem již zmiňoval, neměl jsem možnost získat statistiku četností výskytů jednotlivých dvojic, musel jsem tedy náhodně zkoušet vyhledat určité dvojice a na základě výsledků zvolit jestli semínko použít nebo ne.

V MG4J mohu použít různé operátory pro skládání dotazů a podobně. Také jsem použil místo vyhledání divoké karty služeb anotovaných článků, konkrétně možnosti vyhledání „*upřesněného*“ výrazu divoké karty. Tedy specifikaci co se má vyhledávat. V indexu jsou anotované články o osobách, umělcích, místech, uměleckých dílech a událostech. Je možno tedy vyhledávat například jen osoby dosazením výrazu „*nertag:person*“ nebo místa pomocí „*nertag:location*“. Toho jsem využil hlavně při vytváření vzorů, kde tímto dosáhnu přesnějších výsledků. Kromě toho vyhledávání je „*case-sensitive*“ a je tedy třeba dodržovat velikosti písmen.

Zadávání slov a operátorů do vyhledávání MG4J můžu kombinovat s operátory pro určení posloupnosti jednotlivých částí dotazu. Vyhledávač však nevyhledává přímo věty, ale dokáže vrátit jako jeden výsledek například i celý článek, který může skýtat i 1 000 vět. Ve většině případů jsem musel vyhledávání upravit pomocí operátoru pro zadání maximálního rozsahu slov v mezi elementy dotazu. Velikost rozsahu jsem volil na základě experimentování, tedy do jaké míry mohou být části dotazu rozptýleny, aby vrácené výsledky byly relevantní a použitelné.

Vzhledem k tomu, že v indexu jsou anotována i zájmena osob, vyhledávač vrací výsledky pro vyhledání osob nebo umělců i v rámci spojených anotací se zájmeny. Tedy například vyhledání „*nertag:person*“ vrací všechna jména osob, ale zároveň také všechny výsledky, kde se vyskytují zájmena spojená s osobami. Na druhou stranu vyhledání například konkrétního jména vyhledá pouze výskyty jména a už nedokáže vyhledat zájmena, ke kterým je konkrétní jméno připojeno. Nelze tedy vyhledávat jednoduše a přímo, ale je třeba také experimentovat s kombinacemi vyhledávání „*nertag:person*“ a konkrétními jmény nebo jinými obecnými pravidly. Například při vyhledávání semínek jsem musel často právě takto kombinovat, abych získal relevantní výsledky.

Se zájmeny, také přichází na řadu chybovost vyhledávání. Přivlastňovací zájmena můžou kontextově znamenat úplně něco nebo někoho jiného než jsme chtěli vyhledat. Pro příklad chceme vyhledat nějakou osobu, která je například botanikem. Vyhledávač vrátí, kromě jiných, také výsledek

s větou, kde je „*... his father, the botanist ...*“. Zájmeno „his“ je spojeno se jménem, které však nemusí patřit člověku, který je botanikem. Bohužel nemůžeme automaticky určovat, kdy zájmeno představuje osobu, kterou v kontextu opravdu hledáme. V ukázkách níže jsem zájmena ve výsledcích úmyslně nahrazoval jmény pro demonstraci získání správných výsledků.

Dále musíme brát v úvahu také chybovost již zmíněných nástrojů na zpracování a označování textu. Není výjimkou a je velmi často běžné, že jsou některé jmenné entity označeny špatně, nejsou označeny vůbec nebo jsou nějakým způsobem rozděleny a každá část je označena jako jiná. Pro příklad si vezměme třeba jméno slavné herečky „*Lucy Liu*“. Chybovost nástrojů může slovo „*Lucy*“ úplně vyloučit jako jmennou entitu a druhou část „*Liu*“ označí jako název města. Tedy vyhledání jména vrátí následně výsledek, který obsahuje město Liu.

5.1 Extrakce relace: „ARTIST – has influenced - ARTIST“

Relace pokrývá vztah umělců, kteří ovlivnili jiné umělce. Pro tento vztah již existuje seznam autorů, kteří ovlivnili jiné autory. Tento seznam byl vytvořen v rámci projektu Decipher_ner a obsahuje 11 281 platných dvojic této relace. Z tohoto seznamu jsem vybral tato semínka:

- Paul Gauguin, Edvard Munch
- Tristan Tzara, Samuel Beckett
- Michelangelo, Peter Paul Rubens
- Melozzo da Forlì, Michelangelo
- Rudolf Steiner, Joseph Beuys

Vyhledávání této dvojice bylo obtížné nejen vzhledem k chybám uvedeným výše, ale také, protože nebylo výjimkou, že se v jedné větě vyskytovalo zájmeno nebo jméno umělce dvakrát a celé to tak zapadalo do vyhledávaných výrazů. Příkladem může být vrácený výsledek: „*he was influenced by his trip to great Wall of China*“, kde sice je, že byl umělec ovlivněn, ale už v kontextu nesouhlasí, že byl ovlivněn jiným umělcem. Ve větě tedy je, že byl umělec ovlivněn svým výletem a to už v kontextu nesouhlasí s vyhledávaným výrazem.

5.1.1 Nalezení výskytů a vytvoření n-tice

Nalezené věty:

„*... Prague and visited Paris in 1906, where **Edvard Munch** was influenced by the work of **Paul Gauguin**. In 1903, he became a ...*“

„*...farces such as The Bald Soprano. **Tristan Tzara**'s poetry influenced **Samuel Beckett** (who translated some of it into English ...*“

„*...second was completed in 1602. **Peter Paul Rubens** was heavily influenced by **Michelangelo**. He was introduced to his work ...*“

„*...Marco Melozzo." The paintings of **Melozzo de Forlì** strongly influenced **Michelangelo**, Raphael and Donato Bramante. ...*“

„...the Camphill movement) are widespread. ***Rudolf Steiner****'s paintings and drawings influenced* ***Joseph Beuys*** *and other modern artists. His two ...*"

Vytvořené n-tice:

(0, Paul Gauguin, Edvard Munch, where, EMPTY, was influenced by the work of)
(1, Tristan Tzara, Samuel Beckett, EMPTY, who, 's poetry influenced)
(0, Michelangelo, Peter Paul Rubens, EMPTY, EMPTY, was heavily influenced by)
(1, Melozzo da Forlì, Michelangelo, of, Raphael, strongly influenced)
(1, Rudolf Steiner, Joseph Beuys, EMPTY, and other, 's paintings and drawings influenced)

5.1.2 Vytvoření vzorů

Z první a třetí n-tice:

Vzor: (.*? , nertag:artist , was .*? influenced by , nertag:artist , .*?)
Dotaz: ((.*) < nertag:artist < was < (.*) < influenced < by < nertag:artist < (.*))~50

Z druhé, čtvrté a páté n-tice:

Vzor: (.*? , nertag:artist , influenced , nertag:artist , .*?)
Dotaz: ((.*) < nertag:artist < influenced < nertag:artist < (.*))~30

5.1.3 Nalezení nových výskytů

Zadání prvního dotazu vrátí 181 nových výsledků. Při zadání rozsahu nižším než 50 bylo výsledků velmi málo. Při vyšším zase chybné relace výrazně vzrostly. I přesto, při rozmezí 50 slov můžeme vidět častou chybovost v zájmenech. Také se vyskytuje chybovost, že v jedné větě se vyskytuje umělec, v další pak začíná kontext věty například „*Maria's choice was influenced*" a další umělec. Ovlivnění v tomto případě je opět chybné.

