Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5212-72264

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-5212-72264

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 sú podobné. 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 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 veľa času.

Učebné texty sú prevažne v neštruktúrovanej forme a v prirodzenom jazyku. Pre stroje je mnohokrát náročné správne interpretovať tieto informácie. Jedným z hlavných dôvodov je, 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 pracovať. 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 pomôcť 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 a najrelevantnejšej informácie z vety. Takto extrahované informácie následne ponúkneme používateľovi (študentovi).

2 Spracovanie prirodzeného jazyka

V tejto kapitole priblížime a rozoberieme spracovanie prirodzeného jazyka, jeho využitie v aplikáciach a systémoch a jeho hlavné úlohy. Ďalej analyzujeme nástroje, ktoré sa dajú využiť pri spracovávaní 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 viaceré aspekty. 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)
- Vyhľadávanie 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

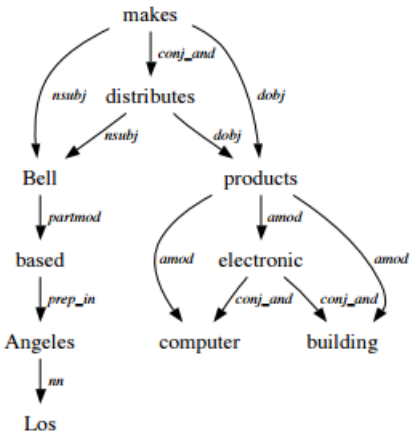
Nájdenie, identifikácia a rozpoznanie koreferencií v texte je úlohou rozpoznávania koreferencií (angl. Coreference resolution). V texte sa často používajú zámena (angl. pronouns) *to*, *tí*, *on*, anglicky *it*, *those*, *he* alebo menné frázy (angl. noun phrase). Tieto zámena a menné frázy sa odkazujú 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

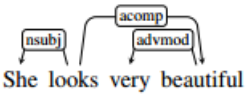
Rozloženie na vzťahy nám poskytuje jednoduchý opis gramatických vzťahov slov vo vete. Aplikovaním rozloženia vzťahov na vetu *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.* vznikne strom vzťahov (angl. dependency tree) (viď. obrázok 1 Strom vzťahov) [4].



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** - adjectival 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 viaceré nástroje, 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 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.

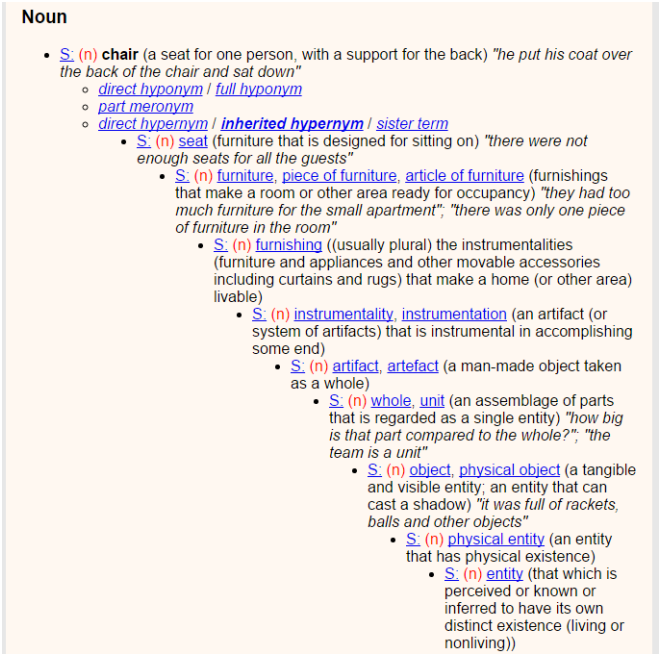
Slová do synetov sú zaraďované podľa významu. To znamená, že slová auto a automobil, ktoré sú pre svoj význam zameniteľné vo vete, sa zaraďujú do rovnakého synsetu. WordNet v súčasnosti (r. 2015) obsahuje 117 000 synsetov. Každý z týchto sysnsetov taktiež obsahuje krátku ukážku použitia slova.

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 (Wordnet)*



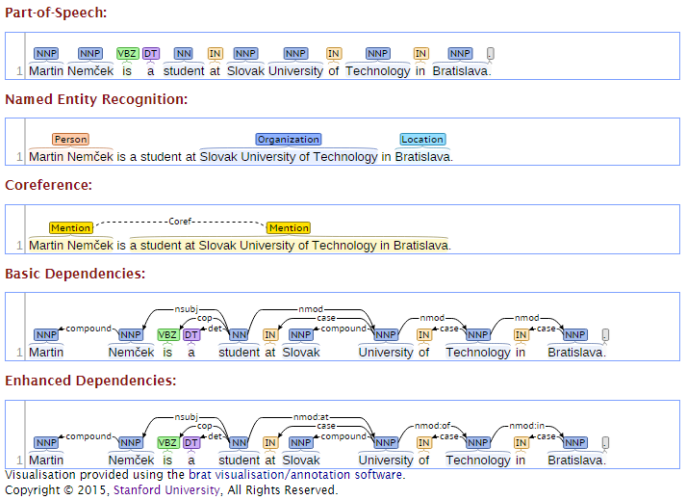
Obr. 4: *Nadradenost' slov (Wordnet)*

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.

