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
Vedúci práce: Ing. Miroslav Blšták

December 2015

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 podoby, s ktorou vedia aj stroje jednoducho narábať. Takýto proces sa nazýva *spracovanie prirodzeného jazyka* (angl. Natural Language Processing - NLP). Spracovanie prirodzeného jazyka má 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 spoznámkovávanie.

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 Spracovanie prirodzeného jazyka

V tejto kapitole priblížime a rozoberieme čo je spracovanie prirodzeného jazyka, jeho využitie v aplikáciach a systémoch a jeho hlavné úlohy. Ďalej zanalyzujeme nástroje, ktoré sa dajú využiť na spracovanie prirodzeného jazyka.

2.1 Spracovanie prirodzeného jazyka

Spracovanie prirodzeného jazyka (angl. Natural Language Processing - NLP) odkazuje na počítačové systémy, ktoré spracovávajú, snažia sa pochopiť alebo generujú jeden alebo viacero ľudských jazykov. Vstupom môže byť text alebo hovorená reč s cieľom prekladu do iného jazyka, pochopenie a reprezentácia obsahu textu, udržanie dialógu s používateľom a iné [1]. Počítače doposiaľ nedokážu plne porozumieť ľudskému jazyku, či už sa jedná o písaný alebo hovorený, a preto hlavným cieľom NLP je vybudovať výpočtové modely prirodzeného jazyka pre jeho analýzu a generovanie [2].

Porozumenie ľudskej reči je mnohokrát náročné 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 po-

hľadu *extrakcie informácií*, ktoré je podrobnejšie popísané v sekcii 2.2.1 Extrakcia informácií. Ďalšie využitia NLP sú napríklad [9]:

- Strojový preklad (angl. Machine Translation)
- Rozpoznávanie reči (angl. Speech Recognition)
- Sumarizáciu textu (angl. Text Summarization)
- Dialógové systémy (angl. Dialogue Systems)
- Vyhľadanie 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 [9].

Extrakcia informácií využíva niekoľko z hlavných úloh spracovania prirodzeného jazyka. Sú to *Značkovanie slovných druhov*, *Rozpoznávanie názvoslovných entít*, a ďalšie [9]. 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.

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 singulár 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 Identifikácia koreferencií

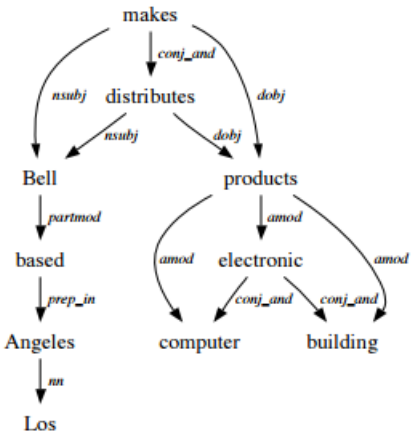
Nájdenie, identifikácia a rozpoznanie koreferencií v texte je úlohou rozpoznávania koreferencií (angl. Coreference resolution). V texte sa často používajú zámena (angl. pronouns) *to*, *tí*, *on*, anglicky *it*, *those*, *he* alebo menné frázy (angl. noun phrase). Tieto zámena a menné frázy sa odkazujú 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 jazyka sa využíva v aplikáciách NLP ako sú extrakcia informácií (viď. 2.2.1 Extrakcia informácií) a odpovedanie na otázky [3].

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 Identifikácia gramatických závislostí

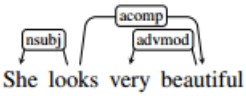
Rozloženie na vzťahy nám poskytuje jednoduchý opis gramatických vzťahov slov vo vete. Aplikovaním rozloženia vzťahov na vetu *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.* vznikne strom vzťahov (angl. dependency tree) (viď. obrázok 1 Strom vzťahov) [4].



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 závislostí 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 [4].



Obr. 2: *Vzťahy vo vete*

Celá závislosť sa skladá primárne z nadradeného tokenu, podradeného tokenu a vzťahu medzi nimi. Na obrázku 2 Vzťahy vo vete vidno, okrem iných aj závislosť, ktorej nadradený token je slovo *looks*, podradený token je slovo *She* a vzťah je *nsubj*.

2.4 Nástroje na spracovanie prirodzeného jazyka

V súčasnosti 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[1] je databáza anglických slov vyvíjaná na Princetonskej univerzite. Databáza obsahuje podstatné mena, prídavné mená, slovesá a príslovky, ktoré sú zatriedené do synonymických sád, synsetov.

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

Vo WordNet 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 pre 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.

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.

[1] www.wordnet.princeton.edu

Obr. 3: *Webové rozhranie*

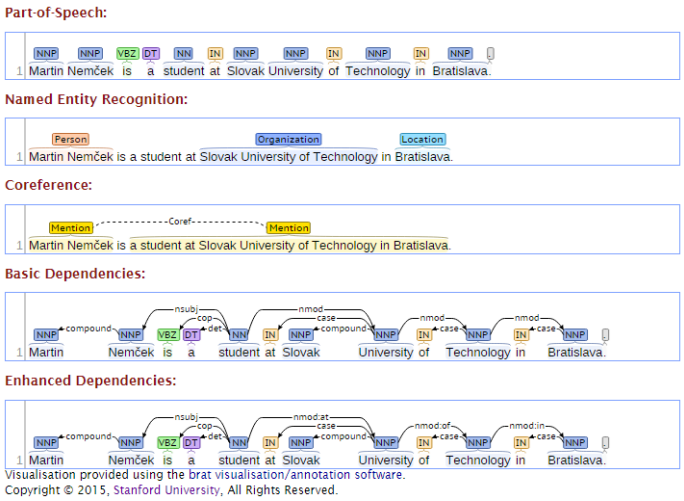
Obr. 4: *Nadradenost'slov*

2.4.2 StanfordNLP

Nástroj StanfordNLP[2] 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.

Dostupné 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[3] je nástroj vytvorený na Cambridge univerzite. Umožňuje prístup k viacerým rôznym slovníkom. Momentálne tento nástroj ponúka prístup k pätnástim

[2] www.nlp.stanford.edu
[3] www.dictionary-api.cambridge.org

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.