Příklady správných hledání:

„... par with ***Ivan Aivazovsky****, and was influenced by the work of* ***Arkhip Kuindzhi*** *..."*

„... ***Wilkes*** *was nominated for the Turner Prize for her show at the Milton Keynes Gallery. This included her sculpture, Non-Verbal Installation, influenced by Lazarus Breaks His Fast, a painting in 1927 by* ***Walter Sickert****. ..."*

„... ***Keith****'s own band, Trax, however, was not as popular. Forsey was obviously being influenced by* ***Moroder*** *..."*

Příklady špatných hledání:

„... persecution when ***Giovanni Canti*** *was a child. In the 1970s and 1980s he was influenced by Jazz and Blues, which shows in* ***his****..."*

*„...****Antonio Ruffo****'s son or grandson, Mattia (il giovane) Stomer (1649-1702) , also was a painter. Style Stom was influenced by the Baroque painter* ***Caravaggio*** *and his followers ..."*

Zadání druhého dotazu vyhledá 7 944 nalezených výskytů. Rozmezí 30 slov postačilo k získání velkého počtu výsledků a zachování slušné hranice chybovosti. Chyba se zájmeny je zde však stále velká.

Příklady správných hledání:

*„… that **Cézanne**'s own attempts to paint the nude were heavily influenced by **Balzac**'s portrayal …“*

*„… **John Dyer Baizley** style is heavily influenced by the famous artist **Alphonse Mucha.** In …“*

Příklady špatných hledání:

*„… primeval forest. **Karel Smirous**'s sense of art composition was also influenced by **his** mother, …“*

*„… is possible that **Le Corbusier** was influenced in **his** choice of pseudonym by the name …“*

*„… **Antoine Samuel Adam Salomon** influenced **his** brother-in-law, the Austrian physician, inventor and politician …“*

5.2 Extrakce relace: „LOCATION NAME – HISTORICAL NAME“

Vztah se zaobírá historickým pojmenováním míst (města, státy) a jejich novými názvy vzhledem k historickému přejmenování. Seznam pro relaci názvů geografických míst a jejich historické názvy jsem získal ze stránky (http://en.wikipedia.org/wiki/Geographical_renaming). V tomto seznamu je přibližně 300 platných vztahů. Zvolil jsem tato semínka:

- Peking, Beijing
- Ceylon, Sri Lanka
- Bombay, Mumbai
- Upper Volta, Burkina Faso
- Titograd, Podgorica

5.2.1 Nalezení výskytů a vytvoření n-tic

Nalezené věty:

„*…of China in 1949, the capital of **Peking** was renamed **Beijing** and the university was consequently renamed as Beijing …*“

„*…part of the administration of the island of **Ceylon** (now known as **Sri Lanka**) under the successive European colonial powers, …*“

„*…married Banu. He died in **Bombay** (now **Mumbai**), Maharashtra, India on 26 March …*“

„*…widow of Thomas Sankara, the President of **Upper Volta** (later renamed **Burkina Faso**) from 4 August 1983 until his assassination …*“

„*…born January 18, 1987 in **Titograd**, now **Podgorica**) is a Montenegrin pop singer …*“

Vytvořené n-tice:

(1, Peking, Beijing, capital of, and the, was renamed)
(1, Ceylon, Sri Lanka, island of, under the, now known as)
(1, Bombay, Mumbai, he died in, Maharashtra, now)
(1, Upper Volta, Burkina Faso, of, from 4, later renamed)
(1, Titograd, Podgorica, 1987 in, is a, now)

5.2.2 Vytvoření vzorů

Z první, druhé a čtvrté n-tice:

Vzor: (of , nertag:location , renamed , nertag:location , .*?)
Dotaz: (of < nertag:location < renamed < nertag:location < (.*))~10

Z třetí a páté n-tice:

Vzor: (in , nertag:location , now , nertag:location , .*?)
Dotaz: (in < nertag:location < now < nertag:location < (.*))~10

Z druhé, třetí a páté n-tice:

Vzor: (.*? , nertag:location , now , nertag:location , .*?)
Dotaz: ((.*) < nertag:location < now < nertag:location < (.*))~10

5.2.3 Nalezení nových výskytů

První dotaz vrátí 187 výsledků při rozsahu 10 slov. Větší rozsah měl za následek výrazný nárůst chybných výsledků. Jak jsem zjistil, jako místa jsou v indexu označeny i budovy, památky, nemocnice, mosty a podobné. Vztah historické jméno místa a nové jméno místa se tedy rozrostl ještě o tyto subjekty. Svou roli zde však také sehrálo chybné označení jmenných entit popsaných výše. Také se vyskytovaly chyby v kontextech. Například firmy, které mají ve svém jméně název nějaké lokace, a které byly přejmenovány na nový název, který odpovídá zase jméně jiného místa.

Příklady správných hledání:

*„... Soffia & Cía. of **Valparaíso**, and renamed the **Flora**. August Oetzmann, a former ...“*
*„... became the heart of the city of **Tiflis** (renamed **Tbilisi** in 1936). The land ...“*
*„... also of **Fort Washington**, and renamed the **Sequoia**. In 1929 it was ...“*
*„... the west became the town of **Ross' Landing**, later renamed **Chattanooga**. Archaeological findings ...“*

Příklady špatných hledání:

*„... Cia de Transportes Sylvia S.A. of **Panama** and renamed **Jasa**. She was ...“*

Druhý dotaz – 12 691 výsledků. Překvapivě dobré výsledky. Chyby se vyskytují jen v rámci toho, že dané místo nebylo přímo přejmenováno, ale jen změnilo vlastníka.

Příklad správných hledání:

*„ ... de León and Florencia Gregorio in **Polo** (now **Valenzuela**), Bulacan. Later, de ...“*

*„ ... van Inlandsche Artsen in **Batavia** (now in **Jakarta**). He completed his education ...“*

Příklady špatných hledání:

*„ ... into social structure in **India** and what is now **Pakistan**. Parts 5 through ...“*

*„ ... on the outskirts of London, in **Romford**, now part of **Havering**. His family were ...“*

*„ ... on Ephraim's farm in **Alvarado** (now part of **Union City**). After his ...“*

Poslední dotaz nezávisle na rozsahu slov vrací pouze chybné výsledky, které v téměř v žádném případě neodpovídají relaci historické jméno míst – nové jméno místa. Bohužel vzhledem také k chybovosti označených jmenných entit. Poměrně často se vyskytuje jméno lokality, které je chybně označeno jako jméno osoby. Tento vzor můžeme označit jako špatný a ideálně vůbec nezařazovat do seznamu vzorů pro extrakci tohoto typu relace.

