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

Máj 2016

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: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU, Bratislava
Vedúci práce: Ing. Miroslav Blšták

Máj 2016

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika
Autor: Martin Nemček
Bakalárska práca: Spracovanie učebných textov
Vedúci práce: Ing. Miroslav Blšták
Máj 2016

Sme zavalení množstvom informácií z rôznych zdrojov. Vo výučbe je náročné vytvoriť poznámky, ktoré zahŕňajú podmnožinu dôležitých informácií zo zdroja. Existuje niekoľko spôsobov, ako extrahovať informácie z textu. V tejto práci navrhujeme systém na extrakciu poznámok z textu, ktoré sú dôležité z pohľadu výučby, a bude umožní študentom vytvárať si personalizované poznámky. Využívame hlavne syntaktickú analýzu textu. Poznámky sa vytvárajú pomocou využitie značiek slovných druhov a závislostí medzi slovami vo vetách. Výsledkom je interaktívny systém na tvorbu poznámok, na základe pravidiel naučených od používateľa.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Informatika
Author: Martin Nemček
Bachelor thesis: Spracovanie učebných textov
Supervisor: Ing. Miroslav Blšták
Máj 2016

We are overwhelmed by information from various topics. The challenge in education is to create notes which covers important subset of information. There are known methods to extract information from text. In this thesis we propose a system to extract the notes from text which are important for educational purpose, so it should create personalized notes for students. We use mainly syntactic text analysis. Notes are created by help of part-of-speech tags and dependencies between words in sentences. The outcome is an interactive system for creating notes based on learned rules from user.

ACKNOWLEDGMENTS

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

DECLARATION

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

.....

Martin Nemček

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam ukážok

1 Úvod

V dnešnej dobe sme obklopení veľkým množstvom dát z rôznych zdrojov, či už internetu, kníh alebo iných. Vo vzdelávaní je problém vytvoriť poznámky zo zdroja, ktoré by zahrňovali podmnožinu dôležitých informácií.

Väčšina učebných zdrojov, textov je písaná v prirodzenom jazyku a má neštruktúrovanú formu. Stroje zatiaľ nedokážu úplne pochopiť prirodzený jazyk z dôvodu komplexnosti a variácie jednotlivých jazykov. Spracovanie prirodzeného jazyka je obor, ktorý sa zaoberá spracovaním prirodzeného jazyka strojmi. V našom systéme využívame niekoľko úloh tohto oboru.

Systém sa zameriava na tvorbu poznámok z viet pomocou extrakcie relevantných informácií z nich. Na tvorbu poznámok sa používa hlavne syntaktická analýza viet a extrakcia vzťahov a závislostí medzi slovami viet. Výstupom nami navrhnutého systému sú personalizované poznámky. Systém implementuje prvky interaktivity, ktoré umožňujú používateľovi modifikáciu automaticky vytvorených poznámok. Na základe zmien sa systém „naučí" nove pravidlá ako spracovávať vety. Tieto nové pravidlá zohľadní pri nasledujúcom spracovávaní zdrojov.

1.1 Motivácia

Študenti musia často spracovávať veľké množstvá dát, pričom dôležitá je pre nich iba časť dát, ktorá obsahuje relevantné informácie. Proces selekcie relevantných informácií im zaberá pomerne dosť času. S pomocou nami navrhnutého systému by si vedeli z dát vytvoriť poznámky a tak získať podmnožinu, pre nich relevantných informácií, rýchlejšie. Vytvorené poznámku by boli personalizované, teda priamo prispôsobené tvorbe poznámok používateľa a tým väčšmi zredukuje potrebný čas, keďže používateľ nebude musieť získané relevantné informácie ešte následne upravovať. Získaný čas by mohli využiť efektívnejšie.

2 Spracovanie prirodzeného jazyka

V kapitole 2.1 Spracovanie prirodzeného jazyka približujeme a rozoberáme spracovanie prirodzeného jazyka, jeho využitie v aplikáciach a systémoch a jeho hlavné úlohy. Ďalej v kapitole 2.3 Nástroje na spracovanie prirodzeného jazyka 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 spracovania prirodzeného jazyka 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 spracovanie prirodzeného jazyka využívajú strojové učenie.

2.2 Úlohy spracovania prirodzeného jazyka

Spracovanie prirodzeného jazyka má niekoľko hlavných úloh. V nasledujúcich častiach sú podrobnejšie opísané tie, ktoré sú relevantné vzhľadom na spracovanie učebných textov. Úlohy spracovania prirodzeného jazyka: [5]

– Značkovanie slovných druhov (angl. Part-of-speech tagging) 2.2.1,

– Rozdelenie vety na menšie časti (angl. Chunking),
– Rozpoznávanie názvoslovných entít (angl. Named Entity Recognition) 2.2.2,
– Označovanie sémantického postavenie (angl. Semantic Role Labeling),
– Rozpoznanie koreferencií (angl. Coreference resolution) 2.2.3,
– Morfologické segmentovanie (angl. Morphological Segmentation),
– Generovanie prirodzeného jazyka (angl. Natural Language Generation),
– Optické rozoznávanie textu (angl. Optical Character Recognition),
– Rozloženie vzťahov (angl. Dependency parsing) 2.2.4,
– a ďalšie.