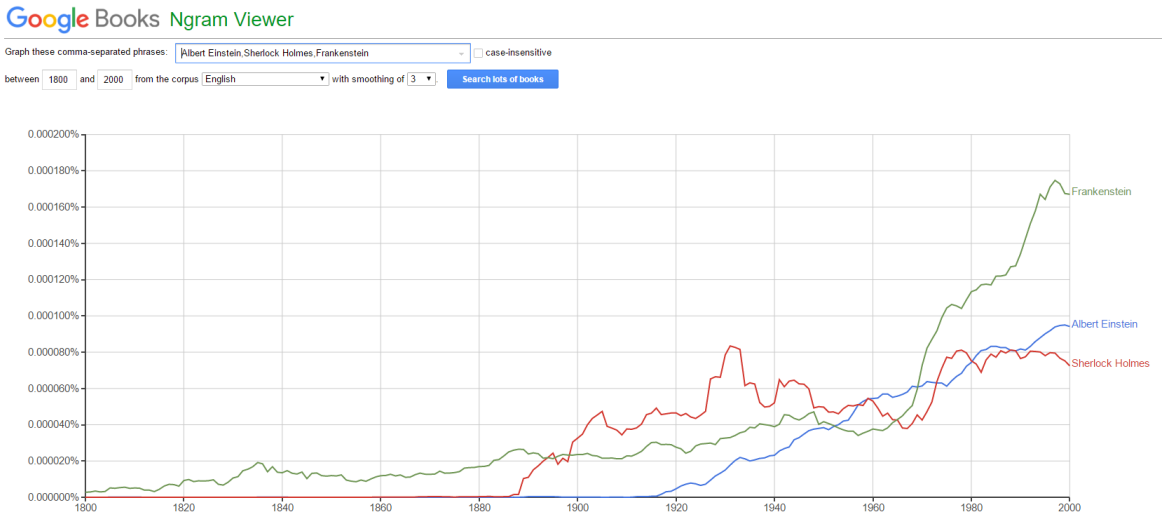
2.4.4 Google Ngram

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.

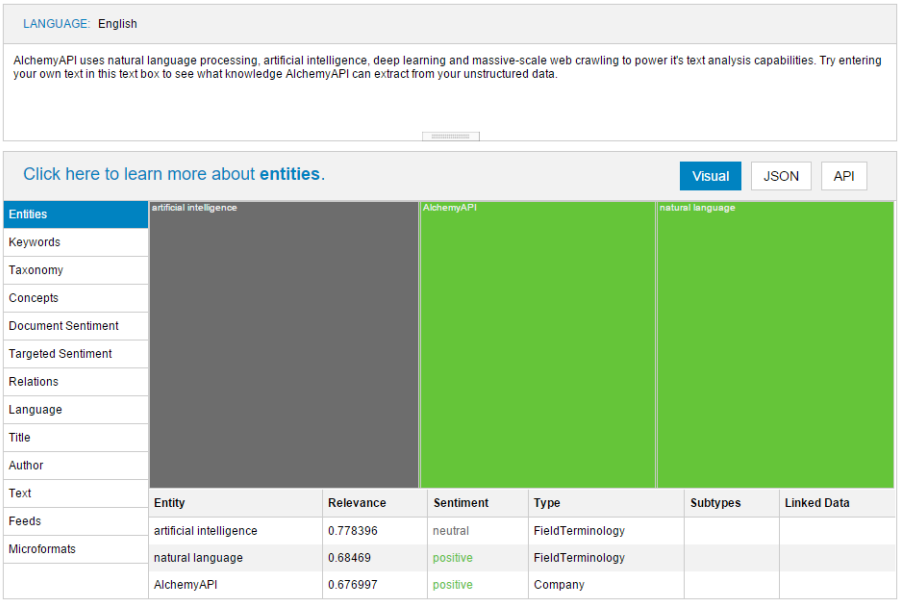
2.4.5 AlchemyAPI

AlchemiAPI[5] je súbor dvanástich nástrojov, 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.

2.5 Zhrnutie

Dummy text..

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

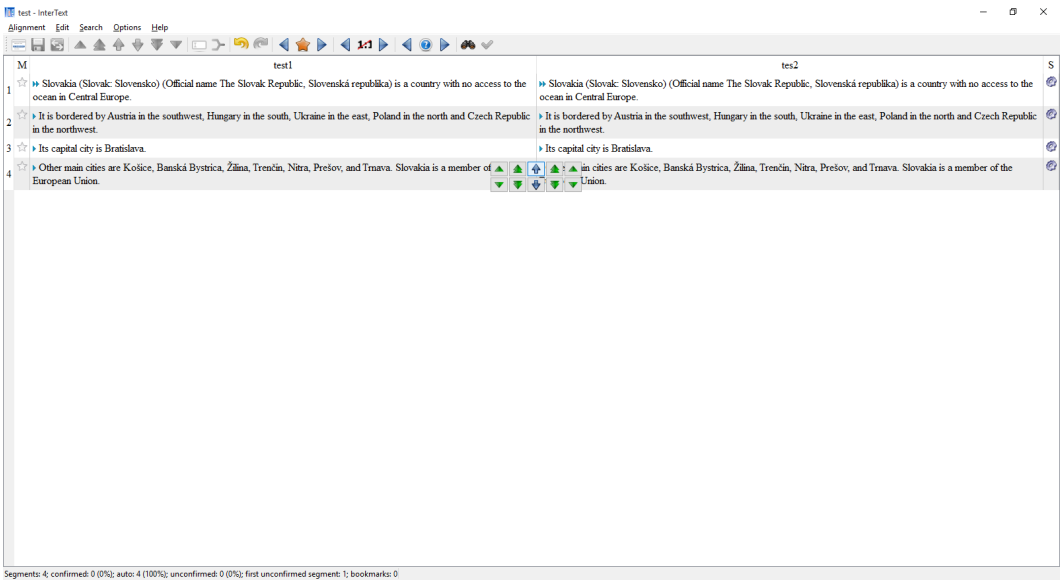
InterText[6] je editor paralelne zarovnaných textov, využívaný na správu viacerých paralelne zarovnaných verzií textu rôznych jazykov na úrovni viet. Táto aplikácia je dostupná vo verzií pre desktop, ale aj pre server.