Příklady správných hledání:

*„ ... which was completed in 1468. **Wade** was now known as **Wadebridge**. The bridge was a strategic ...“*

Příklady špatných hledání:

*„ ... artist from the United States. **Butler** now resides in **Rochester**, Minnesota. Amateur ...“*

*„ ... with the Eielsen Synod. **Immanuel** is now an independent **Lutheran** church. Please ...“*

*„ ... and Michael S. Holland. **Gordon** is now the **Wyoming** state treasurer. In the general ...“*

*„ ... songs in the rock style. **Rojhan** is now living in **Istanbul**. His most famous ...“*

5.3 Extrakce relace: „PERSON – egyptologist“

Zde se používá vztah osob a jejich zaměření jakožto egyptologové. Tento vztah je zastoupen seznamem na stránce „*List of Egyptologist*“ (http://en.wikipedia.org/wiki/List_of_Egyptologists) a obsahuje přes 200 vztahů. Použil jsem semínka:

- Jan Assmann, egyptologist
- Arthur Weigall, egyptologist
- Kim Ryholt, egyptologist
- Thomas Young, egyptologist
- Toby Wilkinson, egyptologist

5.3.1 Nalezení výskytů a vytvoření n-tic

Nalezené věty:

„*...examples are the philosopher Jürgen Habermas, the* ***egyptologist Jan Assmann****, the sociologist Niklas Luhmann, the historian ...*“

„*...alcoholism. This condition is attested by* ***Arthur Weigall****, an* ***egyptologist****, who associated with, and conversed extensively ...*“

„*...Qur, and Qal. The* ***egyptologist Kim Ryholt*** *equates Qareh with the prenomen Khawoserre, which ...*“

„*...father of the physicist, physician and* ***egyptologist*** *Dr* ***Thomas Young****) to lease a farm and water mill ...*“

„*...Hor's existence was debated, with* ***egyptologist Toby Wilkinson*** *contesting the reading and signification of his name ...*“

Vytvoření n-tic:

(0, Jan Assmann, egyptologist, the, the, EMPTY)
(1, Arthur Weigall, egyptologist, by, who, an)
(0, Kim Ryholt, egyptologist, the, equates, EMPTY)
(0, Thomas Young, egyptologist, and, to lease a, Dr)
(0, Toby Wilkinson, egyptologist, with, contesting, EMPTY)

5.3.2 Vytvoření vzorů

Zde jsem musel zanechat druhou část dvojice „*egyptologist*“ pro zajištění vyhledávání pouze dalších egyptologů.

Z první a třetí n-tice:

Vzor: (the , egyptologist , .*? , nertag:person , .*?)
Dotaz: (the < egyptologist < nertag:person < (.*))

Z první, třetí a páté n-tice:

Vzor: (.*? , egyptologist , .*? , nertag:person , .*?)
Dotaz: ((.*) < egyptologist < nertag:person < (.*))

Z druhé n-tice:

Vzor: (by , nertag:person , an , egyptologist, who)
Dotaz: (by < nertag:person < an < egyptologist < who)~10

Ze čtvrté n-tice:

Vzor: (and , egyptologist , Dr , nertag:person , to lease)
Dotaz: (and < egyptologist < Dr < nertag:person < (to < lease))

5.3.3 Nalezení nových výskytů

První dotaz nalezne 75 výsledků. Drtivá většina výsledků je správných a indukuje nové osoby, které byly egyptology. Občas se vyskytuje chybovost se zájmeny. Například „*his father was egyptologist*“. Nebo také chybovost v kontextu: „*he wanted to become an egyptologist, but he become a politician*“. Samozřejmě také chyby v rámci pojmenování jmenných entit.

Příklady správných hledání:

*„… list is "Netjerkare", the German **egyptologist Ludwig Stern** has proposed in 1883 that Netjerkare …“*

*„… existence was proposed by the French **egyptologist Pierre Montet**, who found in Tanis few monuments …“*

*„… work of Polish archaeologist and **egyptologist** Prof. **Kazimierz Michałowski**, who established in 1959 the Research …“*

Příklady špatných hledání:

*„… excavation of the site was undertaken by Maspero, a French **egyptologist**, in 1882. **Pepy I Merire**'s work as later continued …“*

*„… the Pharaoh's name references him. Tut was first voiced by Jeannie Elias, followed by Maryke Hendrikse then finally, by Donna Cherry. Cleo Carter A 12 year old African American girl who wants to become an **egyptologist**, because **Donna Chery**'s father …“*

Další dotaz vyhledá 77 výsledků. Správnost i chybovost je podobná jako u prvního vzoru.

Příklady správných hledání:

*„… from any access to available women. The German **egyptologist Georg Steindorff** explored the Oasis in 1900 and reported …“*

*„… The 5th Earl of Carnarvon, the famous **egyptologist** for whom **Howard Carter** discovered the tomb of Tutankhamun, commenced …“*

*„… travellers, but it was not until 1908 that the first **egyptologist**, **Herbert Winlock**, visited Dakhla Oasis and noted its monuments in some …“*

Příklady špatných hledání:

*„… In the same year, and together with **egyptologist** Boris Alexandrovich Turayev and linguist **Nikolay Yakovlevich Marr**, …“*

*„… of the 19th century French archeologist and **egyptologist** - Paul Pierret believes that the word serer means "**Issa Laye Thiaw** traces the temple. …“*

Třetí a čtvrtý dotaz jsou natolik specifické, že naleznou pouze původní výsledky, ze kterých byly vzory vytvořeny. Nejsou proto dobrými a použitelnými vzory a do seznamu vzorů pro extrakci této relace se nezařadily.

5.4 Extrakce relace: „PERSON – was educated at – SCHOOL“

Dalším vztahem jsou osoby, které studovali na určité škole nebo univerzitě. Pro tento druh vztahu jsem nenalezl žádný přímý seznam nebo datovou sadu. Použil jsem seznam osob, které vystudovali na škole „*Stonyhurst College*“. Seznam je dostupný na stránce Wikipedie (http://en.wikipedia.org/wiki/List_of_people_educated_at_Stonyhurst_College) a nabízí více než 230 platných vztahů, ze kterých byla vybrána semínka:

- Ambrose Rookwood, Stonyhurst College
- Baron Chitnis, Stonyhurst College
- Charles Waterton, Stonyhurst College
- Chris Morris, Stonyhurst College
- Edward Bulfin, Stonyhurst College

5.4.1 Nalezení výskytů

Nalezené věty:

„*... **Ambrose Rookwood** was educated at **Stonyhurst College**, and having studied law under Charles Butler ...*“

„*...and Lucia Mallik, **Baron Chitnis** was born in London and educated at **Stonyhurst College**. ...*“

„*...and Anne Bedingfield. **Charles Waterton** was educated at **Stonyhurst College** in Lancashire where his interest in exploration and ...*“

„*...**Chris Morris** was educated at **Stonyhurst College**, a Jesuit boys' boarding independent school ...*“

„*...Lord Mayor of Dublin in 1870. **Edward Bulfin** was educated at **Stonyhurst College**, and then at Kensington Catholic Public School ...*“

Vytvoření vzorů:

(1, Ambrose Rookwood, Stonyhurst College, EMPTY, and having, was educated at)
(1, Baron Chitnis, Stonyhurst College, Mallik, EMPTY, was born in London and educated at)
(1, Charles Waterton, Stonyhurst College, EMPTY, in, was educated at)
(1, Chris Morris, Stonyhurst College, EMPTY, a Jesuit, was educated at)
(1, Edward Bulfin, Stonyhurst College, EMPTY, and then, was educated at)

5.4.2 Vytvoření vzorů

Školy a univerzity jsou v MG4J označeny jako „*Museum*“.

Ze všech n-tic:

Vzor: (.*? , nertag:person , was educated at , nertag:museum , .*?)

Dotaz: ((.*) < nertag:person < (was < educated < at) < nertag:museum < (.*))

5.4.3 Nalezení nových výskytů

Pro tento jediný vzor vyhledávač nalezne 107 562 výsledků. Ve většině případů správné výstupy. Příčinou chybovosti jsou opět zájmena, chyby špatného označení jmen a názvů a případně chyby v kontextu podobné jako u relací výše.