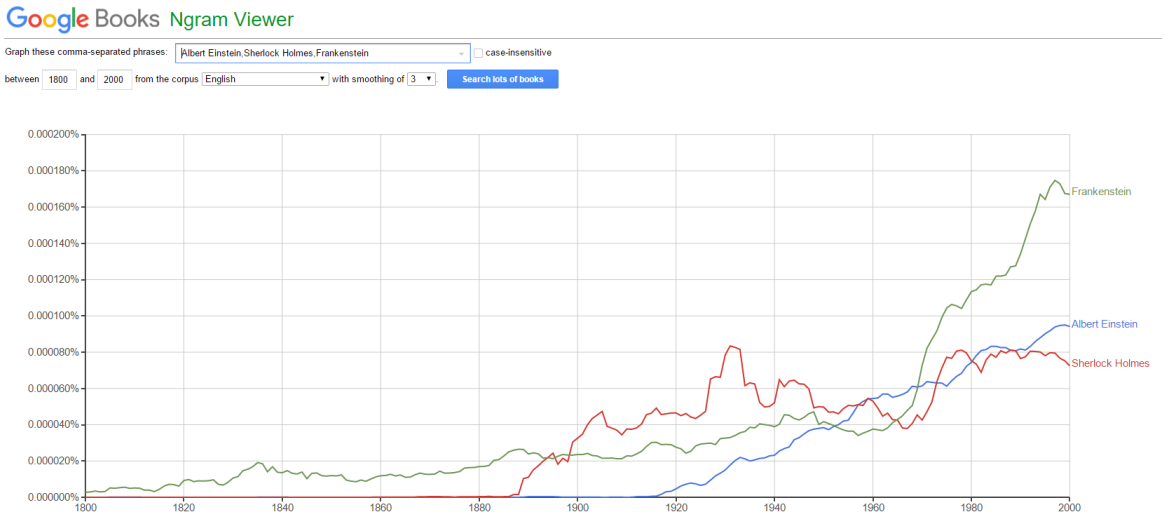
2.4.4 Google Ngram

Google Ngram[4] 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 ľubovoľné 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.

[4] www.books.google.com/ngrams



Obr. 6: *Google Ngram Viewer*

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

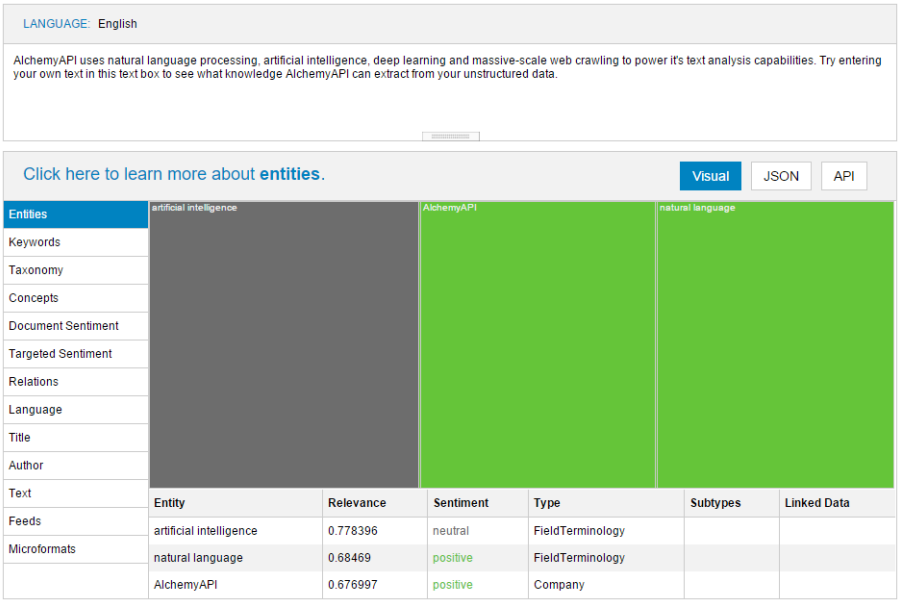
2.4.5 AlchemyAPI

AlchemiAPI[5] 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. 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, obsiahnutú v tomto nástroji.

[5] www.alchemyapi.com



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 tvárí na obrázkoch.

3 Analýza nástrojov na správu paralelných textov

Dostupnosť aplikácií na spracovanie prirodzeného jazyka je veľká a široká. Najväčší podiel tvoria aplikácie zamerané na preklad. My sa zameriame na aplikácie, ktoré umožňujú editovať paralelný text.

Nástroje na správu paralelných textov uľahčujú spracovanie viacerých druhov a verzií textu. Na jednej strane majú zdrojový text alebo súbor a na druhej strane výsledný text alebo súbor. Hlavný dôraz sa kladie práve na transformáciu zo zdrojového textu na cieľový. Transformácia môže mať viacero podôb, ako preklad, zarovnanie alebo zjednodušenie textu, a mnoho ďalších. Texty sú zväčša rozdelené podľa viet, pre zjednodušenie transformácie, pričom vety na jednej úrovni zvyčajne spolu súvisia podľa určitej vlastnosti.

V nasledujúcich častiach si predstavíme niektorých predstaviteľov tohto druhu nástrojov.

3.1 InterText

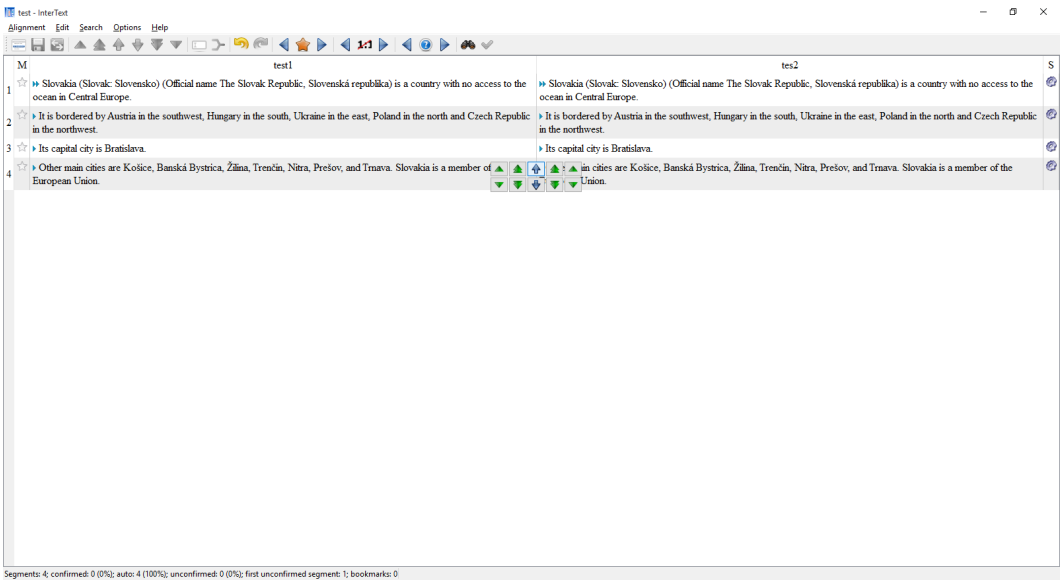
InterText[6] je editor paralelne zarovnaných textov, využívaný na správu viacerých paralelne zarovnaných verzií textu rôznych jazykov na úrovni viet. Táto aplikácia je dostupná vo verzií pre desktop, ale aj pre server.