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 používania 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*

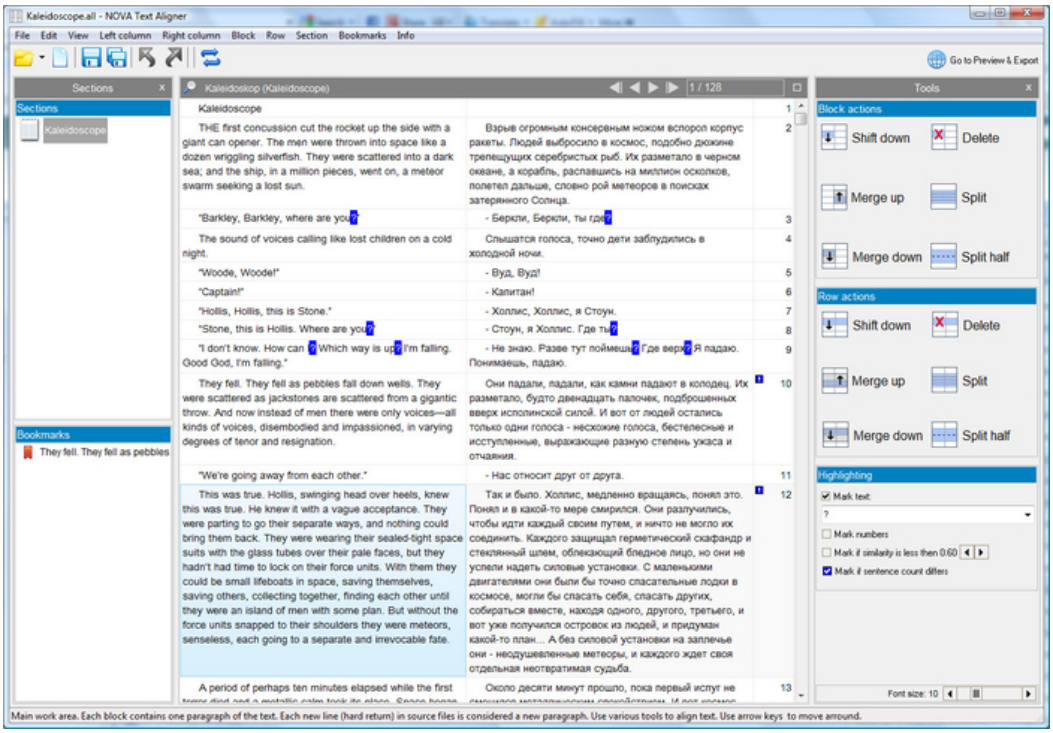
3.2 NOVA Text Aligner

NOVA Text Aligner[7] je aplikácia na zarovnávanie textu, pričom nevyužíva algoritmy na zarovnávanie textu, ale používateľ si musí sám určiť zarovnanie.

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/

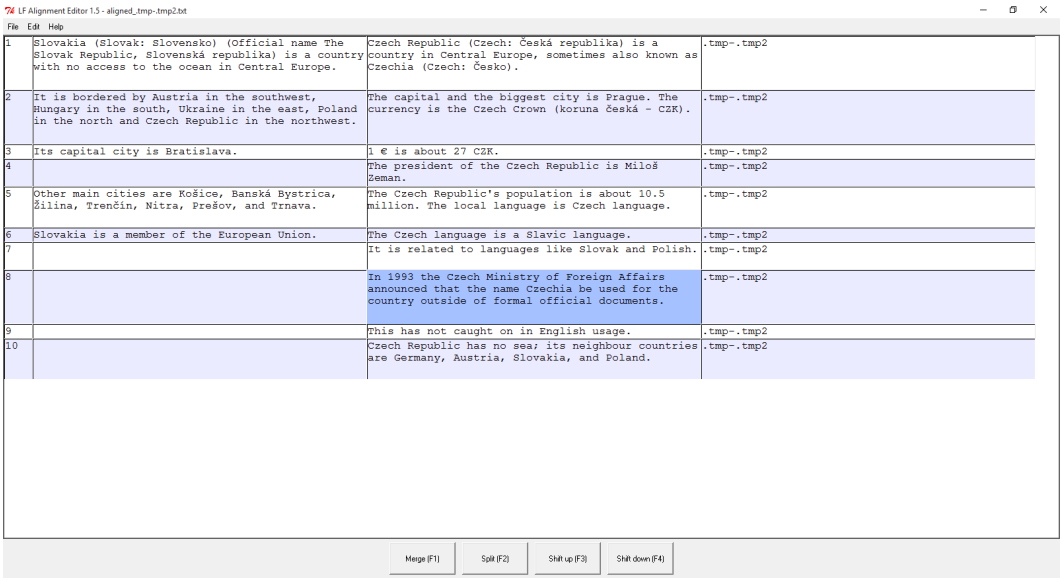


Obr. 9: *Aplikácia NOVA Text Aligner*[8]

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



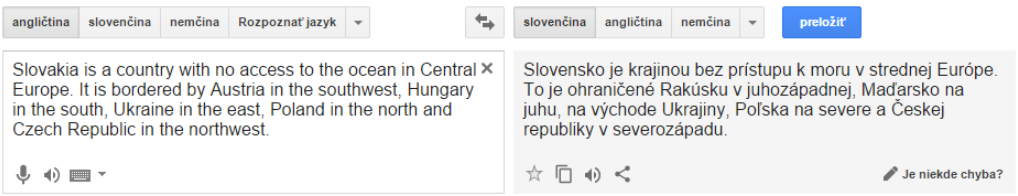
Obr. 10: *Aplikácia LF Aligner*

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*

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 musia byť uložené na externom úložisku, ako napríklad v databáze.

3.5 Uchovávanie textov v databázach

Text je špecifický údajový model s variabilnou štruktúrou. Ak chceme efektívne ukladať texty v databázach, je nutné použiť vhodnú databázu. 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.

3.5.1 Relačné databázy

Relačné databázy boli dlhé roky populárnou a finančne 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) a schéme sa stávajú 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].

3.5.2 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 formáty, 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 reťazec, 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 zmenou 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

[11] www.mongodb.org

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.

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

3.5.3 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].

3.6 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.

[TREBA TO TU NEJAK PREPOJIT, KEDZE TEXT HORE JE ZO ZHRNUTIA NASTROJOV A TEXT POD JE ZO ZHRNUTIA DATABAZ - SPOJIT DO JEDNEHO ZHRNUTIA CELEJ KAPITOLY !!!]