Příklady správných hledání:

*„… **Ali** was educated in mechanical engineering at **Middle East Technical University** in Ankara. …“*

*„… and a Bishop of Llandaff. **Tyler** was educated at **Magdalen College**, Oxford. He held …“*

Příklady špatných hledání:

*„… as a Liberal. The son of **Gilbert Browning**, he was born in St. John's and was educated there, at Upper Canada College and at the **University of Glasgow**. Browning was …“*

*„… **Prince Phillip Duke of Edinburgh** was educated at Royal High School, Edinburgh and the **University of Edinburgh**. …“*

*„… life **Frederick Lawton** was educated at Battersea Grammar School and at **Corpus Christi College**, Cambridge. He served briefly …“*

5.5 Extrakce relace: „PERSON – was born – LOCATION“

Poslední relace má význam osob, které se narodily na konkrétním místě (město, stát). Seznam jsem získal z volně dostupné datové sady platných vztahů, kterou mi poskytnul Doc. Smrž (https://drive.google.com/drive/#folders/0BxLcVibFf8eHV3RhVU1tTlVKcVU). Konkrétně datová sada „*20130403-place_of_birth.json*“. Sada obsahuje 9 566 vztahů. Zvolena byla semínka:

- Jean Cassou, Deusto
- Jeff Evans, South Wales
- Keith Hellawell, Kirkburton
- Mark Powell, Texas
- Martin Wright, Germany

5.5.1 Nalezení výskytů a vytvoření n-tic

Nalezené věty:

„*...in Paris. Biography **Jean Cassou** was born at **Deusto**, near Bilbao, (Spain). ...*“

„*...about television. Biography **Jeff Evans** was born 1960 in **South Wales** and studied languages at the University of Reading ...*“

„*...Biography Early life **Keith Hellawell** was born 18 May 1942 in **Kirkburton**, near Huddersfield, Yorkshire, UK. ...*“

„*...American symphony and opera conductor. **Mark Powell** was born in the west **Texas** town of Big Spring and received his musical ...*“

„*...**Martin Wright** (born 23 June 1974 in **Germany**) is a British bobsledder who competed for ...*“

Vytvořené n-tice:

(1, Jean Cassou, Deusto, Biography, near, was born at)
(1, Jeff Evans, South Wales, Biography, and, was born 1960 in)
(1, Keith Hellawell, Kirkburton, early life, near, was born 18 May 1942 in)
(1, Mark Powell, Texas, EMPTY, town of, was born in the west)
(1, Martin Wright, Germany, EMPTY, is a, born 23 June 1974 in)

5.5.2 Vytvoření vzorů

Ze všech n-tic:

Vzor: (.*? , nertag:person , born , nertag:location , .*?)
Dotaz: ((.*) < nertag:person < born < nertag:location < (.*))~50

Z první a třetí n-tice:

Vzor: (.*? , nertag:person , was born , nertag:location , near)
Dotaz: ((.*) < nertag:person < (was < born) < nertag:location < near)~50

Z první a druhé n-tice:

Vzor: (Biography , nertag:person , was born , nertag:location , .*?)
Dotaz: (Biography < nertag:person < (was < born < in) < nertag:location < (.*))~50

Z druhé, třetí a čtvrté n-tice:

Vzor: (.*? , nertag:person , was born .*? in , nertag:location , .*?)
Dotaz: ((.*) < nertag:person < (was < born < in) < nertag:location < (.*))~50

5.5.3 Nalezení nových výskytů

První vzor najde 716 257 výsledků. Správnost výskytů je velmi vysoká. Chybné výsledky jsou způsobeny stále stejnými chybami popsanými výše. Příklad kontextové chyby: „*his idea was born in Maryland*“.

Příklady správných hledání:

„ ... of Islamic Studies, Cambridge University. ***Josef W Meri*** *was born in* ***Chicago*** *in 1969 and comes from a ...“*

„ ... on July 1. Early life ***Marvin Lee Pelton*** *was born on September 27, 1950, and grew up in* ***Wichita****...“*

„ ... professional boxer who competed in the 1920s. ***Jean Delarge*** *was born in* ***Liège****. Delarge won the gold medal in ...“*

„ ... of the 19th century Daoguang Emperor. ***Yuzhan*** *was born in* ***Dalian****, Liaoning in 1923. He had a keen interest ...“*

Příklady špatných hledání:

*„ ... as Martin (****BR****) CJ to eliminate confusion. Education and pre-Northern Territory Chief Justice Martin was born in* ***Lithgow****, New South Wales in ...“*

Druhý vzor vyhledá 15 901 výsledků. Úspěšnost je zde ještě lepší než u prvního vzoru. Můžeme zde však narazit na chyby v zájmenech a zároveň v kontextu, typu: „*John was not born there*“. Také dostáváme některé výsledky, kde nelze jednoznačně určit zda-li a kde se přesně dotyčná osoba narodila.

Příklady správných hledání:

„ ... and education In December 23, 1907 ***Donald B. Lindsley*** *was born in* ***Brownhelm****, Ohio, a small farming community ...“*

„ ... who played for Sussex. ***Kenneth Scott*** *was born in* ***Uckfield****, Sussex and died during the Second World War ...“*

„ ... and choral conductor. Early years ***Kosta Manojlović*** *was born in* ***Krnjevo*** *near the town of Velika Plana on ...“*

Příklady špatných hledání:

„ ... ***John C. Merrick*** *was born into slavery as the son of a white man and a former slave in 1859. He grew up in* ***Raleigh*** *and Chapel Hill, two cities near Durham, and ...“*

„ ... ***Alan Clark*** *represents Hot Spring County and parts of Garland, Grant, and Saline counties. Background Clark was born to* ***Alfred*** *Eugene Clark (born 1938) and Mildred I. Clark (born 1940), ...“*

Třetí vzor má téměř 100 % správnost a nalézá 36 144 výskytů. Slovo „Biography“ je na Wikipedii úzce spojeno pouze s osobami a proto lze jen velmi těžko najít nějaký špatný výskyt. Chyby se vyskytují jen v rámci špatně pojmenovaných jmen.

Příklady správných hledání:

„ ... Biography The first of three children, ***Francisco Tudela*** *was born in* ***Lima*** *into an upper-class family. ...“*

„ ... Biography Rivka Menashe (later ***Riki Gal****) was born in 1950 to an Orthodox family in the poor neighborhood of* ***Mea Shearim*** *in Jerusalem. ...“*

„ ... cardinal and bishop. Biography ***Girolamo di Corregio*** *was born in* ***Correggio****, Emilia-Romagna in 1511, the son ...“*

„ ... Minister Hamadi Jebali. Biography Early life and career ***Abderrahman Ladgham*** *was born on October 18, 1947 in* ***Le Bardo****, Tunisia. He attended the ...“*

Příklady špatných hledání:

„ ... Biography Family origins García Fernández, founder of the House of ***Villamayor****, was probably born in the 1170's near* ***Villaldemiro*** *...“*

Poslední vzor nalézá 577 840 výsledků a tvoří je z části množina výsledků prvního vzoru. Poskytuje ale také nové. Chyby jsou podobné jako u prvního vzoru.

