Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-0000-00000

Martin Nemček

Spracovanie učebných textov

Bakalárska práca

Vedúci práce: Ing. Miroslav Blšták

December 2015

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-0000-00000

Martin Nemček

Spracovanie učebných textov

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Miesto vypracovania: FIIT STU BA
Supervisor: Ing. Miroslav Blšták

December 2015

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika
Autor: Martin Nemček
Bakalárska práca: Spracovanie učebných textov
Vedúci práce: Ing. Miroslav Blšták
December 2015

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum....

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Informatika
Author: Martin Nemček
Bachelor thesis: Spracovanie učebných textov
Supervisor: Ing. Miroslav Blšták
December 2015

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum....

ACKNOWLEDGMENTS

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.....

DECLARATION

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua....

.....

Martin Nemček

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam ukážok

1 Úvod

Internet je v dnešných dňoch zaplnený obrovským množstvom dát a informácií. Mnohé z týchto dát sa na internete vyskytujú mnohonásobne, či už v identickej podobe alebo s úpravou. Avšak, čím ďalej tým viac, z týchto informácií vyskytujúcich sa na internete, sú informácie irelevantné.

Stáva sa to až príliš často a každý už zažil situáciu, kedy hľadal informácie na konkrétnu tému a musel sa „prehrabať" kopou nepodstatných dát a informácií, ktoré mu boli ponúkané. Stáva sa to medzi všetkými kategóriami používateľov na internete.

Jednou z majoritných kategórií používateľov, ktorí sa s takouto situáciou stretávajú denno denne sú študenti. Študenti všetkých škôl, od základných až po univerzity, získavajú informácie na učenie, projekty alebo zadania primárne z internetu alebo učebných textov kníh. Keď musia prechádzať obrovské množstvá dát z rôznych zdrojov, je to náročne, často až frustrujúce a berie im to veľmi veľa času. Tento čas by mohli využiť efektívnejšie, napríklad na rozvoj svojich vedomostí.

Učebné texty sú často písané v neštruktúrovanej forme a prirodzenom jazyku. Pre stroje je mnohokrát náročné správne interpretovať tieto informácie. Jedným z hlavných dôvodov je fakt, že každý jazyk je odlišný a obsahuje špecifické charakteristiky, ktoré môžu byť napríklad slovosled vety, gramatické kategórie slov, ale aj vetné členy a vzťahy medzi nimi.

Tieto, ale aj mnohé iné charakteristiky jazyka sa dajú využiť pri jeho spracovaní a reprezentáciu do podobu, s ktorou vedia aj stroje jednoducho narábať. Takýto proces sa nazýva *spracovanie prirodzeného jazyka* (angl. Natural Language Processing - NLP). Spracovanie jazyka ma viacero aplikácií, z ktorých sú to napríklad preklad jazyka, vytiahnutie najpodstatnejších entít z textu, prípadne aj štatistika ich výskytu a mnohé ďalšie.

My posunieme spracovanie prirodzeného jazyka ešte o kúsok ďalej a budeme sa zaoberať ako dopomôcť študentom so spracovaním veľkého množstva informácií, hlavne z učebných textov. Študentom najviac pomôže, ak dokážu rýchlo z textu vytiahnuť tie najpodstatnejšie, najdôležitejšie informácie a údaje, ktoré sa im ďalej budú omnoho ľahšie spracovávať a učiť. Proces určovania a extrakcie

najpodstatnejších informácií z učebného textu môžme nazvať spoznamkovávanie.

Zameriame sa hlavne na využitie vetných členov a vzťahov medzi nimi, na určenie najpodstatnejšej, najrelevantnejšej informácie z vety. Takto extrahované informácie následne ponúkneme používateľovi (študentovi).

2 Analýza

V tejto kapitole priblížím a rozoberiem čo je spracovanie prirodzeného jazyka, jeho využitie v aplikáciach a systémoch a jeho hlavné úlohy. Ďalej zanalyzujem nástroje, ktoré sa dajú využiť na spracovanie prirodzeného jazyka a tak isto sa pozriem na aplikácie a systémy, ktorých základom je spracovanie textu.

2.1 Spracovanie prirodzeného jazyka

Spracovanie prirodzeného jazyka (angl. Natural Language Processing - NLP) môže byť definované ako automatické alebo poloautomatické spracovanie ľudského jazyka [6]. Počítače doposiaľ nedokážu plne porozumieť ľudskému jazyku, či už sa jedná o písaný alebo hovorový, a preto hlavným cieľom NLP je vybudovať výpočtové modely prirodzeného jazyka pre jeho analýzu a generovanie [1].

Porozumenie ľudskej reči je mnohokrát náročne aj pre samotných ľudí a nie to ešte pre počítače. Na svete je veľké množstvo jazykov, ktoré sa od seba líšia charakteristikami typickými pre konkrétny jazyk. Navyše, každý človek je odlišný a typický, čo spôsobuje, že výslovnosť rovnakého slova viacerými ľuďmi môže byť odlišná. Ďalej máme slangové slová a slová typické len pre určité územie. Pri spracovávaní prirodzeného jazyka treba vziať do úvahy tieto, a aj ďalšie, premenné. Dosiahnutie tohto cieľa je preto často veľmi náročné.

V súčastnosti najpoužívanejšie algoritmy na NLP využívajú strojové učenie. Dosiahnutie úplného porozumenia a spracovania ľudského prirodzeného jazyka by znamenalo vyriešiť *AI-complete* problém, čo znamená, že obtiažnosť tohto problému je ekvivalentná s obtiažnosťou problému vytvorenia počítača inteligentného ako človek, takzvané „true AI".

2.2 Využitie spracovania prirodzeného jazyka

V súčastnosti má NLP niekoľko hlavných využití v aplikáciach a systémoch. Z hľadiska spracovania učebných textov je pre nás najdôležitejšie využitie z pohľadu *extrakcie informácií*, ktoré je podrobnejšie popísané v sekcii 2.2.1 Extrakcia informácií. Ďalšie využitia NLP sú napríklad [11]:

– Strojový preklad (angl. Machine Translation)
– Rozpoznávanie reči (angl. Speech Recognition)
– Sumarizáciu textu (angl. Text Summarization)
– Dialógové systémy (angl. Dialogue Systems)
– Výber informácií (angl. Information Retrieval)

2.2.1 Extrakcia informácií

Systémy a aplikácie zamerané na extrakciu informácií vyhľadávajú a extrahujú informácie z textov, článkov a dokumentov, pričom reagujú na používateľove informačné potreby. Výstup z takýchto systémov a aplikácií nepozostáva iba zo zoznamu kľúčových slov, ktoré by sa dali pokladať za extrahované informácie, ale naopak sú v tvare preddefinovaných šablón [11].

Extrakcia informácií využíva niekoľko z hlavných úloh spracovania prirodzeného jazyka. Sú to *Značkovanie slovných druhov*, *Rozpoznávanie názvoslovných entít*, a ďalšie [11]. Tieto a aj ostatné úlohy spracovania prirodzeného jazyka sú podrobnejšie opísané v sekcii 2.3 Úlohy spracovania prirodzeného jazyka.

Výber informácií a extrakcia informácií spolu úzko súvisia, ale sú to dve rozdielne využitia NLP. Prvé spomínané využitie slúži na vyhľadávanie relevantných zdrojov informácií v databázach textov, článkov a dokumentov podľa používateľových potrieb. Na vyhľadaných zdrojoch následne prebehne extrakcia informácií.

My sa pri spracovaní textov zameriame hlavne na extrakciu informácií, aby sme dokázali z učebného textu extrahovať relevantné informácie pre študenta, a tým získali poznámky.

Pomenovanie spracovanie prirodzeného jazyka a NLP budem používať zameniteľne v celej práci, pričom budú odkazovať na tu istú vec.

2.3 Úlohy spracovania prirodzeného jazyka

NLP ma niekoľko hlavných úloh. Podrobnejšie si opíšeme tie, ktoré sú relevantné vzhľadom na implementáciu spracovania učebných textov. Úlohy spracovania prirodzeného textu: [5]

- Značkovanie slovných druhov (angl. Part-of-speech tagging) 2.3.1
- Rozdelenie vety na menšie časti (angl. Chunking)
- Rozpoznávanie názvoslovných entít (angl. Named Entity Recognition) 2.3.2
- Označovanie sémantického postavenie (angl. Semantic Role Labeling)
- Rozpoznanie koreferencií (angl. Coreference resolution) 2.3.3
- Morfologické segmentovanie (angl. Morphological Segmentation)
- Generovanie prirodzeného jazyka (angl. Natural Language Generation)
- Optické rozoznávanie textu (angl. Optical Character Recognition)
- Rozloženie vzťahov (angl. Dependency parsing) 2.3.4
- a ďalšie

2.3.1 Značkovanie slovných druhov

Hlavnou úlohou značkovania slovných druhou (angl. Part-of-speech tagging) je každému slovu vo vete priradiť unikátnu značku odrážajúcu jeho syntaktickú úlohu vo vete [5]. Sú to, napríklad v slovenskom jazyku podmet, prísudok, príslovkové určenie alebo v anglickom jazyku noun, adverb, verb, atď. Okrem toho to môže byť tiež označenie určujúce množné číslo, napríklad signulár alebo plurál.