NOSQL databáza je narozdiel do RDBMS modelu (Relation Data Base Management System) navrhnutá, tak aby bola jednoducho rozšíriteľná so zväčšovaním dát. Väčšina NOSQL databáz odstránila niektoré nepotrebné prvky RDBMS modelov, čím sa stali podstatne ľahšími a efektívnejšími. 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.

V systéme 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 predvolené podporované vyhľadávania v štruktúrach ako text. MongoDB (viď. 3.5.2 MongoDB) je vyspelá, dokumentová databáza zahrňajúca všetku funkcionalitu, ktorú potrebujeme.

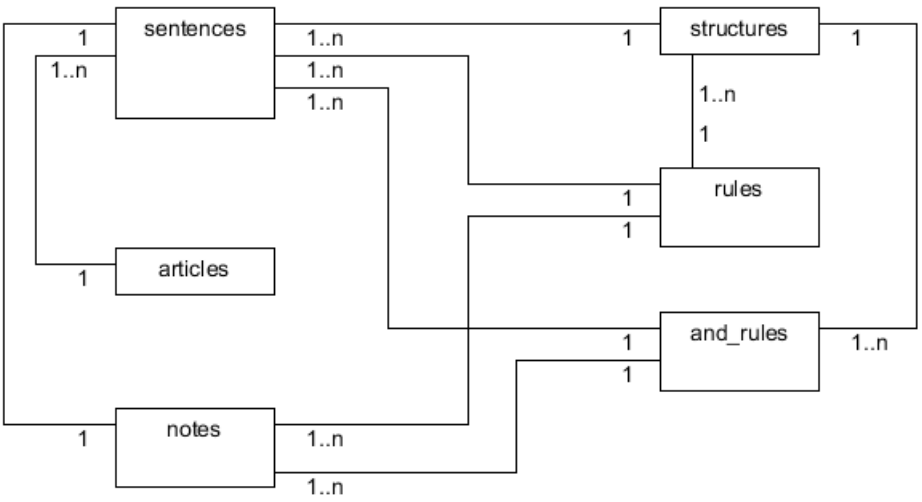
4 Návrh

4.1 Návrh uchovávania textov v databázach

Na uchovávanie dát sme zvolili dokumentovú databázu MongoDB. Ukladané dáta sa dajú rozdeliť do niekoľkých samostatných kolekcií. Sú to:

- rules,
- sentences,
- notes
- structures,
- articles,
- and rules.

Prepojenia medzi jednotlivými kolekciami sú zobrazené na obrázku 12 Databázový model. Nasledujúcich časti opisujú stručne každú kolekciu. Všetky kolekcie obsahujú okrem polí špecifických pre danú kolekciu, aj polia časových značiek označujúcich vytvorenie a aktualizáciu záznamu.



Obr. 12: *Databázový model*

4.1.1 Kolekcia articles

V kolekcií *articles* sa ukladajú spracovávané texty.

Kolekcia obsahuje textové pole *text* na uloženie textu v pôvodnom tvare. Model kolekcie *articles* je zobrazený na obrázku 13 Model kolekcie articles.

Pole	Typ
_id	ObjectId
text	String
created_at	DateTime
updated_at	DateTime

Obr. 13: *Model kolekcie articles*

4.1.2 Kolekcia notes

Kolekcia *notes* uchováva vytvorené poznámky z viet.

Obsahuje textové pole *text* s hodnotou poznámky a dve referencujúce polia. Jedno sa odkazuje do kolekcie *rules* na pravidlo, ktoré bolo použité na vytvorenie poznámky. Druhé referencuje použité and-pravidlo v kolekcií *and_rules*. Toto pole môže byť prázdne, ak sa and-pravidlo pri vytváraní poznámky nepoužilo. Na obrázku 14 Model kolekcie notes je vyobrazený model kolekcie *notes*.

Pole	Typ
_id	ObjectId
text	String
rule_ref_id	ObjectId
and_rule_ref_id	ObjectId
created_at	DateTime
updated_at	DateTime

Obr. 14: *Model kolekcie notes*

4.1.3 Kolekcia sentences

V následujúcej kolekcií *sentences* sa ukladajú spracované vety aj s informáciami o článku, pravidlách a poznámke.

Kolekcia sa skladá z textového polia *text* uchovávajúce hodnotu vety a piatich referencujúcich polí *article_ref_id*, *structure_ref_id*, *rule_id*, *rule_ref_id*, *and_rule_ref_id* a *note_id*. *Article_ref_id* odkazuje na článok z kolekcie *articles*, ktorého súčasťou je daná veta. Pole *structure_ref_id* odkazuje do kolekcie *structures*, ktoré reprezentuje štruktúru vety. Nasledujúce polia *rule_ref_id* a *and_rule_ref_id* odkazujú na použité pravidlo a and-pravidlo pri spracovávaní vety, v tomto poradí. Pole *note_ref_id* odkazuje na poznámku z kolekcie *notes*, ktorá bola vytvorená z vety.

Model tejto kolekcie je načrtnutý na obrázku 15 Model kolekcie sentences.

Pole	Typ
_id	ObjectId
text	String
article_ref_id	ObjectId
structure_ref_id	ObjectId
rule_ref_id	ObjectId
and_rule_ref_id	ObjectId
note_ref_id	ObjectId
created_at	DateTime
updated_at	DateTime

Obr. 15: *Model kolekcie sentences*

4.1.4 Kolekcia structures

V kolekcií *structures* sú uložené štruktúry viet a pravidiel. Štruktúra je zložená hlavne zo závislostí, tokenov, názvoslovných značiek, indexov a iné.