Podporuje viacero formátovaní textu, či už čistý (angl. plain) text alebo XML a taktiž zobrazuje aj HTML značky. Riadky obsahujú vety oddelené znakom konca 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 zjednodušovanie textu v tomto nástroji by bolo pomerne náročné.

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.

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



Obr. 8: *Aplikácia InterText*

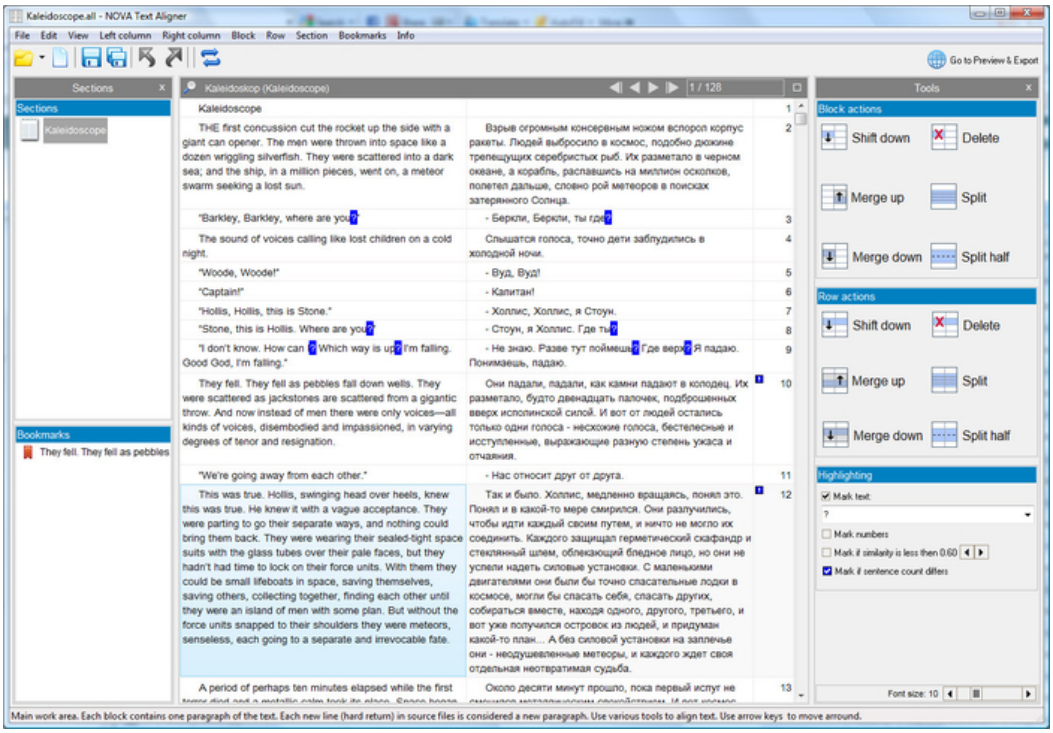
3.2 NOVA Text Aligner

NOVA Text Aligner[7] 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 zvýraznenie 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 typu 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 a funguje iba lokálne. NOVA Text Aligner je dostupná iba v skúšobnej verzií, pre dlhodobé používanie si treba zakúpiť licenciu.

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

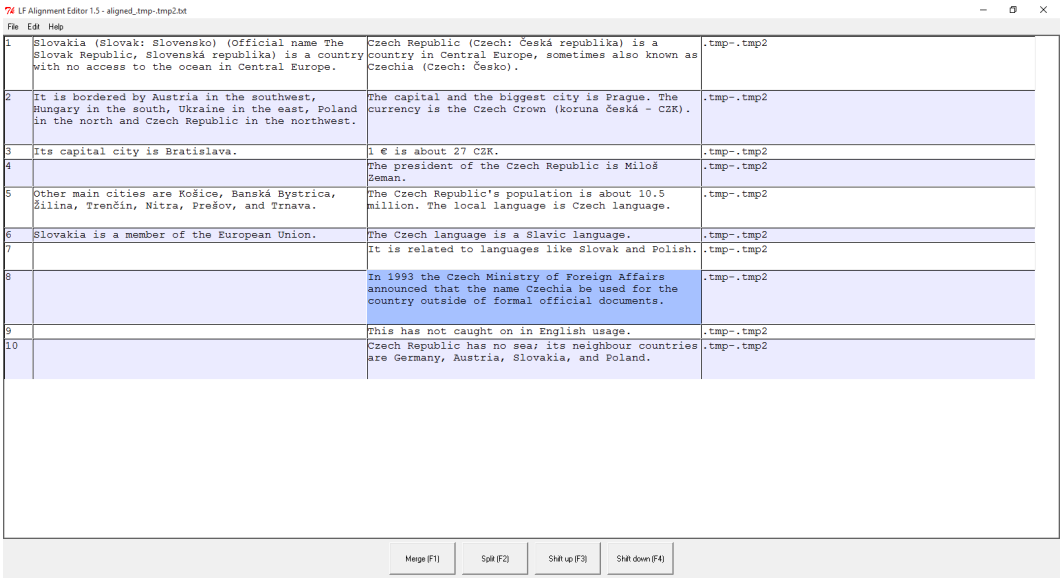


Obr. 9: *Aplikácia NOVA Text Aligner*[8]

3.3 LF Aligner

Aplikácia LF Aligner[9] je zameraná na spracovanie textu rôznych jazykov. Ponúka možnosť použiť až 99 jazykov, čo ale znamená 99 vstupných súborov, každý so 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. Samozrejmosťou je možnosť premiestňovať a zoskupovať riadky, doplnenie ďalšieho súboru na spracovávanie, uloženie zmien súboru prepísaním jeho dát a mnohé ďalšie.