Problémom pri značkovaní slovných druhov je mnohoznačnosť. Mnohoznačnosť je vlastnosť slova spôsobujúca, že slovo môže mať viacero významov a môže byť viacerými slovnými druhmi. V slovenskom jazyku napríklad slovo *kry* môže predstavovať sloveso s významom rozkazu *prikry!*, ale taktiež môže predstavovať podstatné meno s významom *kríky*. V anglickom jazyku to je napríklad slovo *book*, ktoré môže predstavovať podstatné meno (angl. noun) *kniha* alebo sloveso (angl. verb) vo význame *rezervovať*.

2.3.2 Rozpoznávanie názvoslovných entít

Rozpoznávanie názvoslovných entít (angl. Named Entity Recognition) označuje mená a názvy (entity), ktoré sa vyskytujú v texte. Tie následne rozdeľuje do kategórií, ako sú napríklad *osoby*, *organizácie* alebo *lokácie* [5].

Ťažkosti pri rozpoznávaní názvoslovných entít spôsobuje kapitalizácia slov, takzvané písanie entít s veľkým začiatočným písmenom. V anglickom jazyku je to

jednoduché, keďže v angličtine sa entity píšu s veľkým začiatočným písmenom.

Príklad je *Slovak University of Technology*. Avšak v iných jazykoch to neplatí a entity sa nemusia písať s veľkým začiatočným písmenom.

2.3.3 Rozpoznanie koreferencií

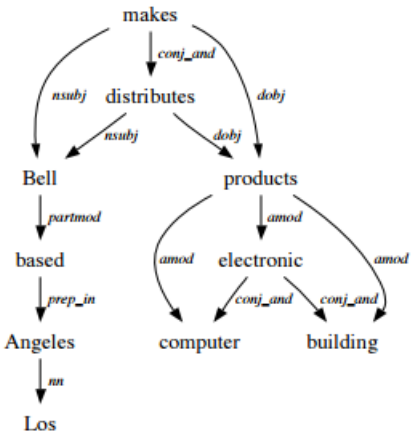
Nájdenie, identifikácia a rozpoznanie koreferencií v texte je úlohou rozpoznávania koreferencií (angl. Coreference resolution). V texte sa často používajú zámena (angl. pronouns) *to*, *tí*, *on*, anglicky *it*, *those*, *he* alebo menné frázy (angl. noun phrase). Tieto zámena a menné frázy sa odkazujú na iné podstatné mená alebo mená a názvy. Je úlohou rozpoznávania koreferencií identifikovať referenciu na podstatné meno alebo meno, alebo názov, väčšinou entity z reálneho sveta, na ktoré sa odkazujú. Táto úloha spracovania prirodzeného textu sa využíva v aplikáciách NLP ako sú extrakcia informácií (viď. 2.2.1 Extrakcia informácií) a odpovedanie na otázky [2].

Príklad: **Martin Nemček** napísal túto bakalársku prácu. **On** študuje na FIIT STU BA.

Tu je vidno, že zámeno *on* sa odkazuje na meno *Martin Nemček*.

2.3.4 Rozloženie vzťahov

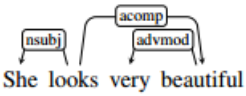
Rozloženie na vzťahy nám poskytuje jednoduchý opis gramatických vzťahov slov vo vete. Aplikovaním rozloženia vzťahov na vetu *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.* vznikne strom vzťahov (angl. dependency tree) (viď. obrázok 1 Strom vzťahov) [3].



Obr. 1: *Strom vzťahov*

V tomto orientovanom stromovom grafe jednotlivé slová vety predstavujú vrcholy, pričom prechody medzi vrcholmi, hrany, reprezentujú vzťahy medzi nimi.

Ďalšia reprezentácia vzťahov zapisuje vzťahy priamo do vety. Na obrázku 2 Vzťahy vo vete vidíme, že medzi slovami *She* a *looks* je vzťah **nsubj** - nominal subject, medzi *looks* a *beautiful* je vzťah **acomp** - adjevtival complement, a v neposlednom rade medzi slovami *very* a *beautiful* je vzťah **advmod** - adverb modifier [3].



Obr. 2: *Vzťahy vo vete*

2.4 Nástroje na spracovanie prirodzeného jazyka

V súčastnosti je vyvinutých alebo sú vo vývoji viacero nástrojov, ktoré sa dajú použiť pri spracovávaní prirodzeného jazyka. Vývoj takýchto nástrojov je podporovaný na známych univerzitách ako sú napríklad Princeton, Stanford alebo Camridge, ale samozrejme svoje slovo tu má aj velikán Google. Pozrieme sa

bližšie na niektoré z týchto nástrojov, čo ponúkajú a ako sa dajú využiť.

2.4.1 WordNet

WordNet je databáza anglických slov vyvíjaná na Princetonskej univerzite. Databáza obsahuje podstatné mena, prídavné mená, slovesá a príslovky, ktoré sú zatriedené do synonymických sád, synsetov [4].

Slová do synetov sú zaraďované podľa významu. To znamená, že slová auto a automobil, ktoré sú pre svoj význam zameniteľné vo vete, sa zaradujú do rovnakého synsetu. WordNet v súčastnosti (r. 2015) obsahuje 117 000 synsetov. Každý z týchto sysnsetov taktiež obsahuje krátku ukážku použitia slova [4].

Vo WordNet-e sa nachádzajú aj vzťahy medzi slovami v zmysle nadradenosti. Tým sa myslí, že stolička je nábytok a nábytok je fyzická vec a takto to pokračuje až po najvyššie slovo, od ktorého „dedia" všetky - entita (viď. obrázok 4 Nadradenosť slov. Okrem vzťahu nadradenosti WordNet obsahuje aj vzťah zloženia. Stolička sa skladá z operadla a nôh. Toto zloženie je typické len konkrétne slovo a neprenáša sa hore stromom nadradenosti, lebo pre stoličku je typické, že sa skladá z operadla a nôh, ale to už nie je typické pre nábytok. Prídavné mená obsahujú aj vzťah antonymity, takže slovo *suchý* bude prepojené so slovom *mokrý* ako so svojím antonymom [4].

Tento nástroj je dostupný vo webovej verzií (viď. obrázok 3 Webové rozhranie), ale ponúka aj stiahnutie jeho databázových súborov, ktoré sa po splnení licenčných požiadaviek dajú využívať v projektoch.

Obr. 3: *Webové rozhranie*

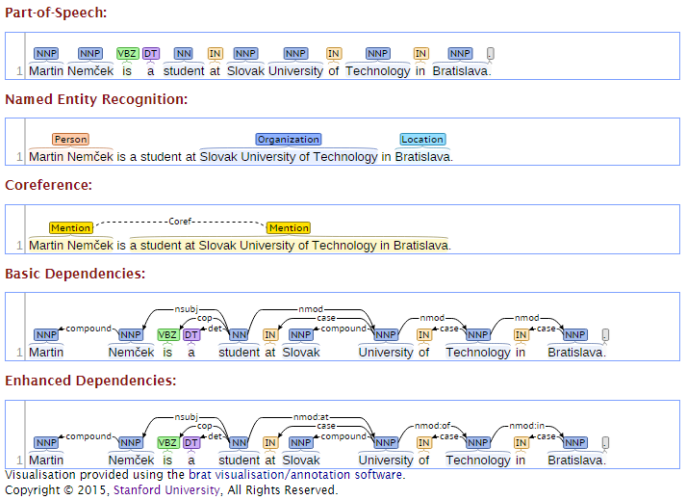
Obr. 4: *Nadradenost'slov*

2.4.2 StanfordNLP

Nástroj StanfordNLP je vyvíjaný na Stanfordskej univerzite. Skladá sa z niekoľkých softvérov, ktoré sa zameriavajú na úlohy spracovania prirodzeného jazyka popísané v sekcií 2.1 Spracovanie prirodzeného jazyka. Sú to softvéry *Stanford Parser*, *Stanford POS Tagger*, *Stanford EnglishTokenizer*, *Stanford Relation Extractor* a mnoho ďalších. *Stanford CoreNLP* zahŕňa viacero z týchto softvérov, a práve tento nástroj budeme používať pri spracovaní učebných textov.

Nástroje StanfordNLP sú implementované v Jave, ale sú dostupné aj v iných programovacích jazykoch ako C#, PHP alebo Python.

Dostupne je aj online webové demo. Na obrázku 5 StanfordNLP online demo vidíme výstupy z nástrojov ponúkaných balíkom StanfordNLP pre jednoduchý vstupný text skladajúci sa z jednej vety „Martin Nemček is a student at Slovak University of Technology in Bratislava.”.



Obr. 5: *StanfordNLP online demo*

2.4.3 CambridgeAPI

CambridgeAPI je vytvorený na Cambridge univerzite. Umožňuje prístup k viacerým rôznym slovníkom. Momentálne tento nástroj ponúka prístup k pätnástim prekladovým slovníkom ako napríklad anglicko-čínsky, anglicko-ruský, anglicko-

arabský, anglicko-japonský a ďalšie. Všetky prekladové slovníky majú primárny jazyk angličtinu. Slovenčinu v súčastnosti nepodporuje.