Kolekcia sa skladá z jedného pola *structure_data*. Toto pole je zoznam dokumentov, obsahujúcich vyššie spomenuté údaje. Dokument v tomto zozname

obsahuje textové pole *relation_name* s názvom vzťahu závislosti a zoznam dokumentov závislostí *dependencies* s týmto názvom vzťahu. Dokument v zozname *dependencies* sa skladá z polí *governor* a *dependent* typu dokument, celočíselného pola *position* uchovávajúceho pozíciu závislosti vo vete alebo poznámke, *comparison_type*, ktoré je celočíselnou reprezentáciou typu porovnania a poľa *token_type*, ktoré je taktiež celočíselnou reprezentáciou typu tokenu. Dokumenty polí *governor* a *depdendent* obsahujú údaje o prislúchajúcich tokenoch závislosti. V textovom poli *POS* sa ukladá značka slovného druhu, pole *index* uchováva index tokenu vo vete, *ner* je textové pole reprezentujúce názvoslovnú entitu tokenu a v poli *lemma* je textová reprezentácia lemy tokenu.

Celý model kolekcie je zobrazený na obrázku 16 Model kolekcie structures.

Pole	Typ
_id	ObjectId
structure_data	Array of Documents

Pole	Typ
relation_name	String
dependencies	Array of Documents

Pole	Typ
governor	Document

Pole	Typ
POS	String
index	Integer
ner	String
lemma	String

Pole	Typ
dependent	Document

Pole	Typ
POS	String
index	Integer
ner	String
lemma	String

Pole	Typ
position	Integer
comparison_type	Integer
token_type	Integer

Pole	Typ
created_at	DateTime
updated_at	DateTime

Obr. 16: *Model kolekcie structures*

4.1.5 Kolekcia rules

Rules je kolekcia, do ktorej sa ukladajú pravidla na vytvorenie poznámok z viet. Vďaka databázovému modelu na obrázku 12 Databázový model a prepojeniam medzi kolekciami je táto kolekcia minimalistická.

Skladá sa z dvoch polí. Pole *sentence_terminators* je zoznam čísel, ktoré reprezentujú konce viet v poznámke. Referencujúce pole *structure_ref_id* odkazuje do kolekcie *structures* na štruktúru, ktorou sa má prípadná veta spracovať. Model kolekcie *rules* je vyjadrený obrázkom 17 Model kolekcie rules.

Pole *sentence_terminators* zväčša obsahuje jeden záznam. Napríklad pri vete *„The president of the Slovak Republic is andrej Kiska."* a poznámke z tejto vety v tvare *„President is Kiska."* bude obsahovať jeden záznam: 3. Číslo 3 preto, lebo koniec vety, v tomto prípade bodka, sa nachádza na tretej pozícií spomedzi tokenov vo vete. Číslovanie pozícií začína indexom nula. V prípade ak veta je súvetie, zložené z viacerých jednoduchých viet, môže pole *sentence_terminators* obsahovať viacero záznamov, ak napríklad chceme z každej jednoduchej vety súvetia získať zjednodušenú vetu a vytvoriť tak zloženú poznámku, skladajúcu sa zo zjednodušených viet.

Pole	Typ
_id	ObjectId
sentence_terminators	Array of Integers
structure_ref_id	ObjectId
created_at	DateTime
updated_at	DateTime

Obr. 17: *Model kolekcie rules*

4.1.6 Kolekcia and rules

Posledná kolekcia uchováva pravidlá pre spracovanie vety a vytvorenie viacnásobnej poznámky z vety. Táto kolekcia je veľmi podobná kolekcií *rules* a obsahuje rovnaké polia doplnené o ďalšie špecifické pole.

Špecifické pole, o ktoré je kolekcia rozšírená oproti kolekcií *rules* je celočíselné pole *set_position*. Toto pole uchováva pozíciu množiny v viacnásobnej poznámke. Model kolekcie je vyobrazený na obrázku 18 Model kolekcie and rules.

Pole	Typ
_id	ObjectId
sentence_terminators	Array of Integers
set_position	Integer
structure_ref_id	ObjectId
created_at	DateTime
updated_at	DateTime

Obr. 18: *Model kolekcie and rules*

4.1.7 Zhrnutie

Pri návrhu databázového modelu a kolekcií sme vychádzali z princípu jednoduchých kolekcií so zoskupením súvisiacich dát a oddelenia ich od zvyšku. Vďaka využívaniu viacerých, medzi sebou prepojených, kolekcií sme zabezpečili neduplikovanie dát, jednoduché vyhľadanie napríklad viet ku článku a iné. Okrem toho nám tento model umožňuje ďalšiu funkcionalitu, ako napríklad aplikovanie jedného pravidla na viacero viet so zhodnou štruktúrou. Oddelenie dát do samostatných štruktúr nám uľahčuje aj prípadne neskoršie rozšírenie databázového modelu alebo zmenu konkrétnych štruktúr. Taktiež uľahčuje prípadné klastrovanie databázy, ak by bolo nutné, keďže každá kolekcia by mohla byť na samostatnom serveri.

4.2 Manažment dát

Pre vytvorenie poznámky z vety potrebujeme hlavne vetu a pravidlo, ktoré sa na vetu aplikuje. Aby sa vytvorila zodpovedajúca poznámka z vety, je nutné použiť vhodné pravidlo. Ak spracovávame doposiaľ nespracovanú vetu, je potrebné nájsť vhodné pravidlo na základe podobnosti spracovávanej vety so spracovanými

vetami. Pri extrakcií informácií z vety dané pravidlom, s cieľom vytvorenia poznámky sa musia vybrať správne informácie. Extrakcia musí fungovať pri udržaní dostatočnej všeobecnosti, aby sa dané pravidlo dalo aplikovať na viacero viet s rovnakou štruktúrou, ale odlišným obsahom.

4.2.1 Vyhľadanie pravidla

Pri spracovávaní vety, pred vytvorením poznámky, aplikovateľné pravidlo musí byť vyhľadané v databáze. Vyhľadá sa v databáze veta, ktorej štruktúra korešponduje so štruktúrou spracovávanej vety. Štruktúry oboch alebo viacerých viet musia obsahovať rovnakú množinu závislostí. To znamená rovnaký počet záznamov a záznamy s rovnakými názvami vzťahov závislostí v *zozname dát štruktúry*.