[8] http://parallel-text-aligner.en.softonic.com/
[9] www.sourceforge.net/projects/aligner



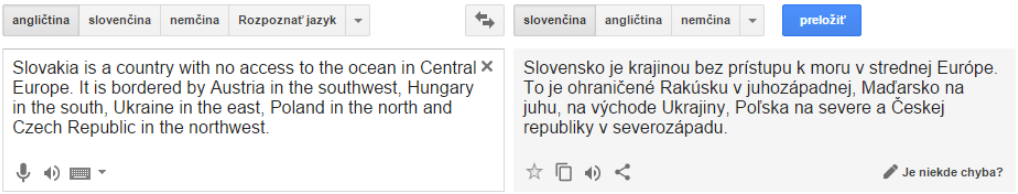
Obr. 10: *Aplikácia LF Aligner*

3.4 Google Translate

Za najznámejšieho zástupcu webových nástrojov na spracovanie paralelných textov sa dá pokladať nástroj Google Translate[10]. Využíva sa na preklad slov, viet, ale dokáže spracovať aj celé texty. Momentálne podporuje preklad z a do 91 jazykov. Dokáže rozpoznať a preložiť hovorenú reč aj písaný text. Pri preklade jednotlivých slov zobrazuje viacero možných prekladov do druhého jazyka, pričom pri preklade z anglického jazyka ponúka aj ukážky viet, v ktorých sa prekladané slovo môže použiť. Správnu výslovnosť preloženého aj prekladaného slova alebo textu, si používateľ môže vypočuť na krátkej zvukovej ukážke.

Na obrázku 11 Google Translate je zobrazený preklad anglického textu do slovenského. Vidno, že preklad do minoritných jazykov ešte nie je dokonalý.

[9] http://parallel-text-aligner.en.softonic.com/
[10] translate.google.com



Obr. 11: *Google Translate*

3.5 Zhrnutie

Analyzovali sme aplikácie, ktoré umožňujú spravovať a editovať paralelný text. Za ich pomoci dokážeme zo vstupného textu získať výstupný text. Napríklad pri preklade máme vstupný text množinu viet v anglickom jazyku, ktorú chceme preložiť do slovenského jazyka a výstupný text je preloženú množinu viet. Pri zjednodušovaní textu je na vstupe taktiež množina viet a na výstupe je každá veta zo vstupnej množiny zjednodušená podľa istých pravidiel. Výstupný text vzniká určitou transformáciou vstupného textu, aplikovaním transformácie na každú vetu zdrojového textu.

Analyzované nástroje nespĺňajú všetky požiadavky na systém schopný spoznámkovať učebný text v takom rozsahu, ktorý by umožňoval používateľovi prispôsobiť si spracovaný text. Systém musí umožňovať editáciu jednotlivých viet výstupného textu podľa vôle používateľa. Tieto úpravy musí zohľadniť pri následnej aplikácií transformácií vstupného textu. Dáta ohľadne spoznámkovávania textu musí ukladať mimo používateľovho úložného priestoru.

4 Smerovanie práce

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

vety, takže používateľovi zabráni zadávať ľubovoľné a nesprávne slová. Predíde sa tým nezmyselným záznamom v databáze.

Do systému doimplementovať „AND poznámkovač", ktorý identifikuje množinu súvisiacu so spojkou AND a pre každú entitu v tejto množine oddelenú čiarkou vygeneruje samostatnú poznámku.

Následne plánujem dokončiť kapitolu návrh, napísať kapitoly záver, anotácia, výsledky a ďalšie.