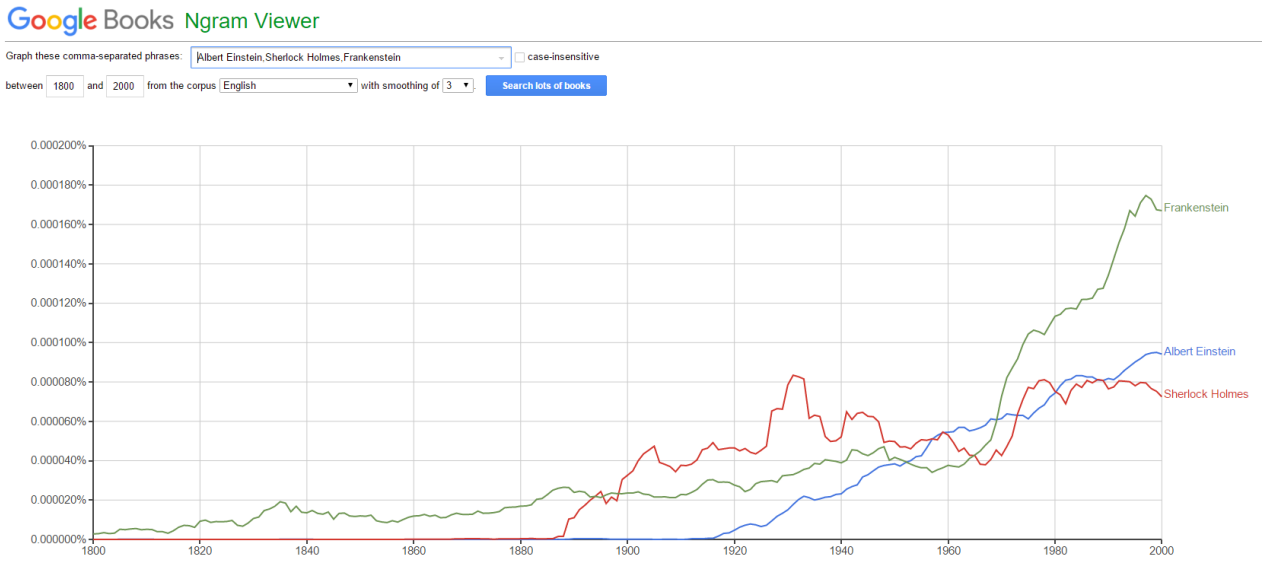
Spomínaný nástroj funguje na princípe dopytovania pomocou HTTP protokolu. Na obdržanie korektnej odpovede je potrebné mať osobný API kľúč. Ten sa dá získať kontaktovaním správcov CambridgeAPI.

2.4.4 Google Ngram

Google Ngram je postavený na ďalšom softvéry tohto giganta, Google Books. V knihách, napísaných od roku 1500 až do súčastnosti, vyhľadáva výskyty n-gramov. Podporuje len niektoré jazyky, ako angličtina, francúzština, ruština, čínština. Na vyhľadávanie v knihách využíva optické rozoznávanie textu, pričom dokáže spracovať aj regulárne výrazy, avšak tie môžu byť použité iba ako náhrada celého slova, ale nie uprostred slova. Slovné spojenie „* Einstein" spracuje, pričom „Albert Einste*n" nie.

N-gram je podľa oxfordského slovníka definovaný ako postupnosť *n* za sebou idúcich slov alebo znakov. *Martin* je n-gram veľkosti jedna, 1-gram alebo unigram. *Martin Nemček* je n-gram veľkosti dva, 2-gram alebo bigram a tak ďalej, pričom *n* môže byť ľubovolné kladné, celé číslo.

Google Ngram Viewer poskytuje vizualizáciu vyhľadaných dát. Je dostupný vo webovom rozhraní. Na obrázku 6 Google Ngram Viewer vidno vizualizáciu výskytu mien *Albert Einstein,Sherlock Holmes,Frankenstein* v knihách od roku 1800 do roku 2000.



Obr. 6: *Google Ngram Viewer*

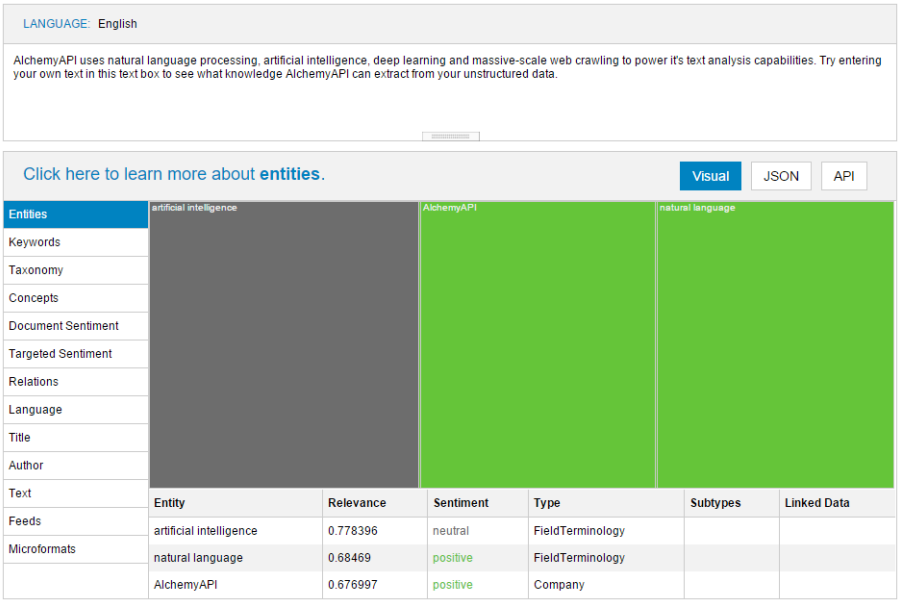
Tento nástroj okrem iného ponúka aj surové (angl. raw) dáta na stiahnutie.

2.4.5 AlchemyAPI

AlchemiAPI obsahuje dvanásť funkcií, z ktorých sú niektoré zamerané na úlohy spracovania prirodzeného jazyka popísané v sekcií 2.1 Spracovanie prirodzeného jazyka, ako napríklad extrakcia entít, extrakcia kľúčových slov, extrakcia vzťahov, ale aj iné zaujímave funkcie, napríklad extrakcia autora z textu.

Na používanie tohto nástroja je potrebné sa zaregistrovať pre obdržanie API kľúču. S týmto kľúčom je tisíc dopytov denne zdarma. Dostupnosť v programovacích jazykoch je široká, kedže ponúka knižnicu v deviatich najpoužívanejších programovacích jazykoch.

Pre AlchemyAPI je dostupné aj online webové demo, viď obrázok 7 AlchemyAPI online demo, kde je vidno širokú ponuku, ktorú tento nástroj ponúka.



Obr. 7: *AlchemyAPI online demo*

Dáta sú vo formáte JSON a okrem spracovania prirodzeného jazyka AlchemyAPI ponúka aj nástroje na extrahovanie obsahu z obrázku alebo rozpoznávanie tvári na obrázkoch.

2.5 Aplikácie na spracovanie prirodzeného jazyka

Dostupnosť aplikácií na spracovanie prirodzeného jazyka je veľká a široká. Najväčší podiel tvoria aplikácie zamerané na preklad. V tejto kapitole si prestavíme niektorých predstaviteľov tejto kategórie aplikácií.

2.5.1 InterText

InterText[1] je editor paralelne zarovnaných textov, využívaný na správu viacerých paralelne zarovnaných verzií textu rôznych jazykov na úrovni viet. Táto aplikácia je dostupná vo verzií pre desktop, ale aj pre server.

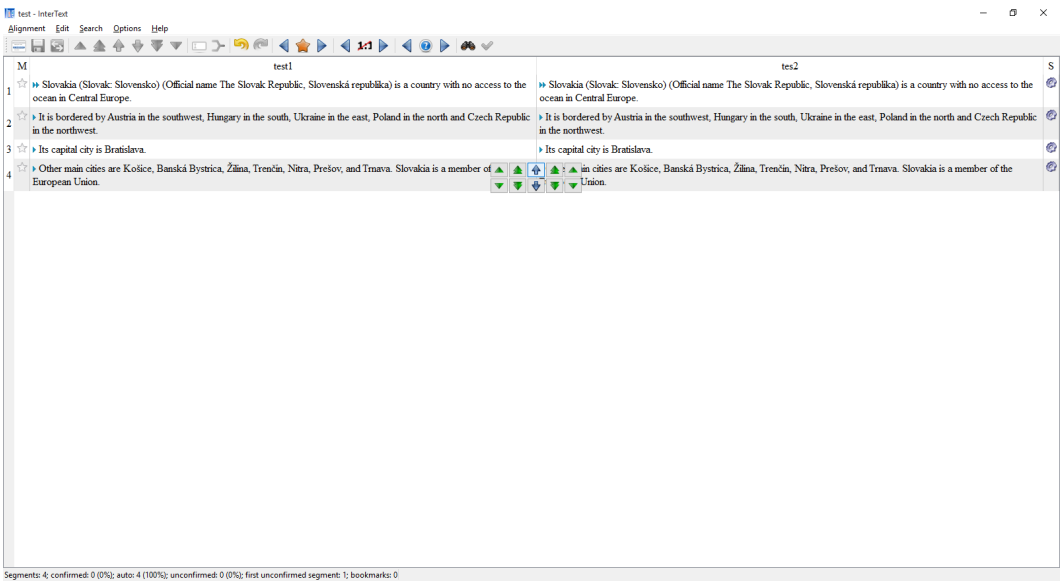
Podporuje viacero formátovaní textu, či už čistý (angl. plain) text alebo XML a taktiž zobrazuje aj HTML značky. Riadky obsahujú vety oddelené znakom konca

[1] http://wanthalf.saga.cz/intertext

riadku a sú očíslované. Umožňuje funkcie ako presúvanie riadkov textu alebo zoskupenie viacerých do jedného, krok vpred a vzad. V spracovávanom texte sa dá vyhľadávať a je možné tento text aj upraviť podľa vlastných potrieb.