Podobná alebo zhodná veta je vyhľadaná, ak hlavná podmienka je splnená. Avšak, splnenie tejto podmienky nezaručí vyhľadanie len jednej, najpodobnejšej vety, ale môže vyhľadať viacero viet. V takom prípade sa vypočíta zhoda viet. Po určení zhody sa extrahuje pravidlo vety, s ktorou má spracovávaná veta najväčšiu zhodu.

Výpočet zhody viet

Zhoda viet pozostáva z troch častí:

- štrukturálna zhoda,
- obsahová zhoda,
- hodnotová zhoda.

Štrukturálna zhoda odzrkadľuje percentuálnu zhody dvoch štruktúr. Pri tejto zhode zisťujeme, či štruktúry obsahujú závislosti s nadradenými značkami slovných druhov a ich konkrétny počet. Štrukturálna zhoda znamená, že vety *„The president of the Slovak Republic is Andrej Kiska."* a *„Andrej Kiska is the president of the Slovak Republic."*, ale aj *„Miloš Zeman is the president of the Czech Republic."* majú rovnakú štruktúru, bez ohľadu na hodnoty a pozície slov vo vete, pokiaľ obsahujú také iste závislosti s rovnakými nadradenými značkami slovných druhov. Pod nadradenou značkou slovného druhu sa chápe, že napríklad značka slovného druhu VBD (Verb, past tense - sloveso v minulom čase) patrí

medzi skupinu značiek slovných druhov slovies. Z toho vyplýva, že nadradená značka slovného druhu *VBD* je VB (Verb - sloveso). Určenie štrukturálnej zhody pozostáva z niekoľkých krokov. Najskôr sa separátne počíta zhoda nadradených značiek slovných druhov nadradeného a podradeného tokenu. Týmito krokmi sa určí, či veta obsahuje ľubovolnú závislosť s rovnakou nadradenou značkou slovného druhu na ľubovolnom tokene. V nasledujúcom kroku sa určí úplná zhoda závislosti s nadradenými značkami slovných druhov, to znamená zistenie, či veta obsahuje ľubovoľnú závislosť s konkrétnymi značkami slovných druhov na nadradenom a zároveň podradenom tokene. V poslednom kroku sa zisťuje zhoda počtu závislostí s rovnakými nadradenými značkami slovných druhov u nadradeného a podradeného tokenu.

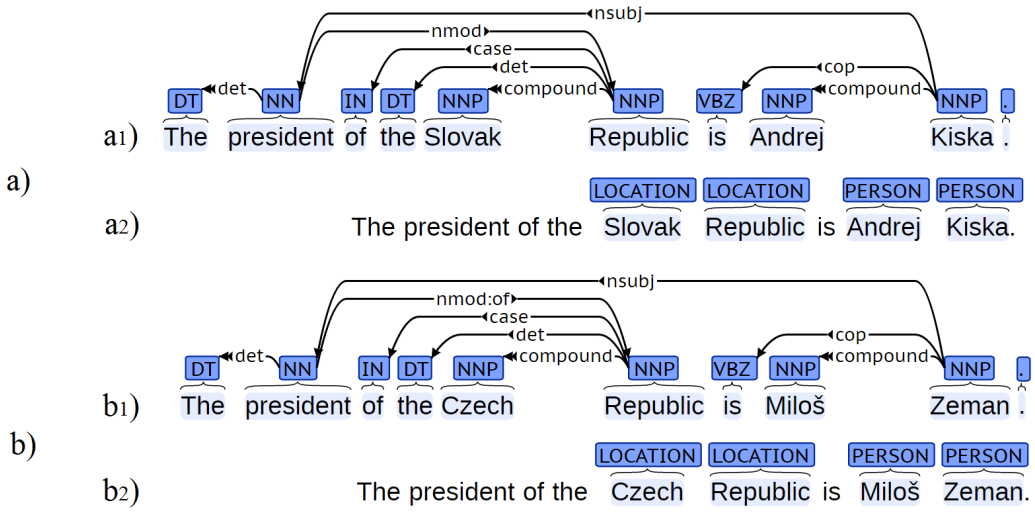
Obsahová zhoda zodpovedá percentuálnej zhode obsahu dvoch viet. Kontrolujú sa indexy slov, konkrétne značky slovných druhov a názvoslovné entity. Pri obsahovej zhode majú vety *„The president of the Slovak Republic is Andrej Kiska."* a *„The president of the Czech Republic is Miloš Zeman."* obsahovú zhodu, bez ohľadu na konkrétne slova na pozíciach vo vete, ak obsahujú rovnaké značky slovných druhov a reprezentujú ich zhodné názvoslovné entity. Výpočet obsahovej zhody sa skladá z viacerých krokov. Začína sa výpočtom zhôd značiek slovných druhov podradeného a nadradeného tokenu. Zhody indexov nadradeného a podradeného tokenu su taktiež vypočítané separátne. Tak isto sa vypočítajú zhody aj názvoslovných entít. Tieto prvé kroky určia, či veta obsahuje ľubovolnú závislosť s rovnakou značkou slovného druho alebo indexom a ľubovoľnú závislosť s rovnakou názvoslovnou entitou. V nasledujúcom kroku je určená polovičná zhoda závislostí. Polovičná zhoda závislosti je zhoda značky slovného druhu a indexu nadradeného alebo podradeného tokenu. Polovičná zhoda sa vypočíta rovnako aj pre názvoslovné entity. Nakoniec, v poslednom kroku, počítame počet úplne zhodných závislostí. Úplná zhoda závislosti je zhoda značky slovného druhu a indexu nadradeného, a zároveň podradeného tokenu. Tak isto sa vypočíta úplná zhoda závislostí aj pre názvoslovné entity.

Posledná časť zhody, hodnotová zhoda, reprezentuje úplnú zhodu dvoch viet. Veta má hodnotovú zhodu len s totožnou vetou.