5 Opis prototypu

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

5.1 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);
```

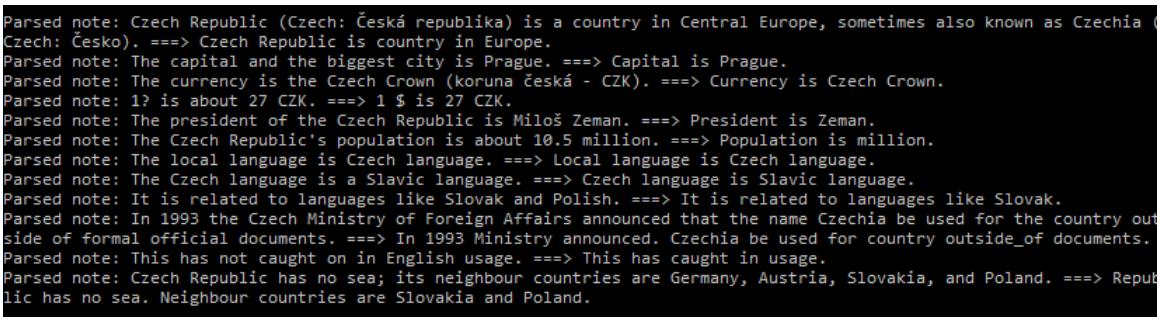
Ukážka 1: *Spustenie StanfordCoreNLP*

Údaje získané 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 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 12 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. 12: *Ukážkový výstup prototypu*

5.1.1 Pravidlá

Pri spracovaní pôvodnej vety sa na túto vetu aplikuje *pravidlo na spracovanie*. Toto pravidlo obsahuje okrem iného zoznam závislostí pôvodnej vety. 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. Pre podrobnejšie informácie o vyhľadávaní a vytváraní pravidiel viď. sekcie 6.3.1 Vyhľadávanie pravidla a 6.3.2 Vytváranie pravidla

Obsahuje aj zoznam závislostí zjednodušenej vety - poznámky. Tieto závislosti sa aplikujú na spracovávané vety na vygenerovanie zjednodušenej vety (viď. 6.3.3 Aplikovanie pravidla).

6 Návrh

6.1 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.

6.1.1 Relačné databázy

Relačné databázy boli dlhé roky populárnou a finančné nenáročnou voľbou pri tvorbe veľkých podnikateľských aplikácií. Momentálne sú používané vo väčšine súčasných aplikácií a pracujú spoľahlivo pri obmedzenom množstve dát [6]. 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 [8].

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 [7].

6.1.2 Textové databázy

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

Textové databázy ukladajú dáta vo forme dokumentov, vďaka čomu ponúkajú vysoký výkon a horizontálnu rozšíriteľnosť [8]. Uložené dokumenty môžu nadobúdať rôzne typy, ako napríklad JSON, BSON, XML a BLOB, ktoré poskytujú veľkú flexibilnosť pre dáta. Každý záznam v takejto databáze preto môže mať inú štruktúru, napríklad počet alebo typ polí, čo šetrí úložným priestorom, keďže

neobsahuje nepotrebné prázdne polia [8].

Dokumenty v databáze sú referencované kľúčom, ktorý môže byť string, cesta, ale dokonca aj dokument [8]. 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.

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.

MongoDB

MongoDB[11] je dokumentová nerelačná databáza vytvorená v C++ spustená v roku 2009 [8]. 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 alebo zmenou typu polia dokumentu určujúceho štruktúru. Umožňuje jednoduché ukladanie dát s hierarchickými vzťahmi alebo komplexnejších štruktúr, ako sú napríklad polia, listy alebo vnorené polia.

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* [8].

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

[11] www.mongodb.org

Tabuľka 1: *Prvky poskytované MongoDB*

	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
Audit	Áno	Áno
Auto-sharding	Nie	Áno

Bohatý dátový model (angl. Rich Data Model) znamená, že dátový model poskytuje veľa funkcionality. Princípom dynamickej štruktúry (angl. Dynamic Structure) je jednoduchá zmena štruktúru, pričom nemusí byť vôbec zadefinovaná a každý záznam môže mať odlišnú štruktúru. Lokálnosť dát (angl. Data Locality) znamená uchovávanie súvisiacich dát pokope. Aktualizovanie polí umožňuje vykonávať nad poliami operácie, ako sú inkrementácia podľa špecifikovaného množstva, vynásobenie hodnotou, premenovanie, aktualizácia iba ak je hodnota väčšia alebo menšia ako špecifická hodnota a ďalšie. Audit (angl. Auditing) je funkcionalita, ktorá umožňuje administrátorom a používateľom sledovať aktivity systému. Auto-sharding pri náraste dát, aby sa zabránilo poklesu priepustnosti operácií čítania a zapisovania, ukladá dáta automaticky na viacero strojov.

MongoDB má vlastnú konvenciu názvov svojich častí. Tie sa v niektorých prípadoch líšia s názvami relačných databáz. Rozdiely sú zobrazene v tabuľke 2 Porovnanie používaných pojmov [6]. Za zástupcu relačných databáz bola vybraná MySQL databáza.

Tabuľka 2: *Porovnanie používaných pojmov [6]*

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

6.1.3 Ostatné databázové systémy

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

Kľúč - hodnota databázy

Nerelačné databázy typu kľúč - hodnota sú v svojej podstate celkom jednoduché, ale zároveň efektívne. Umožňujú používateľovi ukladať dáta ľubovoľne, kedže neobsahujú schémy. Uložené dáta sa skladajú z dvoch častí. Prvá časť je kľuč a druhá časť je hodnota [8], pričom kľúč je samo-generujúci string a hodnota môže byť takmer čokoľvek, od string, JSON cez BLOB až po obrázok [6].

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 ľubovoľný pohľad na dáta z dôvodu chýbajúcej schémy [8].

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

Stĺpcové databázy

Stĺpcové databázy musia mať preddefinovanú schému, v ktorej sú jednotlivé bunky záznamov zoskupené do kolekcie stĺpcov [6]. Dáta nie sú ukladané do tabuliek, ale do masívne distribuovaných architektúr, s 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 stĺpcové databázy v analytických aplikáciách alebo aplikáciách, ktoré získavajú dáta pomocou metódy *data mining* [8].

Grafové databázy

Grafové databázy su špeciálny typ databáz, v ktorých sú dáta uložené 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 [8].

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 sietí. Hlavným predstaviteľom grafových databáz je *Neo4j* [8].

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 [8].

6.1.4 Zhrnutie

NOSQL databáza narozdiel do RDBMS modelu (Relation Data Base Management System) je navrhnutá, aby 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) [8].

Nerelačné databázy neukladajú údaje v tabuľkách a nemajú fixnú schému. Tieto vlastnosti im umožňujú jednoducho spracovávať neštruktúrované dáta, ako sú dokumenty, e-maily a mnoho ďalších [6]. Preto majú čím ďalej, tým viac 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.

Potrebujeme ukladať v databáze texty a informácie o nich. Počet viet, slov, vzťahov je pre každý text odlišný a preto nedokážeme vopred definovať efektívnu schému na ukladanie týchto dát. Textové databázy, so svojou dynamickou a ľahko upraviteľnou štruktúrou, sú na tento účel ideálne, pričom ukladanie textov v tabuľkách by bolo náročne, navyše relačné databázy nemajú default podporované vyhľadávania v štruktúrach ako text. MongoDB (viď. 6.1.2 MongoDB) je vyspelá, textová databáza zahrňajúca všetku funkcionalitu, ktorú potrebujeme navyše rozšírenú o veľa vymožeností.

6.2 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.

Pri návrhu sme vychádzali z princípu čo najjednoduchších kolekcií, ktoré budu obsahovať iba relevantné informácie. V nasledujúcich častiach ich opíšeme bližšie aj s názornými ukážkami.