InterText nezohľadňuje používateľove úpravy textu počas používanie a pri následnom spracovávaní textu sa tak neprispôsobí používateľovi. Okrem toho jeho náplň sa nedá pokladať za zjednodušovanie textu.

Na obrázku 8 Aplikácia InterText je zobrazená aplikácia InterText s testovacím vstupom, na ktorom je vidno väčšinu, už spomenutej, funkcionality, ako presúvanie a zoskupovanie riadkov, číslovanie, atď.



Obr. 8: *Aplikácia InterText*

2.5.2 NOVA Text Aligner

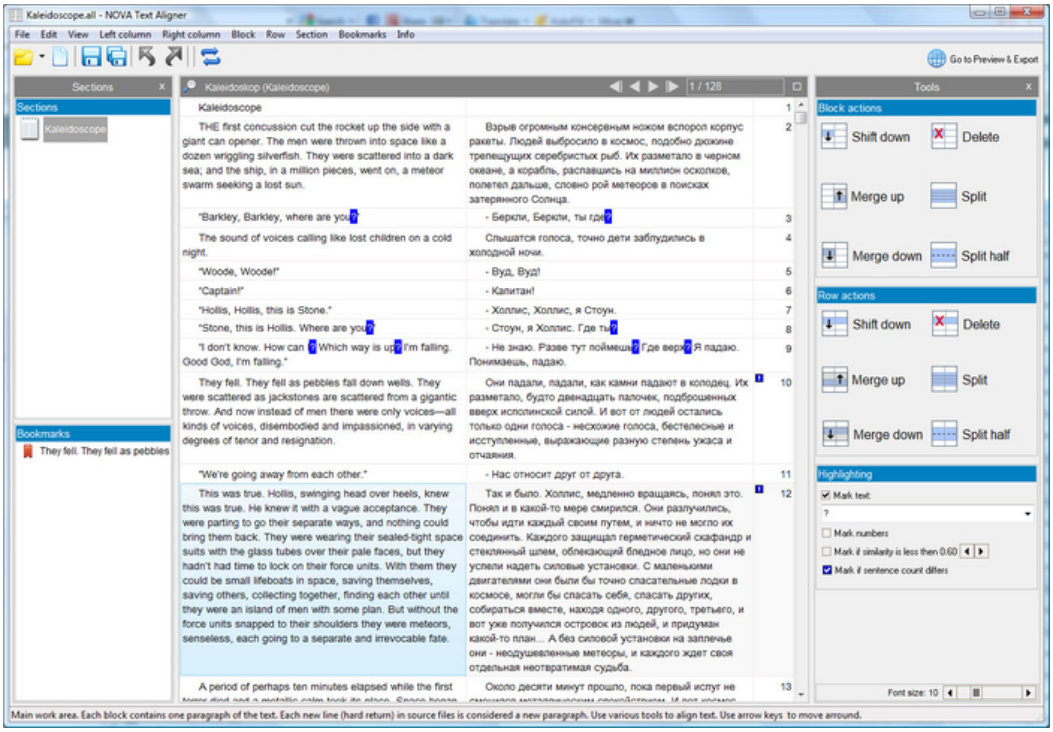
NOVA Text Aligner[2] je aplikácia na zarovnávanie textu, pričom nevyužíva algoritmy na zarovnávanie textu, ale manuálne používateľovo určovanie zarovnania.

Ako vidno na obrázku 9 Aplikácia NOVA Text Aligner hlavná editovacia časť aplikácie je rozdelené do dvoch častí. Umožňuje do ľavej aj pravej časti načítať rôzny text, v ktorom sa dá veľmi jednoducho vyhľadávať, k čomu napomáha zvý-

[2]http://www.supernova-soft.com/wpsite/products/text-aligner/

raznenie vyhľadaných slov. Načítaný text je možné premiestňovať a zoskupovať, či už podľa riadkov alebo aj v celých blokoch a nechýba možnosť editovať text. Je možné si túto aplikáciu prispôsobiť. Ponúka možnosti ako zmena typ písma a pod. Finálny, spracovaný text sa dá exportovať do viacerých formátov, z ktorých populárne sú formáty elektronických knižiek EPUB a MOBI.

Aplikácia je zameraná hlavne na usporadúvanie textu, nezaznamenáva si používateľove zmeny textu a neprispôsobuje sa podľa toho pri ďalšom použití, funguje iba lokálne. NOVA Text Aligner je dostupná iba v skúšobnej verzii, pre dlhodobé používanie si treba zakúpiť licenciu.



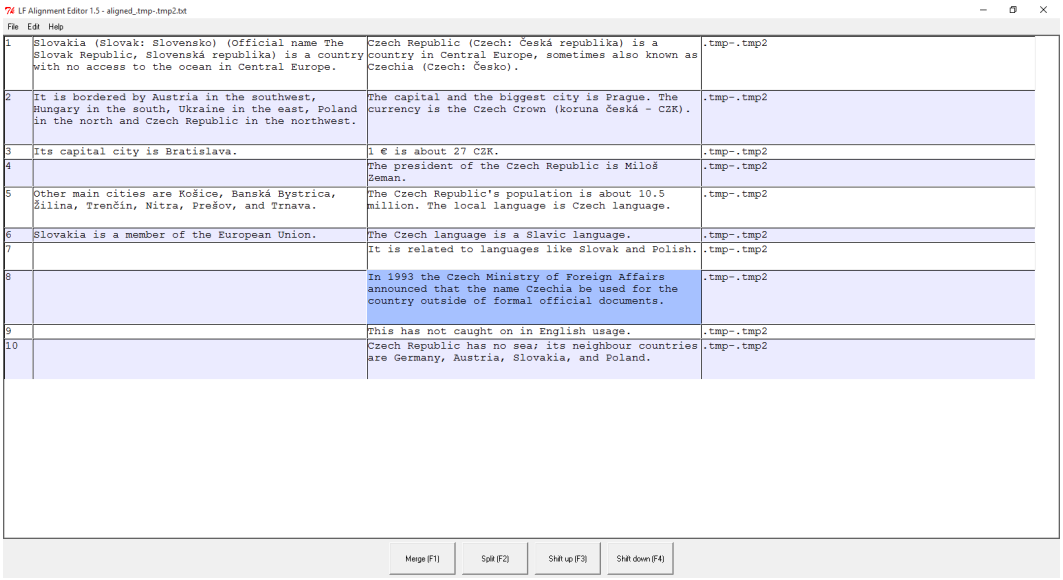
Obr. 9: *Aplikácia NOVA Text Aligner*[3]

2.5.3 LF Aligner

Aplikácia LF Aligner je zameraná na spracovanie textu rôznych jazykov. Ponúka možnosť použiť až 99 jazykov, čo ale znamená 99 vstupných súborov, každý so

[3] http://parallel-text-aligner.en.softonic.com/

zvoleným jazykom. Dokáže spracovať rôzne typy vstupných súborov od čistého textu, PDF súborov, cez URL stránok s textom až po správy Európskeho parlamentu, ktoré automaticky stiahne. Výstup môže byť taktiež viacerých druhov, napríklad cez grafické rozhranie LF Aligner alebo vygenerovanie XLS súboru. Na obrázku 10 Aplikácia LF Aligner vidno grafické rozhranie tejto aplikácie, ktoré ponúka mnohé vymoženosti. Samozrejme je možné premiestňovať a zoskupovať riadky, doplnenie ďalšieho súboru na spracovávanie, uloženie zmien súboru prepísaním jeho dát a mnohé ďalšie.



Obr. 10: *Aplikácia LF Aligner*

2.5.4 Zhrnutie

Všetky analyzované aplikácie sú užitočné vo svojom obore, ale ani jedna nespĺňa všetky požiadavky na systém schopný spoznámkovať učebný text v takom rozsahu, ktorý by umožňoval používateľovi prispôsobiť si spracovaný text. Systém musí vedieť prispôsobovať svoje spracovanie textu podľa používateľových úprav a ponúkať mu k tomu vhodné rozhranie. Takisto musí ukladať dáta mimo používateľovho úložného priestoru.

[3]http://parallel-text-aligner.en.softonic.com/

3 Smerovanie práce

V letnom semestri plánujem dokončiť prototyp. To znamená, spraviť používateľské rozhranie, ktoré bude umožňovať vložiť text na spracovanie, zobrazí poznámky a tak isto umožní používateľovi pre ľubovolnú vetu pozmeniť tvar poznámky poznámky. Tieto zmeny sa uložia do databázy a zohľadnia pri ďalšom použití.

Okrem dokončenia prototypu plánujem napísať všetky potrebné kapitoly a dokončiť tým celú prácu.

4 Opis prototypu

V zimnom semestri som implementoval prototyp aplikácie na spoznámkovanie učebného textu.