Příklady správných hledání:

„ ... the Legislative Assembly of Manitoba. ***Alexander Black*** *was born in* ***Île-à-la-Crosse****, the son of Samuel Black ...“*

„ ... at the end of 1985. ***Jochen Rindt*** *was born in* ***Germany*** *during World War II. His parents were killed ...“*

Příklady špatných hledání:

„ ... college football at ***Florida State****. Early years Carradine was born in* ***Cincinnati****, Ohio. He attended ...“*

„ ... seat of Narrogin. The son of ***Crawford Nalder****, who later served as the state's Deputy Premier, Nalder was born in* ***Wagin****, a small town in ...“*

„ ... family's dog Sui, Sue. According to ***Terri Irwin****'s father, Bindi is an Australian Aboriginal word that means "young girl." Biography Bindi Irwin was born in* ***Buderim****, Queensland. ...“*

5.6 Vyhodnocení extrakce vztahů

Každá kapitola s extrakcí skončila nalezením nových výskytů, ale co dál? Následoval by proces získání n-tic z jednotlivých nalezených výsledků. Z těchto n-tic by se dále vytvořili vzory a z těchto vzorů by se dále experimenty získaly vyhledávací dotazy. Ty by posloužily pro vyhledávání nových výskytů. Takhle by to pokračovalo pořád do kolečka než by byly splněny ukončovací podmínky.

Ukončovacími podmínkami by mohly být například:

- Dosažení maximálního počtu vzorů pro každý vztah.
- Žádný další vzor by už nedokázal najít nový (nebo vůbec nějaký) výskyt.

Jak jsme si ale mohli všimnout, celý proces je až příliš náchylný k chybám. Nejprve zde máme masivní chyby způsobené nedostatečným rozpoznáváním jmenných entit. Což je vzhledem k tomu, že celý proces je založený na hledání jmen a názvů osob, míst, událostí a dalších entit, velmi kritické. Dále možnost vyhledat osoby, místa a jiné entity, v článku jen pomocí zájmen je velmi

užitečné. Na druhou stranu, nemožnost zpětně vyhledat jméno tak, aby vyhledávač správně porovnal zájmeno, které je anotováno tímto jménem, a vrátil tento výsledek, hravě anuluje původní kladnou myšlenku. Navíc vzniklé chyby v kontextu popsané výše, ještě tento klad dokáže obrátit v zápor, protože těchto chyb se při hledání objeví skutečně mnoho. Nepočítaje zájmena, také musíme brát v úvahu chyby v kontextu, které se také nevyskytují zřídkakdy. Pravděpodobně bychom při dalším hledání a hloubání narazili na další typy chyb. Avšak tyto čtyři typy chyb se občas i ve své kombinaci vyskytují běžně napříč celým vyhledáváním relací a indexovanými články Wikipedie.

Vzhledem k množství chyb a různých typů je nemožné nějak automaticky zjistit přesné měření úspěšnosti. Vyhledávač vždy vrátí nějaký počet výsledků z něhož zobrazí jen prvních 20 výsledků. Pro představu jsem ručně zpracoval počet správných a špatných výsledků na těchto prvních 20 výstupů a zpracovat statistiky pravděpodobné úspěšnosti.

5.6.1 Vyhodnocení relace: „ARTIST – has influenced - ARTIST“

Počet vzorů: 2.

Vzor 1. (.*? , nertag:artist , was .*? influenced by , nertag:artist , .*?)

Počet nalezených výskytů:	181
Počet správných výskytů:	9
Počet špatných výskytů:	11
Pravděpodobný počet správných výsledků:	81
Pravděpodobný počet špatných výsledků:	100
Procentuální podíl správných výsledků:	45 %
Procentuální podíl špatných výsledků:	55 %

Vzor 2. (.*? , nertag:artist , influenced , nertag:artist , .*?)

Počet nalezených výskytů:	7 944
Počet správných výskytů:	10
Počet špatných výskytů:	10
Pravděpodobný počet správných výsledků:	3 972
Pravděpodobný počet špatných výsledků:	3 972
Procentuální podíl správných výsledků:	50 %
Procentuální podíl špatných výsledků:	50 %

5.6.2 Vyhodnocení relace: „LOCATION NAME – HISTORICAL NAME“

Počet vzorů: 3.

Vzor 1. (of , nertag:location , renamed , nertag:location , .*?)

Počet nalezených výskytů:	187
Počet správných výskytů:	13
Počet špatných výskytů:	7
Pravděpodobný počet správných výsledků:	122
Pravděpodobný počet špatných výsledků:	65
Procentuální podíl správných výsledků:	65 %
Procentuální podíl špatných výsledků:	35 %

Vzor 2. (in , nertag:location , now , nertag:location , .*?)

Počet nalezených výskytů:	12 691
Počet správných výskytů:	10
Počet špatných výskytů:	10
Pravděpodobný počet správných výsledků:	6 346
Pravděpodobný počet špatných výsledků:	6 346
Procentuální podíl správných výsledků:	50 %
Procentuální podíl špatných výsledků:	50 %

Vzor 3. (.*? , nertag:location , now , nertag:location , .*?)

Počet nalezených výskytů:	75
Počet správných výskytů:	0
Počet špatných výskytů:	20
Pravděpodobný počet správných výsledků:	0
Pravděpodobný počet špatných výsledků:	75
Procentuální podíl správných výsledků:	0 %
Procentuální podíl špatných výsledků:	100 %

5.6.3 Vyhodnocení relace: „PERSON – egyptologist“

Počet vzorů: 4. Statistiky provedeny jen pro první dva. Zbývající nalezly jen samy sebe.

Vzor 1. (the , egyptologist , .*? , nertag:person , .*?)

Počet nalezených výskytů:	75
Počet správných výskytů:	16
Počet špatných výskytů:	4
Pravděpodobný počet správných výsledků:	60

Pravděpodobný počet špatných výsledků: 15
Procentuální podíl správných výsledků: 80 %
Procentuální podíl špatných výsledků: 20 %

Vzor 2. (.*? , egyptologist , .*? , nertag:person , .*?)
Počet nalezených výskytů: 77
Počet správných výskytů: 17
Počet špatných výskytů: 3
Pravděpodobný počet správných výsledků: 65
Pravděpodobný počet špatných výsledků: 12
Procentuální podíl správných výsledků: 85 %
Procentuální podíl špatných výsledků: 15 %

5.6.4 Vyhodnocení relace: „PERSON – was educated at – SCHOOL“

Počet vzorů: 1.

Vzor 1. (.*? , nertag:person , was educated at , nertag:museum , .*?)
Počet nalezených výskytů: 107 562
Počet správných výskytů: 8
Počet špatných výskytů: 12
Pravděpodobný počet správných výsledků: 43 025
Pravděpodobný počet špatných výsledků: 64 537
Procentuální podíl správných výsledků: 40 %
Procentuální podíl špatných výsledků: 60 %

5.6.5 Vyhodnocení relace: „PERSON – was born – LOCATION“

Počet vzorů: 4.

Vzor 1. (.*? , nertag:person , born , nertag:location , .*?)
Počet nalezených výskytů: 716 257
Počet správných výskytů: 15
Počet špatných výskytů: 5
Pravděpodobný počet správných výsledků: 537 193
Pravděpodobný počet špatných výsledků: 179 064
Procentuální podíl správných výsledků: 75 %
Procentuální podíl špatných výsledků: 25 %

Vzor 2. (.*? , nertag:person , was born , nertag:location , near)

Počet nalezených výskytů:	15 901
Počet správných výskytů:	15
Počet špatných výskytů:	5
Pravděpodobný počet správných výsledků:	11 926
Pravděpodobný počet špatných výsledků:	3 975
Procentuální podíl správných výsledků:	75 %
Procentuální podíl špatných výsledků:	25 %

Vzor 3. (Biography , nertag:person , was born , nertag:location , .*?)