6.2.1 Kolekcia texts

V kolekcií *texts* sa ukladajú celé texty, ktoré sú spracovávané.

Kolekcia obsahuje iba jedno pole textového typu slúžiace na uloženie textu v pôvodnom tvare. Štruktúra uložených dát v kolekcií *texts* je zobrazená na obrázku 13 Štruktúra kolekcie texts.

Pole	Typ
_id	ObjectId
text	String

Obr. 13: *Štruktúra kolekcie texts*

6.2.2 Kolekcia sentences

V ďalšej kolekcii *sentences* ukladáme spracovávané vety a vytvorené poznámky z týchto viet, pričom vety sa odkazujú na texty, z ktorých pochádzajú v kolekcií *texts*. Umožní nám to jednoducho zistiť, v akom texte sa daná veta nachádzala.

Dáta sú uložené v dokumentoch, ktoré obsahujú 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 kolekcii *texts*. Štruktúra dát v tejto kolekcií je načrtnutá na obrázku 14 Štruktúra kolekcie sentences.

Pole	Typ
_id	ObjectId
texts	Array of ObjectIds
originalSentence	String
note	String

Obr. 14: *Štruktúra kolekcie sentences*

6.2.3 Kolekcia rules

V poslednej kolekcii pomenovanej *rules* sa ukladajú pravidla na spracovávanie viet, ktoré sa odkazujú na vety v kolekcii *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.

Každý záznam v tejto kolekcií obsahuje 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 pole obsahovať jednú hodnotu 3, keďže zjednodušená veta bude v tvare *„President is Zeman."*. Pre zloženú vetu v tvare *„Czech Republic has no sea; its neighbour countries are Germany, Austria, Slovakia and Poland."* bude spomínané pole obsahovať dve hodnoty, keďže táto veta sa skladá z dvoch. Prvá obsahujúca informáciu o mori a druhá s informáciou o susedných štátoch, a tak sa aj spracuje pri zjednodušovaní.

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 zo závislostí zjednodušenej vety. Zoznamy majú

nasledujúcu štruktúru. Obsahujú dokumenty. Tieto dokumenty majú názov vzťahu závislosti a ich zoznam, pričom sa párujú práve podľa názvu. Tento vnorený zoznam obsahuje už konkrétne závislosti. Každá závislosť uložená v databáze sa skladá z nadradeného tokenu (angl. governor), podradeného tokenu (angl. dependent) a pozície tejto závislosti medzi všetkými závislosťami vety. 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.

Celá štruktúra dát v kolekcii *rules* sa dá vyjadriť diagramom 15 Štruktúra kolekcie rules.

Pole	Typ
_id	ObjectId
sentences	Array of ObjectIds
sentenceEnds	Array of Integers
originalDependencies	Array of Documents
	Pole: dependencyName — Typ: String
	Pole: dependencies — Typ: Array of Documents
	Pole: governor — Typ: Document (Pole: POS — Typ: String; Pole: index — Typ: Integer)
	Pole: dependent — Typ: Document (Pole: POS — Typ: String; Pole: index — Typ: Integer)
	Pole: position — Typ: Integer
noteDependencies	Array of Documents (rovnako ako originalDependencies)

Obr. 15: *Štruktúra kolekcie rules*

Dáta sú v MongoDB databáze uložené v binárnom JSON formáte. Na ukážke 5 Ukážka dát kolekcie rules je zobrazená časť uložených údajov o pôvodnej vete. Ukážka celého záznamu pre vetu „The president of the Czech Republic is Milos Zeman.” je priložená v prílohe C Ukážka celého záznamu.

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

Ukážka 2: *Ukážka dát kolekcie rules*

6.3 Manažment dát

V nasledujúcich častiach si priblížime narábanie s dátami z databázy, ako vyhľadanie pravidla, jeho aplikovanie alebo vytvorenie pravidla, ak žiadne nebolo vyhľadané.

6.3.1 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 vzťahov v závislostiach v tomto zozname.

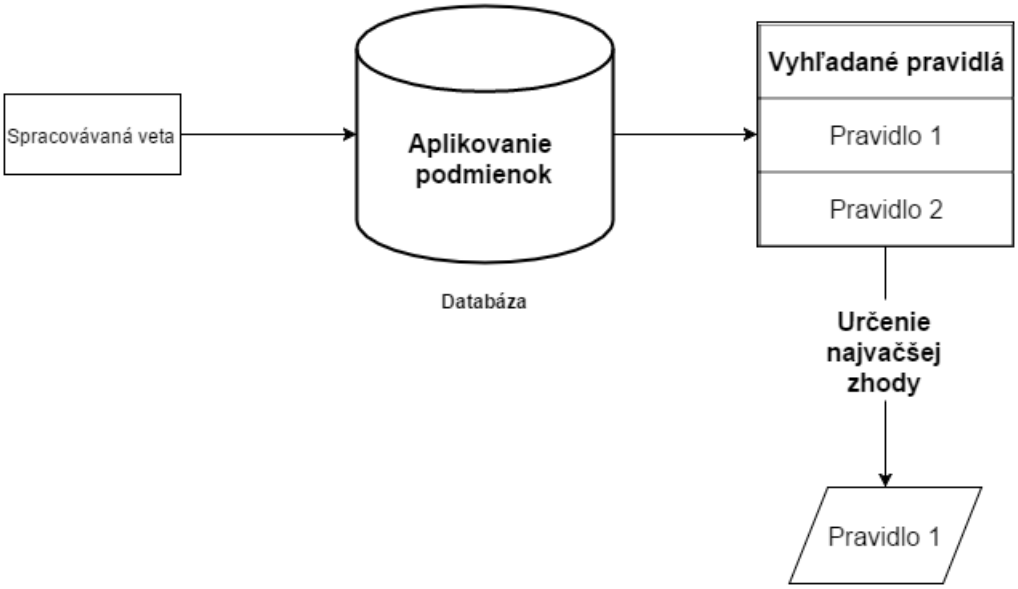
Pri použití týchto podmienok vieme rýchlo vyhľadať pravidlo, ktoré súvisí s podobnou vetou. Avšak, môže nastať situácia, kedy je pre spracovávanú vetu vhodných viacero pravidiel. Vtedy sa rozhoduje podľa zhody pôvodných viet, ktoré vybrať. Vyberá sa, a následne aplikuje, to s najväčšou zhodou.

Určovanie najväčšej zhody má viacero krokov. Najskôr sa spočítavajú zhody POS značiek nadradených a podradených tokenov zvlášť a následne, indexy slov prislúchajúcich tokenom taktiež nezávisle od seba. Tým sa zisťuje, či spracovávaná veta obsahuje ľubovoľnú 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.

Predpokladajme situáciu kedy spracovávame vetu „The local language is Czech language." a v databáze máme, okrem iného, uložené pravidlá pre vety „The local language is Czech language" a „The Czech language is a Slavic language.". Vtedy nastane, že pre spracovávanú vetu je vhodných viacero pravidiel. Keďže obe vety majú v *zoznam závislostí pôvodnej vety* práve 5 záznamov a tieto záznamy sa skladajú práve z množiny vzťahov {det, amod, nsubj, cop, root}, vyhľadanie pravidlá nám vráti minimálne tieto dve pravidlá. Po aplikovaní určenia najväčšej zhody týchto dvoch pravidiel a spracovávanou vetou, zistíme, že pravidlo prvej menovanej vety má so spracovávanou vetou cca. $99,99\%$ zhodu a pravidlo druhej vety má cca. $63,57\%$ zhodu. Aplikuje sa prvé pravidlo.

Ukážkový proces vyhľadania pravidla a určenie zhody je zobrazený na obrázku 16 Vyhľadanie pravidla.



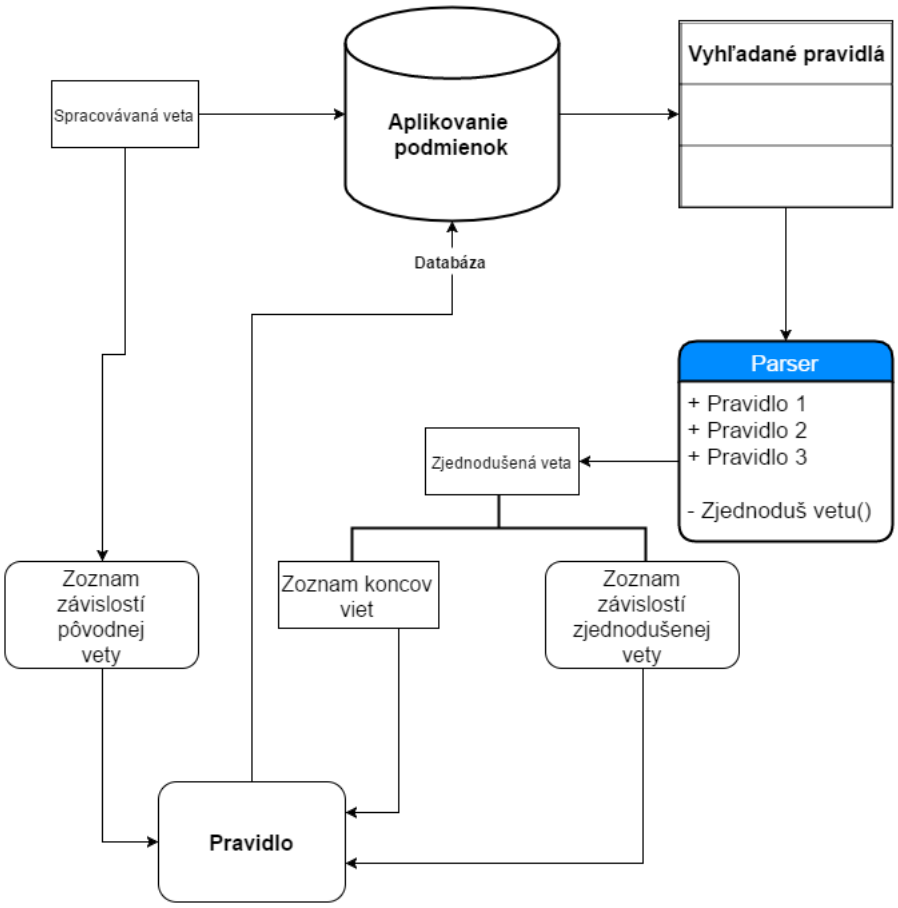
Obr. 16: *Vyhľadanie pravidla*

6.3.2 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 konce viet. Tieto informácie sa spolu uložia do dokumentu (záznamu) do databázy ako **pravidlo**.

Na obrázku 17 Vytvorenie pravidla je znázornený proces nevyhľadania pravidla, použitia parsera s následným uložením nového pravidla.



Obr. 17: *Vytvorenie pravidla*

6.3.3 Aplikovanie pravidla

Máme spracovávanú vetu a pravidlo na aplikovanie. Z princípu vyhľadávania pravidiel (viď. 6.3.1 Vyhľadávanie pravidla pre podrobnosti) vieme, že spracovávaná veta obsahuje závislosti zo zoznamu závislostí pôvodnej vety a tým pádom obsahuje aj závislosti zo zoznamu závislostí zjednodušenej vety.

Proces aplikovania pravidla prebieha nasledovne. Pre každú závislosť v zozname závislostí zjednodušenej vety, sa vyhľadá táto závislosť v spracovávanej vete. Z nájdenej závislosti sa zoberie slovo prislúchajúce podradenému tokenu a pridá sa do výslednej zjednodušenej vety na miesto svojho indexu. V prípade ak sa jedná o závislosť *nominal subject*, zoberie sa aj slovo prislúchajúce nadradenému

tokenu a taktiež sa pridá do výslednej vety. Po prejdení všetkých závislostí zo zoznamu sa urobia posledné úpravy zjednodušenej vety, ako kapitalizácia prvého písmena vety a rozdelenie vety na viacero viet, ak tak definovalo pravidlo.

Funkcia, ktorá bude aplikovať pravidlo na vetu s cieľom získania zjednodušenej vety bude vyzerať ako je naznačené v ukážke 3 Aplikovanie pravidla.