4.1 Notenizer

Notenizer je prototyp aplikácie na extrahovanie relevantných informácií z učebných textov. Využíva nástroj Stanford CoreNLP, ktorý je implementovaný v Jave, ale cez IKVM je portnutý aj na C#. Na ukážke 1 Spustenie StanfordCoreNLP je ukázané prepojenie nástroja StanfordCoreNLP s aplikáciou Notenizer.

```
String jarRoot = @"stanford-corenlp-3.5.2-models";

Properties properties = new Properties();
// Zvolime, ktore nastroje chceme pouzit.
// pos = part-of-speech tagger
// ssplit = sentence split
// atd.
properties.setProperty("annotators", "tokenize, ssplit, pos, parse");
properties.setProperty("sutime.binders", "0");
properties.setProperty("ner.useSUTime", "false");

// Nastavenie aktualneho priecinku, aby StanfordCoreNLP vedel najst
// vsetky potrebne subory
String currentDirectory = Environment.CurrentDirectory;
Directory.SetCurrentDirectory(jarRoot);
StanfordCoreNLP pipeline = new StanfordCoreNLP(properties);
Directory.SetCurrentDirectory(currentDirectory);
```

```
// Vytvorenie anotacie z textu
Annotation annotation = new Annotation(text);

// Spustenie
pipeline.annotate(annotation);
```

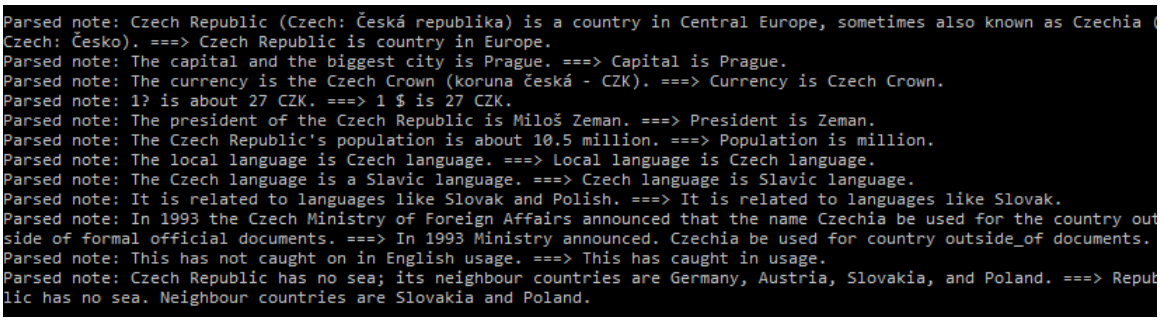
Ukážka 1: *Spustenie StanfordCoreNLP*

Údaje získane z tohto nástroja, napríklad POS značky, vzťahy medzi slovami, pozície slov a mnoho ďalších, Notenizer ďalej spracováva. Najdôležitejšie vlastnosti, ktoré sa využívajú v najväčšej miere pri spracovávaní sú závislosti (angl. dependency) medzi slovami vo vete.

Spracovávaný text sa postupne spracováva po vetách. Každá veta sa samostatne „rozparsuje", spoznámkuje. Vety sa parsujú na základe pravidiel. Na začiatku je daná statická sada pravidiel na spracovanie viet a textov. Po tom, ako sa celý text spracuje, tak sa použité pravidlá uložia do databázy aj s informáciami o pôvodnej vete a novo vytvorenej, zjednodušenej vete. Následne pri opätovnom používaní aplikácie, keď sa začne spracovávať text, tak sa vyhľadajú pre každú vetu pravidlá v databáze, vyberú sa tie s najväčšou zhodou a podľa toho sa spracuje daná veta. Statické pravidla na spracovanie vety sa v tomto prípade použijú len v prípade, ak v sa v databáze nenašli žiadne pravidlá na spracovanie vety, ktoré by pre danú vetu vyhovovali, to znamená, že takúto alebo podobnú vetu zatiaľ Notenizer nespracovával.

Na obrázku 11 Ukážkový výstup prototypu je ukázaný ukážkový výstup prototypu pre vstupný text z wikipédie: *Czech Republic (Czech: Česká republika) is a country in Central Europe, sometimes also known as Czechia (Czech: Česko). The capital and the biggest city is Prague. The currency is the Czech Crown (koruna česká - CZK). 1€ is about 27 CZK. The president of the Czech Republic is Miloš Zeman. The Czech Republic's population is about 10.5 million. The local language is Czech language. The Czech language is a Slavic language. It is related to languages like Slovak and Polish. In 1993 the Czech Ministry of Foreign Affairs announced that the name Czechia be used for the country outside of formal official documents. This has not caught on in English usage. Czech Republic has no sea; its neighbour countries are Germany, Austria, Slovakia, and Poland.*

Výstup je v tvare [pôvodná veta] ===» [poznámka z pôvodnej vety].



Obr. 11: *Ukážkový výstup prototypu*

4.1.1 Pravidlá

Pri spracovaní pôvodnej vety sa na túto vetu aplikuje *pravidlo na spracovanie*. Toto pravidlo obsahuje okrem iného zoznam závislostí slov vo vete. Podľa týchto závislostí slov vo vete sa v spracovávanej vete vyhľadajú slová, ktoré majú byť použité v poznámke. Vyhľadávajú sa, okrem iného, podľa POS značiek a indexov vo vete.

4.1.2 Ukladanie a hodnoty pravidiel

Po spracovávaní sa do databázy uložia použité pravidla aj s ostatnými informáciami o pôvodnej a novej vete. Ukladá sa

- Hodnota pôvodnej a novej vety
- Zoznam indexov slov, za ktorými bola v poznámke ukončená veta (ak pôvodná veta je súvetie, tak z nej môže vzniknúť viacero poznámok)
- Všetky závislostí slov v pôvodnej vete
- Všetky závislosti slov v poznámke

pričom každá závislosť slov vo vete sa skladá z

- Názvu závislosti
- Hodnota governora závislosti
- Hodnota dependenta závislosti

– Pozícia závislosti vo vete
– POS značka a index slova pre governora aj dependenta

4.1.3 Vyhľadanie pravidla

Pri spracovávaní vety sa v prvom kroku pozrie do databázy a vyhľadá sa pravidlo na spracovanie tejto vety. Pravidlo sa v databáze vyhľadáva podľa nasledovných podmienok.

1. Počet závislostí vety
2. Názvy závislostí vety

Veta v databáze, ktorej pravidlo chceme použiť, musí spĺňať tieto dve pravidlá. Musí mať rovnaký počet závislostí vo vete ako aktuálne spracovávaná veta a taktiež názvy všetkých závislostí musia sedieť.

Avšak pri týchto podmienkach môže nastať situácia, kedy pre aktuálne spracovávanú vetu bude vyhovovať viacero viet z databázy. V tomto prípade určujeme pravidlo vety s najväčšou zhodou a to sa následne aplikuje.

Zistenie najväčšej zhody ma viacero krokov. Najskôr sa spočítavajú zhody POS značiek governerov a dependentov a indexy slov nezávisle od seba, čiže sa len zisťuje, či spracovávaná veta obsahuje závislosť s nejakou hodnotou indexu, alebo governora, atď. V druhom kroku sa spočítavajú zhody POS značiek a zároveň aj indexov slov vo vete. To znamená, že sa zisťuje, či sa v spracovávanej vete nachádza napríklad závislosť, ktorej governor ma hodnotu *car* a zároveň má index hodnotu 3. V poslednom, treťom, kroku sa zisťuje, či sa závislosti zhodujú na všetkých hodnotách, čiže governor a jeho hodnota a index a dependent a jeho hodnota a index *súčasne*.

Tieto tri hodnoty sa na záver spočítajú a tým získame percentuálne ohodnotenie zhody viet.

5 Návrh

5.1 Uchovávanie textov v databázach

Text je špecifický údajový model s variabilnou štruktúrou. Ak chceme efektívne ukladať texty v databázach, je nutné aby sme použili databázu, ktorá je tomu prispôsobená, pri ktorej nebudeme zbytočne čerpať pamäť a takisto bude jednoduché narábať s dátami. To znamená bezproblémové ukladanie, získavanie, vyhľadávanie a spracovanie textov na úrovni databázy. V nasledujúcich kapitolách sa pozrieme, aké typy databáz existujú a aké možnosti z pohľadu ukladania textov ponúkajú.

5.1.1 Relačné databázy

Relačné databázy boli dlhé roky populárnou a finančné nenáročnou voľbou pri tvorbe veľkých podnikateľských aplikácií [9]. Momentálne sú používané vo väčšine súčasných aplikácií a pracujú spoľahlivo pri obmedzenom množstve dát [7]. Problém s relačným modelom relačných databáz nastáva, keď vzniká potreba aplikácie s obrovským množstvom dát. Menovite rozšíriteľnosť (angl. scalability) sa stáva najväčším problémom relačných databáz [10].

Tento typ databáz oplýva veľkou úrovňou jednotvárnosti, ukladá dáta v tabuľkách zložených z riadov a stĺpcov. Každý záznam (riadok) v tabuľke predstavuje zjednodušený objekt alebo vzťah z reálneho života. Výhodou relačných databáz je možnosť jednoduchého vytvorenia prispôsobeného pohľadu na dáta [8].

5.1.2 Textové databázy

S rozmachom variácie dát v posledných rokoch sa začali objavovať a vznikať nerelačné databázy, aby pokryli požiadavky na nové aplikácie. [9]. Textové databázy sú druhom nerelačných databáz.

Textové databázy ukladajú dáta vo forme dokumentov, vďaka čomu ponúkajú vysoký výkon a horizontálnu rozšíriteľnosť [10]. Uložené dokumenty môžu nadobúdať rôzne typy, ako napríklad JSON, BSON, XML a BLOB, ktoré poskytujú veľkú flexibilnosť pre dáta [9]. Každý záznam v takejto databáze preto môže mať inú štruktúru, napríklad počet alebo typ polí, čo šetrí úložným priestorom, keďže

neobsahuje nepotrebné prázdne polia [10].

Dokumenty v databáze sú referencované kľúčom, ktorý môže byť string, cesta, ale dokonca aj dokument [10]. Majú dynamickú schému, čo umožňuje vytvárať záznamy bez toho, aby bolo potrebné predtým definovať štruktúru. Uľahčujú zmenu štruktúry záznamov jednoduchým pridaním, odstránením alebo zmenením typu poľa. Vďaka svojej štruktúre sú dokumenty ľahko namapovateľné na objekty z objektovo-orientovaných programovacích jazykov a odstraňujú tým potrebu pre použitie objektovo-relačnej mapovacej vrstvy [9].