Každý krok, pri výpočte všetkých časti zhody, má priradené ohodnotenie. Ak je podmienka v kroku vyhodnotená ako správna, ohodnotenie kroku je pripočítané do finálnej hodnoty. Finálna zhoda je percentuálne ohodnotenie zhody. Ohodnotenie krokov odzrkadľuje dôležitosť daného kroku vo výpočte presnej zhody, pričom závisí od počtu závislostí a krokov, tak že finálna zhoda nemôže presiahnuť hodnotu 100%. Pseudokód 1 zobrazuje algoritmus výpočtu zhody, konkrétny príklad je zobrazený na obrázku 19 Príklad určenia zhody viet

Algoritmus 1 Výpočet zhody

```
1: procedure VYPOCETZHODY(spracovávanáVeta, zavislostiPorovnávanejVety)
2:     vypočítaj ohodnotenia krokov
3:     for all závislosti porovnávanej vety do
4:         for all závislosti spracovávanej vety do
5:             for all porovnania do
6:                 if aplikujPorovnanie(spracovávanáVeta, porovnanie, závislosť) then
7:                     do zhody na type porovnania pripočítaj ohodnotenie kroku
       return zhoda
```



Obr. 19: *Príklad určenia zhody viet*

Predpokladajme situáciu z obrázka 19 Príklad určenia zhody viet. V databáze máme uloženú spracovanú vetu *a* aj s pravidlom pre túto vetu. Spracovávame vetu *b*. Na časti *a1* obrázka 19 Príklad určenia zhody viet sú znázornené závislostí a na časti *a2* názvoslovné entity vety *a*. Na časti *b1* sú znázornené závislostí a na časti *b2* názvoslovné entity vety *b*. V tejto situácií potrebujeme vypočítať zhodu medzi vetami *a* a *b*.

Pri štrukturálnej časti zhody sa prechádza cez všetky závislosti vety *a*. Prvá závislosť je závislosť so vzťahom *DET* a nadradeným tokenom s nadradenou značkou slovného druhu *NN* a podradeným tokenom s nadradenou značkou slovného druhu *DT*. V prom kroku sa pozrie, či veta *b* obsahuje ľubovoľnú závislosť s podradeným tokenom s nadradenou značkou slovného druhu *DT*. V ďalšom kroku, sa zistí, či veta *b* obsahuje ľubovoľnú závislosť s nadradeným tokenom s nadradenou značkou sloveného druhu *NN*. Môže to byť aj iná závislosť, ako tá z prvého kroku. Pokračuje sa zistením úplnej zhody závislosti. Určí sa , či veta **b** obsahuje ľubovoľnú závislosť s podradeným tokenom s nadradenou značkou slovného druhu práve *DT* a zároveň s nadradeným tokenom s nadradenou značkou slovného druhu *NN*. Takto sa iteruje cez všetky závislosti vety *a*. Na záver sa určí zhoda počtu závislostí s rovnakou štruktúrou. Napríklad veta *a* obsahuje práve dve závislosti, ktoré majú na podradenom tokene nadradenú značku slového druhu *DT* a na nadradenom tokene nadradenú značku slovného druhu *NN*. Zistí sa, či aj veta *b* obsahuje práve dve takéto závislosti.

Prvá závislosť vo vete *b* je so vzťahom *det*, nadradeným tokenom so značkou slovného druhu *NN* a indexom 1 a podradeným tokenom so značkou slovného druhu *DT* a indexom 0. Token slova *Slovak* má názvoslovnú entity typu *LOCATION - lokácia*. Ak slovo nemá vyobrazený typ názvoslovnej entity, znamená to, že má názvoslovnú entity typu *OTHER - ostatné*. V prvok kroku pri určovaní obsahovej časti zhody zisťujeme, či veta *b* obsahuje závislosť so vzťahom *det* a tokenmi so značkou slovného druhu *NN* alebo *DT* a indexmi rovnými 0 alebo 1. Toto je separátny výpočet značiek slovných druhov a indexov. V tomto isto kroku sa tiež pozrie, či veta *b* obsahuje ľubovoľnú závislosť s podradeným tokenon s názvoslovnou entitou typu *ostatné* (názvoslovná entita tokenu *THE* vo vete *a*) a ľubovoľnú závislosť s nadradeným tokenom s názvoslovnou entitou typu *ostatné*

(názvoslovná entita tokenu *president* vo vete *a*). V nasledujúcom kroku zisťujeme, či veta *b* obsahuje závislosť so vzťahom *det* a nadradeným alebo podradeným tokenom so značkou slovného druhu *NN* a indexom 1 alebo značkou slovného druhu *DT* a indexom 0. Toto je polovičná zhoda. V poslednom kroku hľadáme vo vete *b* závislosť so vzťahom *det* a nadradeným tokenom práve so značkou slovného druhu *NN* a indexom 1 a zároveň podradeným tokenom práve so značkou slovného druhu *DT* a indexom 0. Zároveň v tomto kroku sa zisťuje, či veta *b* obsahuje závislosť s podradeným tokenom s názvoslovnou entitou typu *ostatné* a zároveň s nadradeným tokenom s názvoslovnou entitou typy *ostatné*. Iterácia pokračuje s nasledujúcou závislosťou, pokým sa nevyhodnotia všetky.

Pri určení hodnotovej časti zhody sa porovnajú texty *„The president of the Slovak Republic is Andrej Kiska."* a *„The president of the Czech Republic is Miloš Zeman."* a určí sa, či su zhodné.

Aplikovaním určenia zhody viet medzi vetami *a* a *b* zistíme, že veta *b* má štrukturálnu časť zhody s vetou *a* 100%. Tak isto má obsahovú časť zhody rovnú 100%. Hodnotová časť zhody je 0%.

4.2.2 Aplikovanie pravidla

Z princípu vyhľadania pravidla (viď. 4.2.1 Vyhľadanie pravidla), spracovávaná veta musí obsahovať, nie všetky, závislostí zo *zoznamu dát pôvodnej vety*, vety prepojenej s pravidlom a tým pádom aj závislostí zo *zoznamu dát poznámky* pravidla.