Počet nalezených výskytů:	36 144
Počet správných výskytů:	18
Počet špatných výskytů:	2
Pravděpodobný počet správných výsledků:	32 530
Pravděpodobný počet špatných výsledků:	3 614
Procentuální podíl správných výsledků:	90 %
Procentuální podíl špatných výsledků:	10 %

Vzor 4. (.*? , nertag:person , was born .*? in , nertag:location , .*?)

Počet nalezených výskytů:	577 840
Počet správných výskytů:	17
Počet špatných výskytů:	3
Pravděpodobný počet správných výsledků:	491 164
Pravděpodobný počet špatných výsledků:	86 676
Procentuální podíl správných výsledků:	85 %
Procentuální podíl špatných výsledků:	15 %

6 Systém pro extrakci relací

Částí této práce je navrhnutí a implementace systému pro automatickou extrakci relací. Vzhledem k výše popsaným chybám a komplikacím není možné vytvořit plně funkční automatický systém, který by dokázal kompletní proces extrakce sám s úspěchem zvládnout. Pokud by byl k dispozici propracovanější nástroj pro rozpoznávání jmenných entit, pak by se o tom dalo uvažovat. Když se ale podíváme na jednotlivé extrakce zjistíme, že téměř každý krok je třeba kontrolovat a ověřovat správnost uživatelem. Některé kroky dokonce vyžadují uživatelovo experimentování pro získání relevantních výsledků. Rozhodl jsem se systém pro extrakci relací ponechat z části na uživateli a nabídnout mu systém, který nahradí alespoň část z celého procesu extrakce relací. Navrhnul jsem program, který by se dal označit jako soubor drobných funkcí podporující tento proces extrakce.

S tímto programem vypadá proces extrakce takto:

- Uživatel použije semínka a experimenty je vyhledá v grafickém webovém rozhraní vyhledávače MG4J. Použitelné správné výskyty uživatel manuálně zkopíruje a uloží do souboru.
- Uživatel použije systém pro proces získání n-tic, vytvoření vzorů a návrh vyhledávacích dotazů.
- Uživatel experimentuje se vzory navrhnutými systémem ve vyhledávači MG4J. Z výsledků uživatel manuálně extrahuje nová semínka.
- Opakování procesu, dokud nejsou splněna ukončovací kritéria stanovená uživatelem.

Konkrétně systém zastupuje kompletní proces získání n-tic z výskytů indexu. Z vytvořených n-tic systém vytváří vzory, na jejichž základě se vytváří návrhy na vyhledávací dotazy. Uživatel spustí program, který ve smyčce očekává příkazy uživatele. Všechny ovládací příkazy jsou uvedeny v příloze Manuál – Program pro podporu extrakce relací. Celým postupem programu může uživatel projít zadáním jednoho příkazu. Nebo může uživatel procházet postupem po jednotlivých etapách. Kdykoliv může uživatel nechat nahrát ze souboru nové výskyty, vytvořit n-tice, vytvořit vzory, vytvořit dotazy a té vytisknout data z každého uvedeného kroku na obrazovku nebo uložit do souborů. Program také umožňuje vyčištění dat uložených v systému.

7 Závěr

Cílem této práce bylo seznámit se se zpracováním rozsáhlých textových korpusů, s možnostmi indexace a anotování informací a s metodami extrakce relací. Jednotlivými částmi si projít a hlouběji proniknout do těchto problematik. Podrobně vyzkoušet a extrahovat vybrané typy relací. Na základě zkušeností s extrakcemi navrhnout systém pro automatickou extrakci relací.

Části této práce mi rozhodně nabídly opravdu široký rozhled problematiky a zamyšlení nad jednotlivými kroky. Při zpracování desítek a stovek GB informací již musí programátor uvažovat o každém kroku programu, který s tímto masivním objemem dat pracuje, protože zvláště tady lze citelně poznat dobu zpracování, která se mnohdy protahuje na dny. Dále pohled na indexování obecně nejen z pohledu databází, ale také z pohledu samotného systému indexace. Velmi zajímavé bylo uvědomit si důležitost rozpoznávání jmenných entit a také problémů, vzniklých špatným rozpoznáním. Podrobné zkoumání metod extrakce mne uvedlo do tématu, které je v tuto chvíli mezi informatiky, zkoumajícími oblast extrakce informací a relací, velmi lukrativní a ne příliš probádané. Dokazuje to i zásadní nedostatek materiálů v českém jazyce. Konečné informace zjištěné samotným průběhem extrakce různých typů relací a problémů s nimi spojených, byly také více než zajímavými faktory. Navrhnutí systému a zprovoznění už jen záviselo nad důkladným zamyšlením nad stavem indexovaných článků a vybraným procesem extrakce relací.

Mým hlavním přínosem v tomto tématu bylo zjištění a upozornění na možné chyby spojené s extrakcemi relací. Jakým způsobem může zpracování dat použitím nekvalitních a neodladěných programů, ještě před indexací, ovlivnit veškeré další zpracování a proces extrakce relací. Jako další přínos můžeme započítat navrhnutý program pro podporu vybrané metody extrakce. Automatický proces je vzhledem k vzniklým problémům nereálný. Alespoň částečná podpora dokáže pokrýt část extrakce, která je více méně nezávislá na stavu indexovaných dat.

Z pohledu dalšího vývoje projektu by se nejdříve musel projekt vrátit do fáze rozpoznávání jmenných entit a tento program vylepšit do maximální možné úspěšnosti. Špatně rozpoznaná jména a názvy mají kritický dopad na celý projekt. Po znovu zpracování dat, ideálně nejnovějšího staženého zdroje Wikipedie, tato data znovu naindexovat. Vytvořený program pro podporu extrakce relací by se mohl rozšířit do podoby automatického systému. Stále by však zůstal faktor polo-kontrolované vybrané metody extrakce, kdy by na správnost výsledků musel dohlížet uživatel. Také by jistě bylo zajímavé podrobně projít jinou metodou extrakce relací a implementovat systém i pro tuto metodu pro možnost porovnání výsledků.

V tomto oboru prozatím příliš dokončených projektů není. Můžeme se ale setkat se systémy uvedenými v teoretické části této práce. Pro mnou vybranou metodu je k dispozici systém DIPRE, který údajně funguje velmi dobře díky velmi důkladné analýze a rozpoznávání jmenných entit. Jako příklady dalších systémů můžeme uvést příklady jako Snowball, KnowItAll a podobné.