Primárne využitie týchto databáz je v aplikáciách, ktoré potrebujú ukladať dáta, ktorých štruktúra je vopred neznáma alebo sa mení. Predstaviteľmi sú napríklad *MongoDB* alebo *CouchDB* databázy.

5.1.2.1 MongoDB

MongoDB je dokumentová nerelačná databáza vytvorená v C++ spustená v roku 2009 [10]. Ukladá dáta v dokumentoch vo formáte BSON (Binary JSON), ktorých štruktúra sa môže meniť. Využíva dynamickú štruktúru schém, preto dokáže vytvárať záznamy bez preddefinovanej štruktúry dát, lebo štruktúra sa vytvára za behu, pričom môže byť veľmi jednoducho pozmenená pridaním, odstránením, zmenou typu polia dokumentu určujúceho štruktúru. Umožňuje jednoduché ukladanie dát s hierarchickými vzťahmi alebo komplexnejších štruktúr, ako sú napríklad polia, listy alebo vnorené polia [9].

Vlastnosti ako chybová tolerancia, perzistencia a konzistencia dát sú súčasťou MongoDB. Oproti klasickým dokumentovým databázam ponúka aj vymoženosti, ako agregácia, ad hoc dopyty, indexovanie, a pod. Taktiež má svoj vlastný plnohodnotný dopytovací jazyk *mongo query language* [10].

Prvky poskytované databázou MongoDB sú prvky zahrnuté v relačných databázach rozšírené o ďalšiu funkcionalitu. Porovnanie poskytovaných prvkov je v tabuľke 1 Prvky poskytované MongoDB [9].

Tabuľka 1: *Prvky poskytované MongoDB [9]*

	MySQL	**MongoDB**
Bohatý dátový model	Nie	Áno
Dynamická štruktúra	Nie	Áno
Dátové typy	Áno	Áno
Lokálnosť dát	Nie	Áno
Aktualizovanie polí	Áno	Áno
Ľahké pre programátorov	Nie	Áno
Komplexné transakcie	Áno	Nie
Kontrola	Áno	Áno
Auto-sharding	Nie	Áno

MongoDB má vlastnú konvenciu názvov svojich častí. Tie sa v niektorých prípadoch líšia s názvami relačných databáz. Rozdiely sú zobrazene v tabuľke 2 Porovnanie konvencie názvov [7]. Za zástupcu relačných databáz bola vybraná MySQL databáza.

Tabuľka 2: *Porovnanie konvencie názvov [7]*

MySQL	**MongoDB**
Databáza	Databáza
Tabuľka	Kolekcia
Index	Index
Riadok	BSON dokument
Stĺpec	BSON pole (angl. field)
Spojenie	Vnorené dokumenty a prepojenie
Primárny kľúč	Primárny kľúč
Zoskupenie	Agregácia

5.1.3 Ostatné

Okrem relačných a textových dokumentov existuje ešte niekoľko druhov databáz. V nasledujúcich častiach si priblížime niektoré z nerelačných databáz.

5.1.3.1 Kľúč - hodnota databázy

Nerelačné databázy typu kľúč - hodnota sú v svojej podstate celkom jednoduché, ale zároveň efektívne. Umožňujú používateľovi ukladať dáta ľubovolne, kedže

neobsahujú schémy. Uložené dáta sa skladajú z dvoch častí. Prvá časť je kľuč a druhá časť je hodnota [10], pričom kľúč je samo-generujúci string a hodnota môže byť takmer čokoľvek, od string, JSON cez BLOB až po obrázok [7].

Kľúč - hodnota databázy sú veľmi podobné hašovacím tabuľkám, kde kľúč je indexom do tabuľky, pomocou ktorého používateľ môže pristúpiť k hodnote daného kľúču. Tento typ databáz uprednostňuje rozšíriteľnosť pred konzistenciou. Ponúka vysokú konkurenčnosť (angl. concurrency), rýchle vyhľadávanie a schopnosť uloženia veľkého množstva dát za cenu spojovacích a agregačných operácií. Taktiež je veľmi náročné vytvoriť ľubovolný pohľad na dáta z dôvodu chýbajúcej schémy [10].

Najznámejšími predstaviteľmi tohto typu databáz sú *Amazon DynamoDB* a *RIAK*.

5.1.3.2 Stĺpcové databázy

Stĺpcové databázy musia mať preddefinovanú schému, v ktorej sú jednotlivé bunky záznamov zoskupené do kolekcie stĺpcov [7]. Dáta nie su ukladané do tabuliek, ale do masívne distribuovaných architektúr, za hlavným zámerom, aby agregácia dát mohla prebehnúť veľmi rýchlo s redukovaním I/O aktivity.

Tento typ databáz taktiež poskytuje veľkú rozšíriteľnosť v ukladaní dát.

Najvhodnejšie je využívať takéto databázy v analytických aplikáciách alebo aplikáciach, ktoré získavajú dáta pomocou metódy *data mining* [10].

5.1.3.3 Grafové databázy

Grafové databázy su špeciálny typ databáz, v ktorých sú dáta uložene vo forme grafu. Graf pozostáva z vrcholov a hrán, pričom vrcholy predstavujú objekty a hrany reprezentujú vzťahy medzi nimi. Každý vrchol okrem iného obsahuje aj ukazovateľ na priľahlé vrcholy, čo umožňuje prechádzať obrovské množstvo dát rýchlejšie ako v relačných databázach [10].

Údaje sa ukladajú v polo-štruktúrovanej forme, kde je kladený hlavný dôraz na prepojenia medzi dátami. Grafové databázy spĺňajú vlastnosť ACID a sú veľmi vhodné pre biometrické aplikácie alebo aplikácie sociálnych sieti. Hlavným predstaviteľom grafových databáz je *Neo4j* [10].

5.1.3.4 Objektovo orientované databázy

Objektovo orientované databázy ukladajú dáta vo forme objektov, rovnako ako sú údaje reprezentované v objektoch v objektovo orientovaných programovacích jazykoch (OOP). Tieto databázy podporujú všetky vymoženosti OOP, ako enkapsulácia, polymorfizmus, ale aj dedenie. Objektovo orientované databázy robia moderný vývoj softvéru jednoduchším [10].

5.1.4 Zhrnutie

NOSQL databáza narozdiel do RDBMS modelu (Relation Data Base Management System) je navrhnutá aby sa bola jednoducho rozšíriteľná so zväčšovaním sa. Väčšina NOSQL databáz odstránila niektoré nepotrebné prvky RDBMS modelov, čím sa stali podstatne ľahšími a efektívnejšími ako ich náprotivok RDBMS systémy. Toto na druhej strane spôsobilo, že NOSQL model negarantuje vlastnosti ACID (Atomicity, Consistency, Isolation, Durability), ale naopak garantuje vlastnosti BASE (Basically Available, Soft state, Eventula Consistency) [10].

Nerelačné databázy neukladajú údaje v tabuľkách, nemajú fixnú schému a majú jednoduchý dátový model. Tieto vlastnosti im umožňujú jednoducho spracovávať neštrukturované dáta, ako sú dokumenty, e-maily a mnoho ďalších [7]. Tieto databázy majú čím ďalej, tým majú viacero využití.

Existuje hneď niekoľko prípadov, kedy je lepšie použiť nerelačnú databázu namiesto relačnej databázy. Keď je potrebné, aby aplikácia dokázala spracovávať rôzne typy a tvary dát alebo pri potrebe spravovať aplikáciu efektívnejšie pri rozširovaní, je rozhodne výhodnejšie použiť nerelačnú databázu. Niektoré databázy, ako napríklad textová databáza MongoDB uľahčuje vývoj aplikácií, keďže jeho dokumentová štruktúra dát je jednoducho namapovateľná na moderné, objektovo-orientované programovacie jazyky a tým pádom nie je potreba využívať komplexnú objektovo-relačnú mapovaciu vrstvu, ktorá je nutná pri použití relačných databáz na prevod objektov z programovacie jazyka na perzistentné objekty v databáze. Všeobecne je omnoho ľahšie rozšíriť schému / model nerelačnej databázy ako rozširovať schému relačnej databázy [9].

My budeme využívať textovú databázu, konkrétne MongoDB, na ukladanie spracovaných dát. Keďže skoro každý text a veta je odlišná, tak pre každý text a

vetu budeme ukladať, odlišné alebo odlišný počet dát, takže nebudeme mať fixne danú schému. Kvôli tomu, ale aj pre ostatné vlastnosti dokumentovej databázy MongoDB, podrobnejšie opísané v kapitole 5.1.2.1 MongoDB, bola voľba tejto databázy jednoznačná.

5.2 Náš návrh uchovávania textov v databázach

Dáta budeme ukladať v dokumentovej databáze MongoDB. Keďže spracovávané dáta sa dajú rozdeliť do troch kategórií, budeme využívať primárne tri databázové kolekcie na ich ukladanie. Sú to:

- sentences,
- rules,
- texts.

V nasledujúcich častiach ich opíšeme bližšie aj s názornými ukážkami.

5.2.1 Kolekcia texts

V kolekcií *texts* sa budú ukladať celé texty, ktoré budú spracovávané.

Schéma tejto kolekcie bude veľmi jednoduchá, nakoľko bude obsahovať iba jedno pole textového typu slúžiace na uloženie textu v pôvodnom tvare.