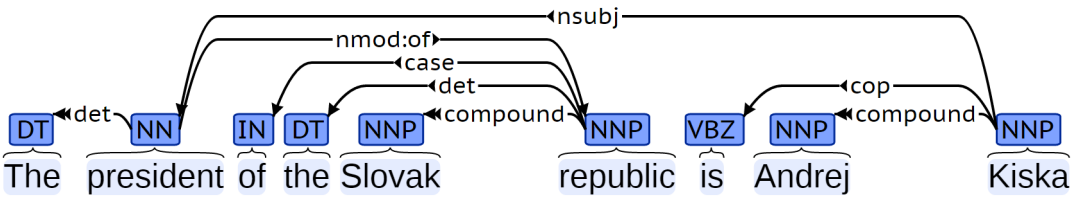
Proces aplikovania pravidla na vetu s cieľom vytvorenia poznámky sa skladá z niekoľkých krokov. Pre všetky závislosti zo *zoznamu dát poznámky*, príslušná závislosť je vyhľadaná v spracovávanej vete. Pri vyhľadávaní príslušnej závislosti sa závislosti neporovnávajú, okrem iného, na základe POS značiek svojich tokenov, ale podľa nadradených POS značiek svojich tokenov. To znamená, že ak token obsahuje POS značku NNP (proper noun, singular - [SLOVENSKY EKVIVALENT]), jeho nadradaná POS značka je NN (noun - podstatné meno). Vyhľadáva sa teda podľa množiny POS značiek podstatného mena, a to konkrétne

NN, NNS, NNP a *NNPS*. Tento spôsob vyhľadávania nám umožňuje [BLA BLA, blizie opisane v BLA BLA]. Avšak, môže to spôsobiť vyhľadanie viac ako jednej príslušnej závislosti. Preto zhoda závislostí musí byť vypočítaná (viď. Výpočet zhody závislostí na strane 37). Po vypočítaní zhody závislosti a získanie závislosti s najväčšou zhodou, slovo korešpondujúce s tokenom, ktorý sa ma z danej závislosti vybrať, sa pridá do poznámky na pozíciu indexu tokenu. Po spracovaní všetkých závislostí, posledné minoritné úpravy su vykonané nad poznámkou, ako napríklad rozdelenie na viacero viet, ak tak určovalo pravidlo, kapitalizácia prvých písmen viet poznámky a iné. Pseudokód aplikovania pravidla na vetu s cieľom vytvoriť poznámku je zobrazený na algoritme 2.

Algoritmus 2 Aplikovanie pravidla

```
1: procedure APPLYRULE(sentence, rule)
2:     note ← new Note
3:     for all ruleDependencies do
4:         dependency ← findDependency(sentence, ruleDependency)
5:         if isFound(dependency) then
6:             add(note, getDependent(dependency))
7:             if isNominalSubject(relation(dependency)) then
8:                 add(note, getGovernor(dependency))
9:     splitToSentences(note, sentencesEnds(rule))
       return note
```

Pre vetu „The president of the Slovak republic is Andrej Kiska.” nám nástroj Stanford CoreNLP poskytne závislostí vyobrazené na obrázku 20 Zásivlostí jednoduchej vety. Ak na túto vetu aplikujeme pravidlo v tvare zobrazené na obrázku 21 Príklad pravidla, výsledná poznámka bude „President is Kiska.”.



Obr. 20: *Zásivlostí jednoduchej vety*

Konce viet	Závislostí								
3	Vzťah	Pozícia	Porovnanie	Typ tokenu	Tokeny				
					Typ	POS	NER	Lemma	Index
	nsubj	1	-	podradený	nadradený	NNP	PERSON	Kiska	9
					podradený	NN	OTHER	President	2
	nsubj	8	-	nadradený	nadradený	NNP	PERSON	Kiska	9
					podradený	NN	OTHER	President	2
	cop	6	-	podradený	nadradený	NNP	PERSON	Kiska	9
					podradený	VBZ	OTHER	be	7

Obr. 21: *Príklad pravidla*

Výpočet zhody závislostí

Výpočet zhody závislostí pozostáva s niekoľkých krokov a princíp výpočtu je veľmi podobný s výpočtom zhody viet zo sekcie 4.2.1 Vyhľadanie pravidla. Porovnávajú sa vždy nadradené aj podradené tokeny. Porovnanie ma niekoľko krokov. Začína sa s porovnaním POS značiek. Pokračuje sa názvoslovnou entitou, indexom, lemou a nakoniec sa porovná vzdialenosť pozícií tokenov vo vetách. Každý krok je príslušne ohodnotený a ak porovnanie bolo úspešné, ohodnotenie sa pripočíta k finálnej hodnote reprezentujúca percentuálne zhodu závislostí.

4.2.3 Vytvorenie pravidla

Ak nám proces vyhľadania pravidla nevyhľadal žiadne pravidlo, znamená to, že sme doposiaľ nespracovávali takú istú alebo podobnú vetu. V tomto prípade sú použité statické pravidlá na spracovanie vety. Výstupom bude zjednodušená veta - poznámka.

Vytvorí sa záznam o pôvodnej vete, ktorý okrem iného obsahuje *zoznam dát o pôvodnej vete*, ktorý sa vyskladá zo závislostí vety. Následne sa vytvorí záznam o pravidle, ktorý okrem iného obsahuje *zoznam dát o poznámke*, vyskladaný zo závislosti poznámky. Nakoniec sa tieto dva záznamy prepoja a tým sa vytvorí nové pravidlo na spracovanie takej istej alebo podobnej vety, akú sme práve spracovali.

Literatúra

[1] James F. Allen. Natural language processing. In *Encyclopedia of Computer Science*, pages 1218–1222. John Wiley and Sons Ltd., Chichester, UK, 2003.

[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. 22: *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	Značka slovného druhu
index	Index slova vety prislúchajuceho tokenu.

Obr. 23: *Legenda diagramov kolekcií*