```
// Funkcia aplikuje pravidlo na vetu a vrati zjednodusenu vetu
private Note ApplyRule(NotenizerSentence sentence, NotenizerRule rule)
{
  // Vytvorenie objektu zjednodusenej vety
  Note note = new Note(sentence);
  // Vytvorenie objektu casti zjednodusenej vety - zjedn. veta sa moze skladat
  // z viacerych
  NotePart notePart = new NotePart(sentence);

  foreach (NotenizerDependency dependencyLoop in rule.RuleDependencies)
  {
    // Vyhlada zavislost a prida hodnotu prisluchajuceho slova do vyslednej
    // zjednodusenej vety
    ApplyRulesDependency(sentence, dependencyLoop, notePart);
  }

  // Finalne upravy
  note.Add(notePart);
  note.SplitToSentences(rule.SentencesEnds);

  return note;
}
```

Ukážka 3: *Aplikovanie pravidla*

Pre vetu „The president of the Czech Republic is Miloš Zeman.” nám nástroj Stanford CoreNLP poskytne závislosti slov vo vete, ktoré sú v textovej podobe výstupu zobrazené v ukážke 4 Závislosti jednoduchej vety. Závislosti sú v tvare

[názov závislosti] ([slovo prislúchajúce nadradenému tokenu] - [index slova vo vete], [slovo prislúchajúce podradenému tokenu] - [index slova vo vete]).

Ak na túto vetu aplikujeme pravidlo, ktoré obsahuje v zozname závislostí zjednodušenej vety dve závislosti:

1. nsubj

- podradený token
  - POS: NN
  - index: 2
- nadradený token
  - POS: NNP
  - index: 9

2. cop
   - podradený token
     - POS: VBZ
     - index: 7
   - nadradený token
     - POS: NNP
     - index: 9

Tak výsledná zjednodušená veta bude „President is Zeman.”.