5.2.2 Kolekcia sentences

V ďalšej kolekcií *sentences* budeme ukladať spracovávané vety a vytvorené poznámky z týchto viet, pričom vety sa budú odkazovať na texty, z ktorých pochádzajú v kolekcií *texts*. Toto nám umožní jednoducho zistiť, v akom texte sa daná veta nachádzala.

Schéma tejto kolekcie bude taktiež pomerne jednoduchá. Dáta budu uložené v dokumentoch s jednoduchou štruktúrou. Bude obsahovať tri polia. Jedno, textové, určené na uchovanie pôvodného znenia vety, druhé, tiež textové, na uchovanie novo vytvorenej vety po spracovaní vety uloženej v prvom poli a tretie pole, ktoré bude odkazovať na záznam v kolekcií *texts*.

5.2.3 Kolekcia rules

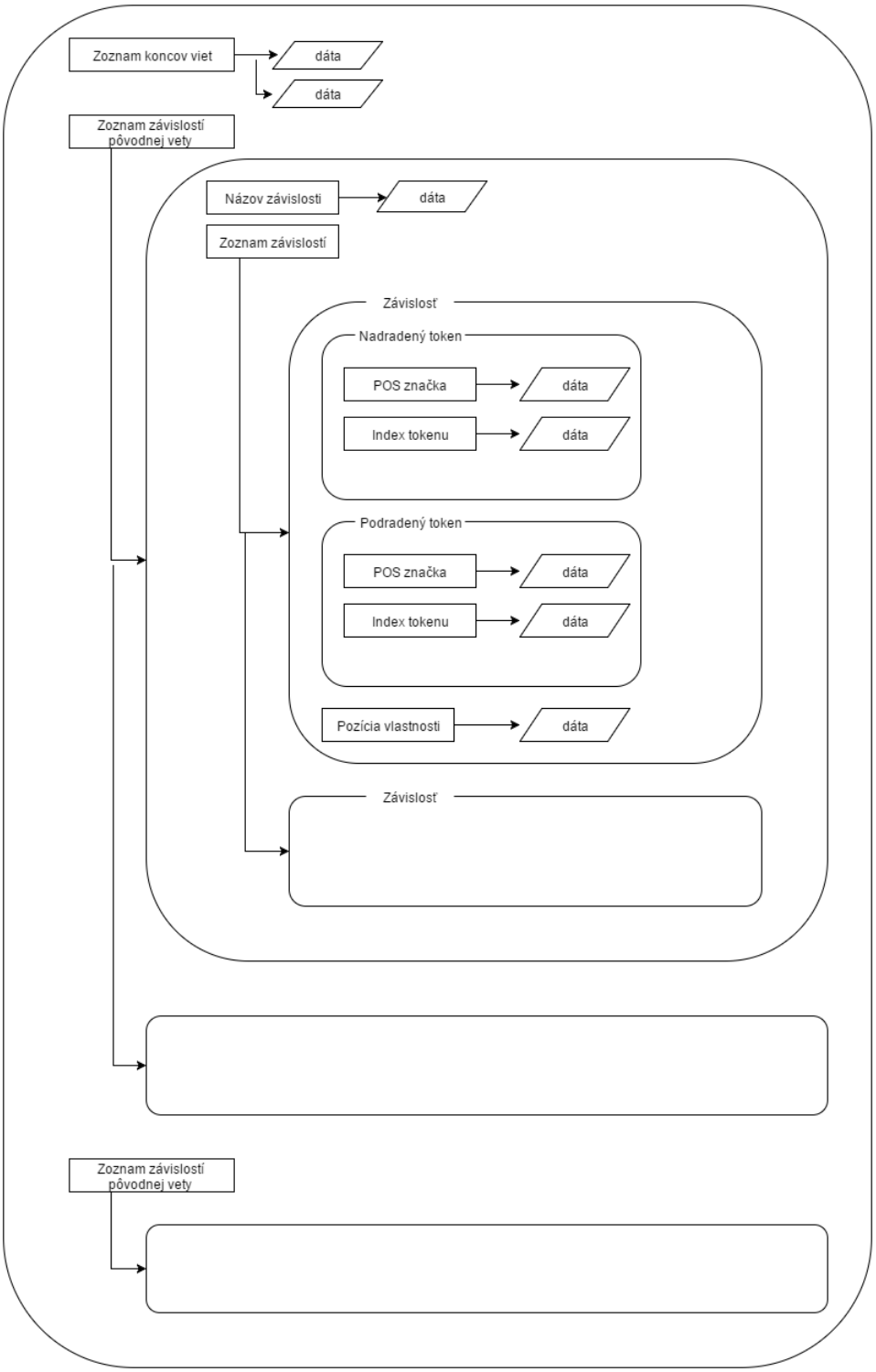
V poslednej kolekcií pomenovanej *rules* sa budú ukladať pravidla na spracovávanie viet, ktoré sa budú odkazovať na vety v kolekcií *sentences*, ktoré boli podľa daného pravidla spracované. Ukladaním viet a pravidiel na ich spracovanie do separátnych kolekcií zabránime duplikovaniu dát a zrýchlime vyhľadávanie. Referencia do kolekcie *sentences* nám poskytuje možnosť jednoduchého a rýchleho vyhľadanie viet, na ktoré bolo konkrétne pravidlo aplikované a aký bol výstup aplikovania tohto pravidla.

Pravidlo sa skladá hlavne z dvoch častí. Zoznam závislostí pôvodnej vety a zoznam závislostí zjednodušenej vety. Práve závislosti z druhého menovaného zoznamu sa aplikujú na spracovávanú vetu s cieľom zjednodušiť ju. Dáta uložené v tejto kolekcií budú mať zložitejšiu štruktúru.

Každý záznam bude obsahovať pole celých čísel určujúcich pozície slov, za ktorými je vo vytvorenej zjednodušenej vete ukončenie vety. V prípade jednoduchých viet to bude posledné slovo vety, ale pri súvetiach to môže byť viacero slov na ľubovolných miestach vety. Pre jednoduchú vetu *„The president of the Czech Republic is Miloš Zeman."* bude toto ple obsahovať jednú hodnotu 3, keďže zjednodušená veta bude v tvare *„President is Zeman."*.

Okrem poľa určujúceho konce viet, bude každý záznam obsahovať dva hlavné zoznamy závislostí. Prvý zoznam bude pozostávať zo závislostí pôvodnej vety a druhý zoznam bude zložený z závislostí zjednodušenej vety. Zoznamy budú mať nasledujúcu štruktúru. Budú obsahovať dokumenty. Tieto dokumenty budú mať názov vlastnosti a ich zoznam, pričom sa budú párovať práve podľa názvu. Tento vnorený zoznam bude obsahovať už konkrétne závislosti. Každá závislosť sa skladá z nadradeného tokenu (angl. governor), podradeného tokenu (angl. dependent) a pozície tejto závislosti medzi všetkými závislosťami vety. Tieto tokeny sú dokumenty skladajúce sa z dvoch polí, jedno textové, obsahujúce skratku POS značky a druhé číselne, obsahujúce pozíciu slova vo vete, ku ktorému sa daný token viaže.

Celý strom štruktúry dát v kolekcií *rules* sa dá vyjadriť diagramom 12 Štruktúra kolekcie rules.



Obr. 12: *Štruktúra kolekcie rules*

Dáta sú v MongoDB databáze uložené v binárnom JSON formáte. Na ukážke 2 Ukážka dát kolekcie rules je zobrazená časť uložených údajov o pôvodnej vete.

Ukážka 2: *Ukážka dát kolekcie rules*

```
{
  "originalDependencies" : [
    {
      "dependencyName" : "det",
      "dependencies" : [
        {
          "governor" : {
            "pos" : "NN",
            "index" : 2
          },
          "dependent" : {
            "pos" : "DT",
            "index":1
          },
          "position" : 0
        },
        { ... }
      ]
    }
  ]
}
```

5.2.4 Vyhľadávanie pravidla

Pred spracovaním vety sa vyhľadá pravidlo v databáze vhodné na jej zjednodušenie. Pri vyhľadávaní sa berie do úvahy viacero podmienok.

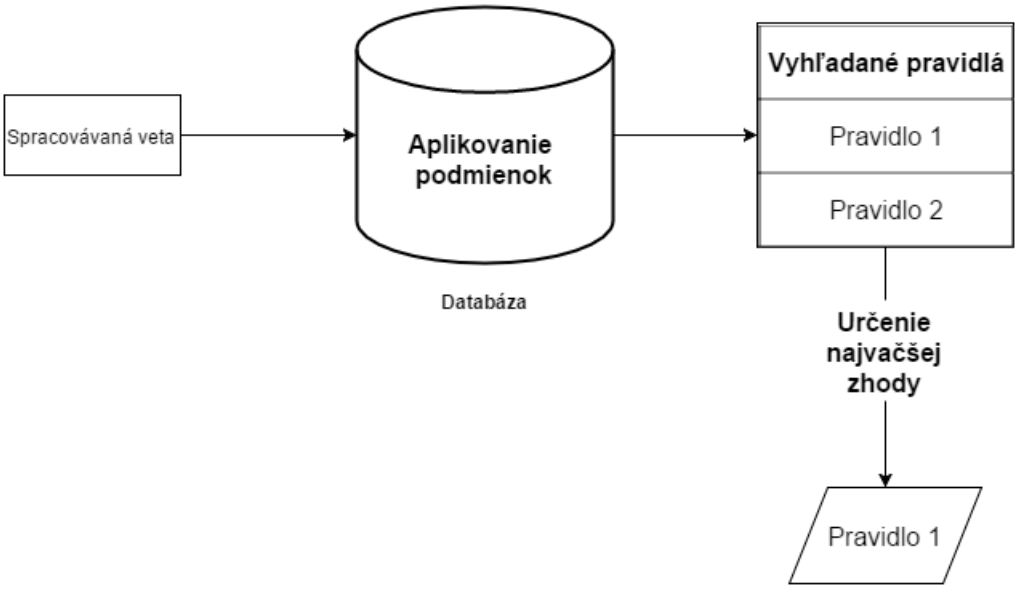
Spracovávaná veta, pre ktorú hľadáme pravidlo, musí mať rovnaký počet záznamov v *zozname závislosti pôvodnej vety* a zároveň musia byť napárované práve všetky názvy závislosti v tomto zozname.