Literatura

BACK, Nguyen a Sameer BADASKAR. 2007. *A Review of Relation Extraction* [online]. Pittsburgh, 13 s. [cit. 2015-05-12]. Dostupné také z: http://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf

BAUGH, Wesley. 2015. Wikipedia Extractor. *GitHub* [online]. [cit. 2015-05-12]. Dostupné z: https://github.com/bwbaugh/wikipedia-extractor

Indexace. 2014. *KTD: Česká terminologická databáze knihovnictví a informační vědy (TDKIV)* [online]. Praha [cit. 2015-05-12]. Dostupné z: http://aleph.nkp.cz/F/51UAS8L1EKHQVH7AI6P5A3MYA2KDQP79B7XE3C2QG5KC1VL8Y6-53969?func=full-set-set&set_number=100881&set_entry=000007&format=999

SEBASTIANO, Vigna. MG4J: Managing Gigabytes for Java. *Sebastiano Vigna* [online]. [cit. 2015-05-12]. Dostupné z: http://mg4j.di.unimi.it/

SEDLÁČEK, Petr. 2006. *Extrakce informací z textu*. Fakulta informatiky, Masarykova univerzita, str. 3. Dostupné také z: http://is.muni.cz/th/51819/fi_m/Diplomova_prace.pdf

SCHMID, Helmut. TreeTagger: - a language independent part-of-speech tagger. *Centrum Für informations und sprachverarbeitung: Ludwig Maxmilians universitas München* [online]. [cit. 2015-05-12]. Dostupné z: http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

VOLOKH, Alexander a Günter NEUMANN. 2014. MDParser. *MDParser* [online]. [cit. 2015-05-12]. Dostupné z: http://mdparser.sb.dfki.de/

VLČEK, Lukáš. 2013. *Elasticsearch: Vyhledáváme hezky česky* [online]. [cit. 2015-05-12]. ISSN 1803-5620. Dostupné z: http://www.zdrojak.cz/clanky/elasticsearch-vyhledavame-cesky/

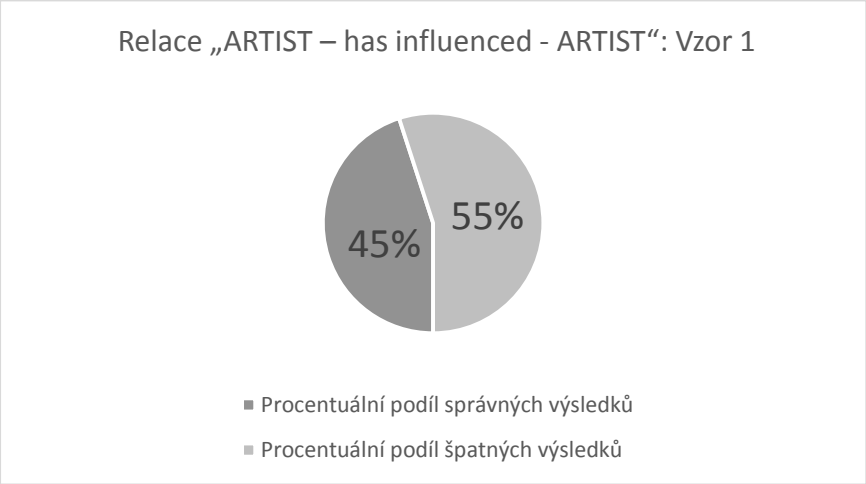
Wikipedie. 2001. *Wikipedie: Otevřená encyklopedie* [online]. [cit. 2015-05-12]. Dostupné z: http://cs.wikipedia.org/wiki/Wikipedie

Seznam příloh

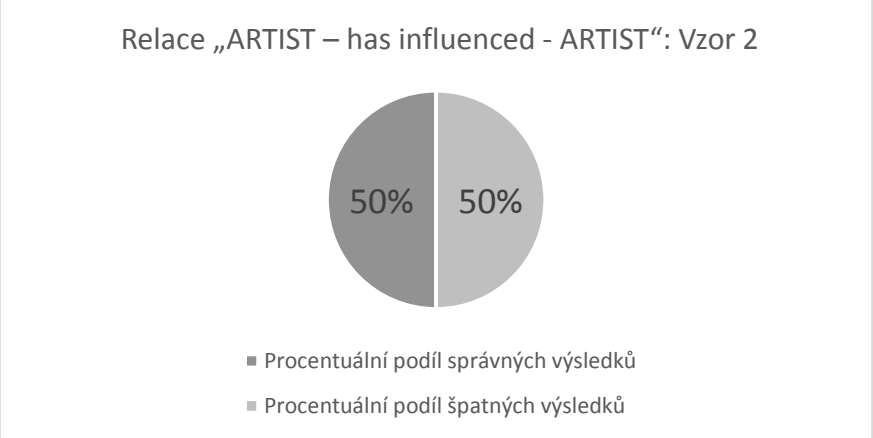
Příloha 1. Manuál – Program pro podporu extrakce relací
Příloha 2. Zdrojové texty – Program pro podporu extrakce relací
Příloha 3. CD/DVD
Příloha 4. Grafy