```
root(ROOT-0, Zeman-9)
det(president-2, The-1)
nsubj(Zeman-9, president-2)
case(Republic-6, of-3)
det(Republic-6, the-4)
compund(Republic-6, Czech-5)
nmod:of(president-2, Republic-6)
cop(Zeman-9, is-7)
compund(Zeman-9, Milos-8)
```

Ukážka 4: *Závislosti jednoduchej vety*

Literatúra

[1] James F. Allen. Natural language processing. In *Encyclopedia of Computer Science*, pages 1218–1222. John Wiley and Sons Ltd., Chichester, UK.

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

[3] 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.

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

[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] 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.

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

[8] 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.

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

A Zoznam vzťahov závislostí

V nasledujúcej tabuľke sú zobrazené skratky vzťahov závislostí slov vo vete ako sa používajú v programe s celým názvom, vysvetlením, príkladom vety a použitím vzhľadom na príkladovú vetu.

Skratka	Názov	Vysvetlenie	Príklad	Použitie
nsubj	Nominal subject	Menná fráza, ktorá je syntaktickým subjektom klauzuly.	Clinton defeated Dole.	nsubj(Clinton, defeated)
nsubjpass	Nominal subject passive	Menná fráza v pasívnom tvare, ktorá je syntaktickým subjektom klauzuly.	Dole was defeated by Clinton.	nsubjpass(Dole, defeated)
dobj	Direct object	Sloveso v mennej fráze označujúce entitu, nad ktorou sa koná akcia.	She gave me a raise.	dobj(gave, raise)
nummod	Numeric modifier	Číselný modifikátor podstatného mena.	Sam spent forty dollars.	nummod(forty, dollars)
nmod	Nominal modifier	Podstatné meno alebo menná fráza služiaca ako doplnok.	The Chair's office.	nmod(Chair, office)
amod	Adjectival modifier	Prídavné meno ako modifikátor podstatného mena.	Sam eats red meat.	amod(red, meat)
neg	Negation modifier	Negácia.	Bill is not a scientist	neg(not, scientist)
compound	Compound	Zloženie slov, ktoré spolu majú význam.	I have four thousand sheep.	compound(four, thousand)
aux	Auxiliary	Vedľajšie sloveso klauzuly.	Regan has died.	aux(has, died)
cop	Copula	Vzťah medzi sponovým slovesom (to be) a jeho doplnkom.	Bill is honest.	cop(is, honest)
conj	Conjuct	Spojenie rovnocenných slov.	Bill is big and honest.	conj(big, honest)
cc	Coordinating conjuction	Vzťah medzi spojkou a slovom, patricim k nej.	Bill is big and honest.	cc(big, and)
dep	Unspecified dependency	Nešpecifikovaná závislost.		
root	Root	Koreň vety. V skutočnosti veta žiadne také slovo neobsahuje.	{ROOT} I love French fries.	root(ROOT, love)

Obr. 18: *Zoznam závislostí*

B Legenda diagramov kolekcií

V priloženej tabuľke je legenda pre diagramy zobrazujúce štruktúru dát ukladaných v kolekciách v databáze.

Pole	Popis
_id	Generovaná identifikačná hodnota záznamu.
text	Spracovaný text alebo článok.
texts	Odkazy do kolekcie *texts*.
originalSentence	Znenie pôvodnej vety pred spracovaním.
note	Znenie novej vety po spracovaní. Zjednodušená veta – poznámka.
sentences	Odkazy do kolekcie *sentences*.
sentenceEnds	Pozície koncov viet v spracovávanej vete alebo súvetí.
originalDependencies	Štruktúra závislosti pôvodnej vety.
noteDependencies	Štruktúra závislosti zjednodušenej vety – poznámky.
dependencyName	Názov závislosti.
dependencies	Zoznam závislosti.
governor	Nadradený token.
dependent	Podradený token.
position	Pozícia závislosti v zozname všetkých závislosti vety.
POS	Part-of-speech značka
index	Index slova vety prislúchajuceho tokenu.

Obr. 19: *Legenda diagramov kolekcií*

C Ukážka celého záznamu

V nasledujúcej ukážke je zobrazený celý záznam z databázy pre vetu „The president of the Czech Republic is Miloš Zeman.”.


```
{
  "_id" : ObjectId("562d5aa22a085409d0bdba1d"),
  "originalSentence" : "The president of the Czech Republic is Milos Zeman.",
  "note" : "President is Zeman.",
  "createdBy" : 0,
  "articleId" : -1,
  "sentencesEnds" : [3],
  "originalDependencies" : [{
    "dependencyName" : "det",
    "dependencies" : [{
      "governor" : {
        "pos" : "NN",
        "index" : 2
      },
      "dependent" : {
        "pos" : "DT",
        "index" : 1
      },
      "position" : 0
    }, {
      "governor" : {
        "pos" : "NNP",
        "index" : 6
      },
      "dependent" : {
        "pos" : "DT",
        "index" : 4
      },
      "position" : 3
    }]
  }, {
    "dependencyName" : "nsubj",
    "dependencies" : [{
      "governor" : {
        "pos" : "NNP",
        "index" : 9
      },
      "dependent" : {
        "pos" : "NN",
        "index" : 2
      },
      "position" : 1
```

```
    }]
  }, {
    "dependencyName" : "case",
    "dependencies" : [{
      "governor" : {
        "pos" : "NNP",
        "index" : 6
      },
      "dependent" : {
        "pos" : "IN",
        "index" : 3
      },
      "position" : 2
    }]
  }, {
    "dependencyName" : "compound",
    "dependencies" : [{
      "governor" : {
        "pos" : "NNP",
        "index" : 6
      },
      "dependent" : {
        "pos" : "NNP",
        "index" : 5
      },
      "position" : 4
    }, {
      "governor" : {
        "pos" : "NNP",
        "index" : 9
      },
      "dependent" : {
        "pos" : "NNP",
        "index" : 8
      },
      "position" : 7
    }]
  }, {
    "dependencyName" : "nmod",
    "dependencies" : [{
      "governor" : {
        "pos" : "NN",
        "index" : 2
      },
      "dependent" : {
        "pos" : "NNP",
        "index" : 6
      },
      "position" : 5
    }]
```

```
  }, {
    "dependencyName" : "cop",
    "dependencies" : [{
      "governor" : {
        "pos" : "NNP",
        "index" : 9
      },
      "dependent" : {
        "pos" : "VBZ",
        "index" : 7
      },
      "position" : 6
    }]
  }, {
    "dependencyName" : "root",
    "dependencies" : [{
      "governor" : {
        "pos" : "",
        "index" : -1
      },
      "dependent" : {
        "pos" : "NNP",
        "index" : 9
      },
      "position" : 8
    }]
  }],
  "noteDependencies" : [{
    "dependencyName" : "nsubj",
    "dependencies" : [{
      "governor" : {
        "pos" : "NNP",
        "index" : 9
      },
      "dependent" : {
        "pos" : "NN",
        "index" : 2
      },
      "position" : 0,
      "comparisonType" : 0
    }, {
      "governor" : {
        "pos" : "NNP",
        "index" : 9
      },
      "dependent" : {
        "pos" : "NN",
        "index" : 2
      },
      "position" : 2,
```

```
      "comparisonType" : 0
    }]
  }, {
    "dependencyName" : "cop",
    "dependencies" : [{
      "governor" : {
        "pos" : "NNP",
        "index" : 9
      },
      "dependent" : {
        "pos" : "VBZ",
        "index" : 7
      },
      "position" : 1
    }]
  }]
}
```


Ukážka 5: *Ukážka dát kolekcie rules*