Pri použití týchto podmienok vieme rýchlo vyhľadať pravidlo, ktoré súvisí s podobnou vetou. Avšak, môže nastať situácia, kedy je pre spracovávanú vetu vhodných viacero pravidiel. Vtedy sa rozhoduje podľa zhody pôvodných viet, ktoré vybrať. Vyberá sa, a následne aplikuje, to s najväčšou zhodou.

Určovanie najväčšej zhody má viacero krokov. Najskôr sa spočítavajú zhody POS značiek nadradených a podradených tokenov zvlášť a následne, indexy slov prislúchajúcich tokenom taktiž nezávisle od seba. Tým sa zisťuje, či spracovávaná veta obsahuje ľubovolnú závislosť s rovnakou hodnotou POS značky alebo indexu či už nadradeného alebo podradeného tokenu. V druhom kroku sa určuje polovičná zhoda závislosti, teda či spracovávaná veta obsahuje zhody POS značiek a zároveň indexov slov v nadradenom tokene alebo v podradenom tokene. V poslednom, treťom sa zisťuje počet úplných zhôd závislostí, čo znamená zhoda POS značiek a indexov zároveň, v nadradenom a podradenom tokene zároveň. Tieto tri hodnoty sa na záver spočítajú a tým získame percentuálne ohodnotenie zhody viet.

Toto určovanie najväčšej zhody sa uskutoční pre každé vyhovujúce pravidlo a vyberie sa pravidlo s najväčšou zhodou.

Ukážkový proces vyhľadania pravidla a určenie zhody je zobrazený na obrázku 13 Vyhľadanie pravidla.



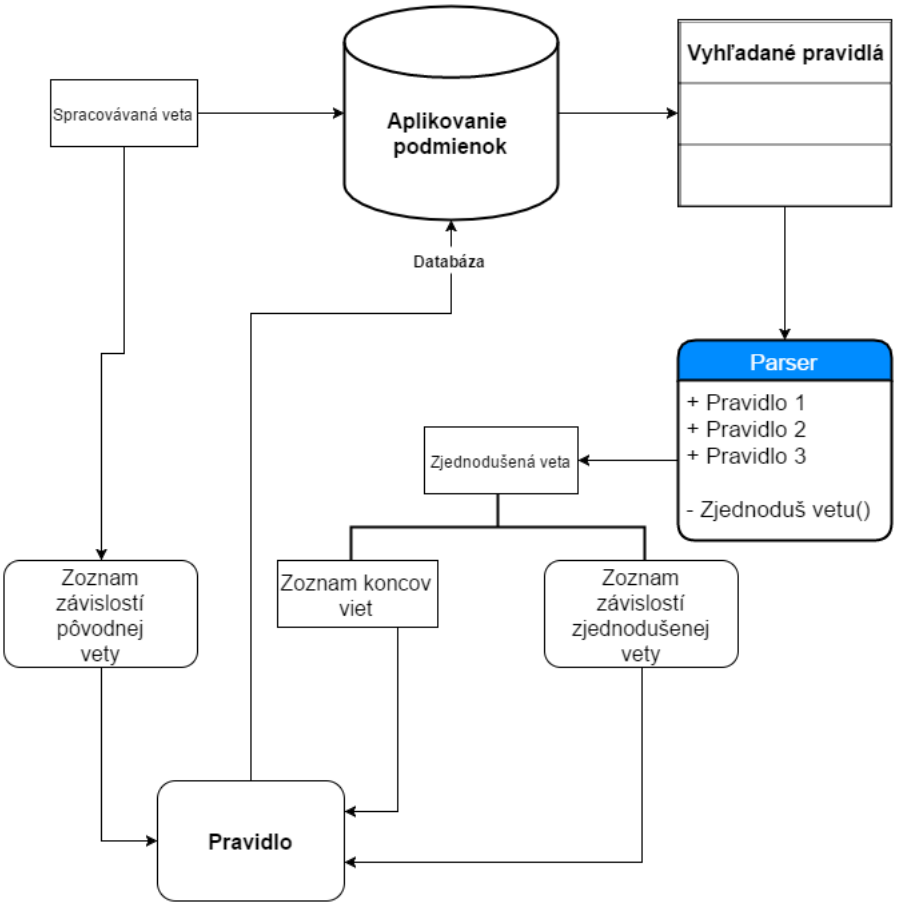
Obr. 13: *Vyhľadanie pravidla*

5.2.5 Vytváranie pravidla

Ak nám proces vyhľadania pravidla nevyhľadal žiadne pravidlo, znamená to, že sme doposiaľ nespracovávali takú istú alebo podobnú vetu. V tomto prípade použijeme náš parser, ktorý operuje nad staticky danou sadou pravidiel. Výstupom parseru bude zjednodušená veta, ktorej pravidlo sa následne uloží do databázy a pri ďalšom spracovávaní takej istej alebo podobnej vety sa toto pravidlo vyhľadá a aplikuje ak bude mať dostatočné veľkú zhodu.

Zo závislostí pôvodnej vety sa vytvorí *zoznam závislostí pôvodnej vety*, zo závislostí zjednodušenej vety sa vytvorí *zoznam závislostí zjednodušenej vety* a zo zjednodušenej vety sa určia aj konce viet. Tieto informácie sa spolu uložia do dokumentu, záznamu, do databázy.

Na obrázku 14 Vytvorenie pravidla je znázornený proces nevyhľadania pravidla, použitie parsera s následným uložením nového pravidla.



Obr. 14: *Vytvorenie pravidla*

5.2.6 Aplikovanie pravidla

6 Results

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

6.1 Subsection

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare quam littera gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.

7 Conslusions

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi sit amet arcu. Fusce pharetra dapibus elit. Duis malesuada. Proin at elit vitae quam cursus tristique. Quisque fermentum. Praesent dictum. Nullam vehicula. Nunc pharetra dolor ut velit. Sed pulvinar, est sed congue tempor, nibh arcu cursus enim, quis consequat magna lacus sed pede. In sagittis. Etiam volutpat, velit id tincidunt egestas, augue ligula auctor eros, sit amet viverra sapien tortor at odio. In diam libero, fringilla ut, adipiscing condimentum, ultricies at, dui. Phasellus vitae risus.

Pellentesque vulputate ante ut diam. Sed adipiscing malesuada odio. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nam a leo. Praesent velit. Aenean vehicula accumsan quam. Nulla dolor lorem, imperdiet a, ullamcorper hendrerit, ultrices at, urna. Integer placerat ligula id purus. Sed id nisl. Pellentesque tincidunt neque in lacus. In non quam et felis suscipit viverra.

Literatúra

[1] Akshar Bharati and Vineet Chaitanya. *Natural language processing: A Paninian perspective*. Prentice Hall of India, New Delhi, 2004.

[2] Volha Bryl, Claudio Giuliano, Luciano Serafini, and Katerina Tymoshenko. Supporting natural language processing with background knowledge: Coreference resolution case. In *9th International Semantic Web Conference (ISWC2010)*, November 2010.

[3] Marie catherine De Marneffe and Christopher D. Manning. Stanford typed dependencies manual, 2008.

[4] Randee Tengi Christiane Fellbaum. What is wordnet?

[5] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[6] Ann Copestake. Natural language processing. 2004.

[7] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah. A comparative study: Mongodb vs. mysql. In *Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on*, pages 1–6, June 2015.

[8] David Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[9] Inc MongoDB. Mongodb and mysql compared.

[10] Ameya Nayak, Anil Poriya, and Dikshay Poojary. Article: Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, 5(4):16–19, March 2013. Published by Foundation of Computer Science, New York, USA.

[11] Preeti and BrahmaleenKaurSidhu. Natural language processing. *Int.J.Computer Technology & Applications*, 2013.

A Technical documentation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

A.1 Implementation

A.1.0.1 Modul abc

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

A.1.0.2 Modul def

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius. Claritas est etiam processus dynamicus, qui sequitur mutationem

consuetudium lectorum. Mirum est notare quam littera gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.

B User documentation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

B.1 Instalation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

B.2 Run the application

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

C Electronic medium

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat:

/Application

- implementácia opisovaného riešenia

/Documentation

- bakalárska práca spolu s anotáciami v slovenskom a anglickom jazyku

/Documentation/Latex

- latex zdrojové súbory dokumentácie

/Documentation/BibTeX

- BibTeX súbor s použitými referenciami

/Documentation/Resources

- dostupné použité zdroje

/Resources

- vstupne/testovacie dáta opisované v dokumente

/Source/Dependencies

- inštalačné súbory pre knižnice, ktoré potrebuje aplikácia

read.me - popis obsahu média v slovenskom a anglickom jazyku