Příloha 4: Grafy

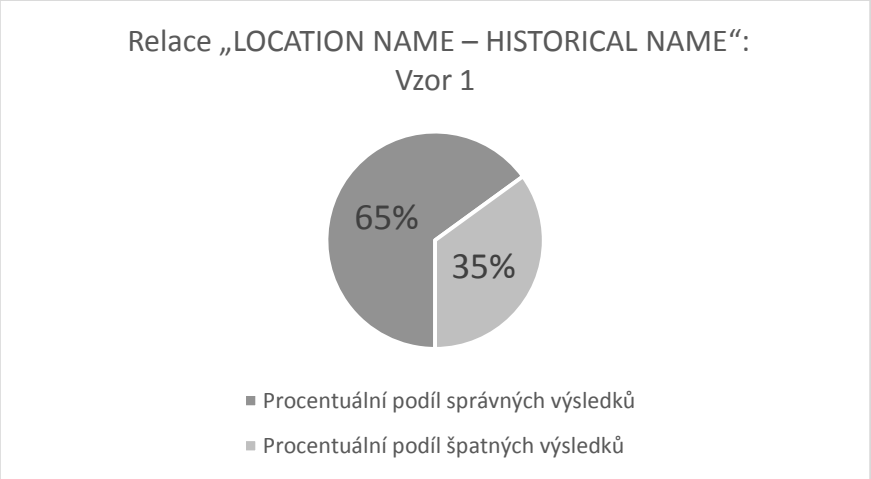
Graf č. 1



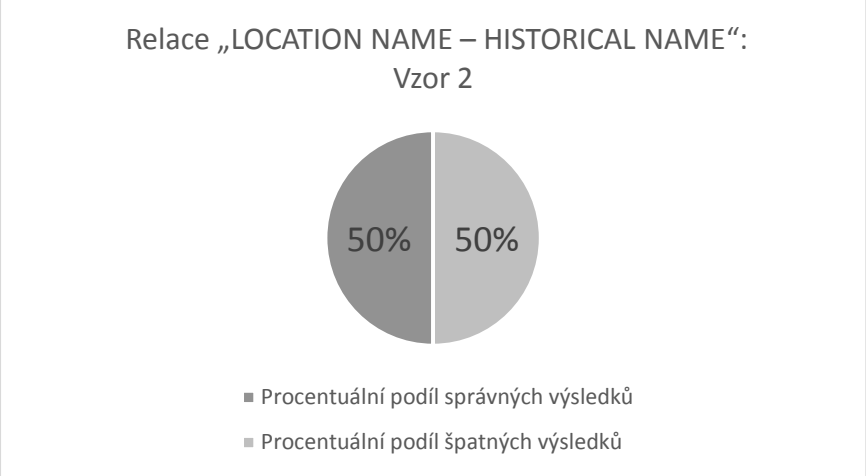
Graf č. 2



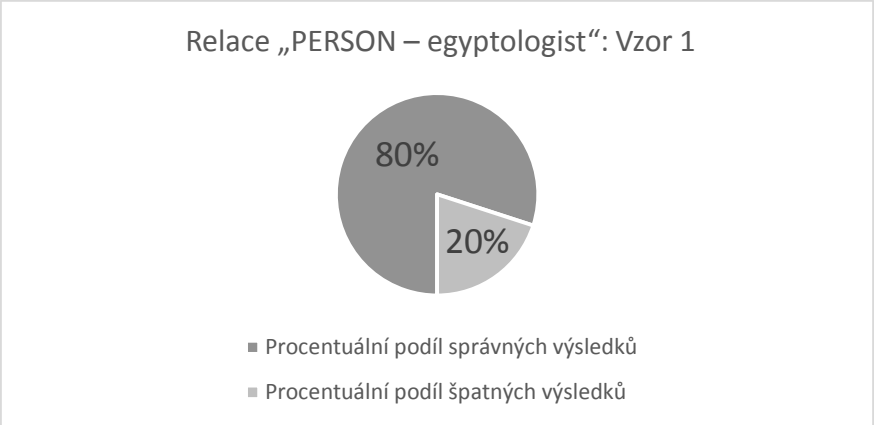
Graf č. 3



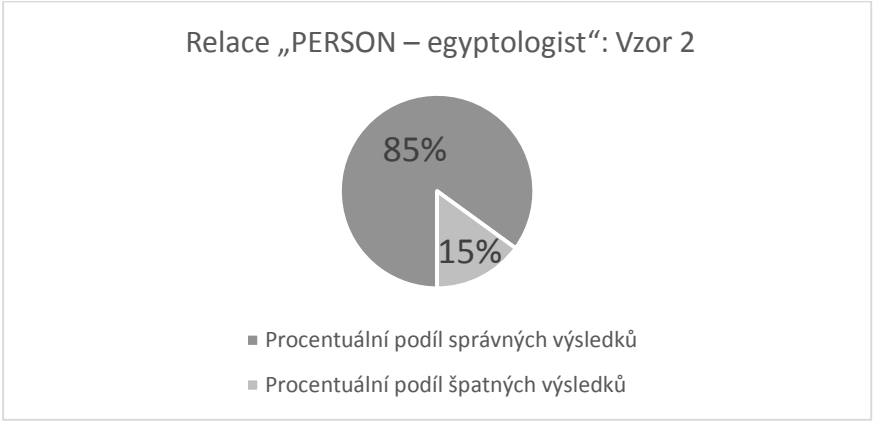
Graf č. 4



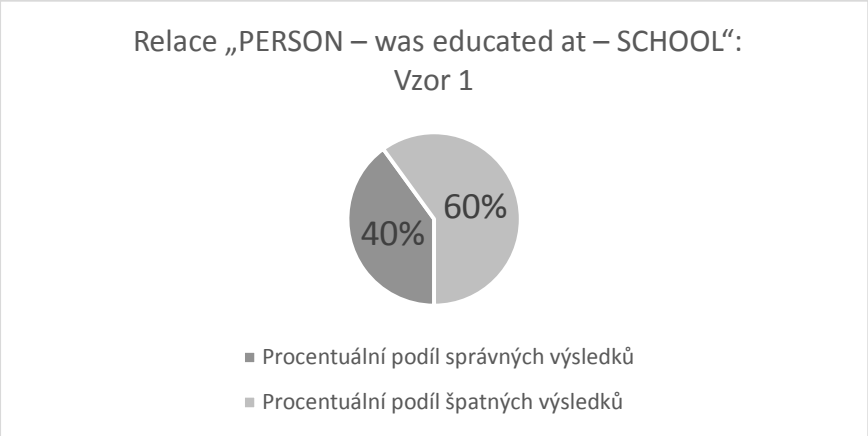
Graf č. 5



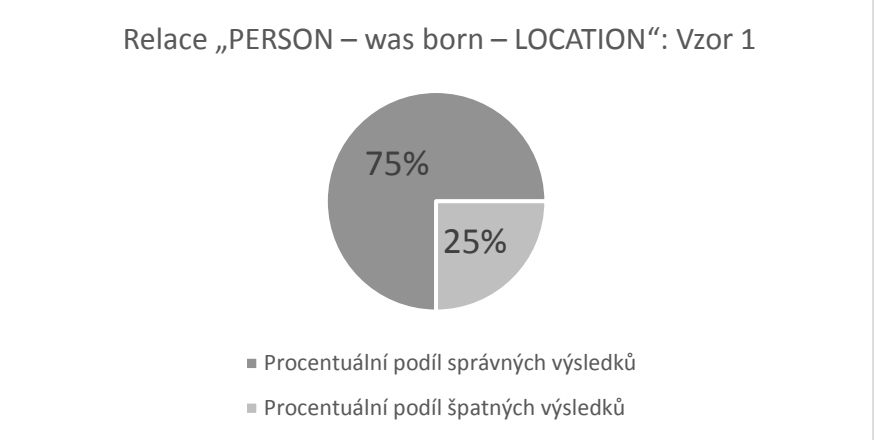
Graf č. 6



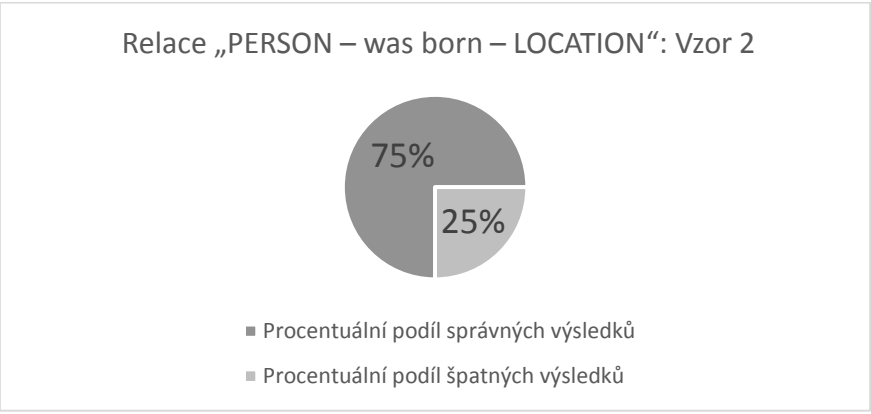
Graf č. 7



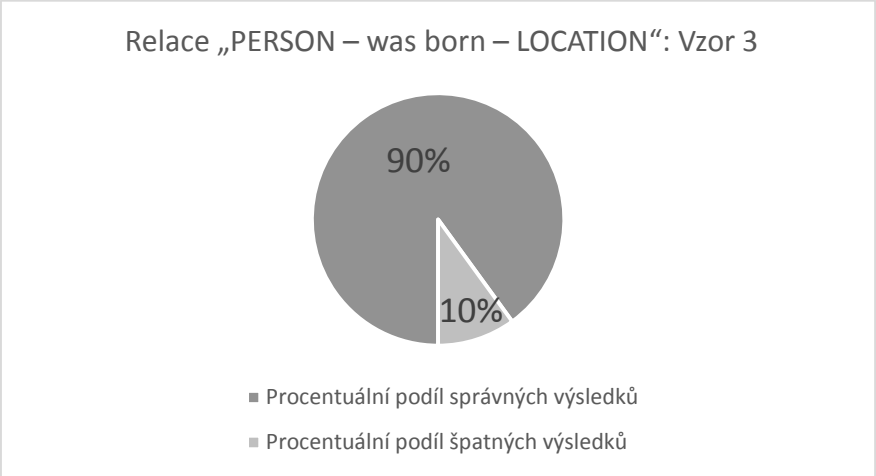
Graf č. 8



Graf č. 9



Graf č. 10



Graf č. 11