Hlavné úlohy spracovania prirodzeného jazyka sú implementované a využívané vo viacerých smeroch. Z hľadiska spracovania učebných textov je pre nás najdôležitejšie využitie extrakcie informácií, ktoré je podrobnejšie popísané v sekcii 2.2.5 Extrakcia informácií. Ďalšie využitia spracovania prirodzeného jazyka 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),
– a ďalšie.

2.2.1 Značkovanie slovných druhov

Hlavnou úlohou značkovania slovných druhov (angl. Part-of-speech tagging) je každému slovu vo vete priradiť unikátnu značku označujúcu 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 vymenovaných označení to môže byť aj 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 napríklad slovo *book*, ktoré môže predstavovať podstatné meno *kniha* alebo sloveso vo význame *rezervovať*.

2.2.2 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, z dôvodu písania entít s veľkým začiatočným písmenom.

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

2.2.3 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 spracovania prirodzeného jazyka, ako sú extrakcia informácií (viď. 2.2.5 Extrakcia informácií) a odpovedanie na otázky [3].

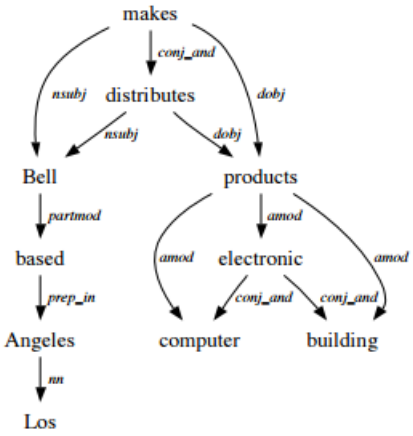
Príklad: **Martin Nemček** napísal túto bakalársku prácu. **On** študuje na FIIT STU BA.

Zámeno *on* sa odkazuje na meno *Martin Nemček*.

2.2.4 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,*

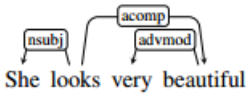
makes and distributes electronic, computer and building products. vznikne strom vzťahov (angl. dependency tree) (viď. obrázok 1 Strom vzťahov) [4].



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*.

2.2.5 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 2.2 Úlohy spracovania prirodzeného jazyka.

Výber informácií a extrakcia informácií spolu úzko súvisia, ale sú to dve rozdielne využitia spracovania prirodzeného jazyka. 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 Nástroje na spracovanie prirodzeného jazyka

V súčastnosti 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 aj mimo univerzít napríklad v Google.

2.3.1 WordNet

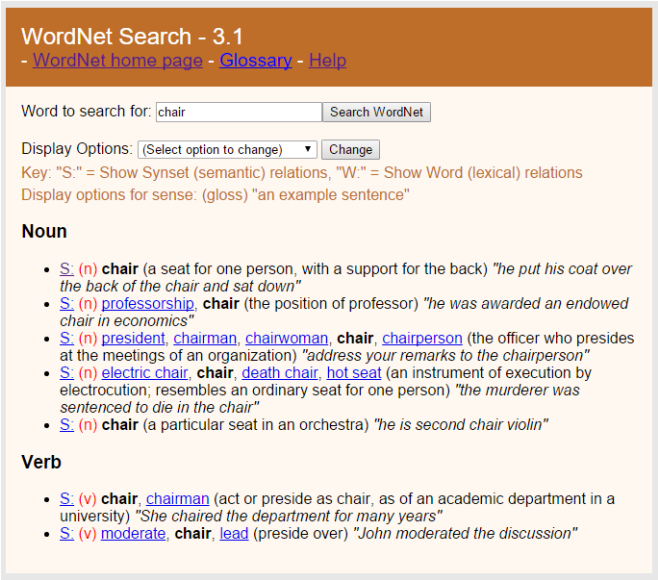
WordNet[1] je databáza anglických slov vyvíjaná na Princetonskej univerzite. Databáza obsahuje podstatné mena, prídavné mená, slovesá a príslovky, ktoré sú zatriedené do synonymických sád, synsetov.

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 nich obsahuje aj krátku ukážku použitia slova.

[1] www.wordnet.princeton.edu

Vo WordNet sa nachádzajú aj vzťahy medzi slovami v zmysle nadradenosti. To znamená, 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.



Obr. 3: *Webové rozhranie (Wordnet)*

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

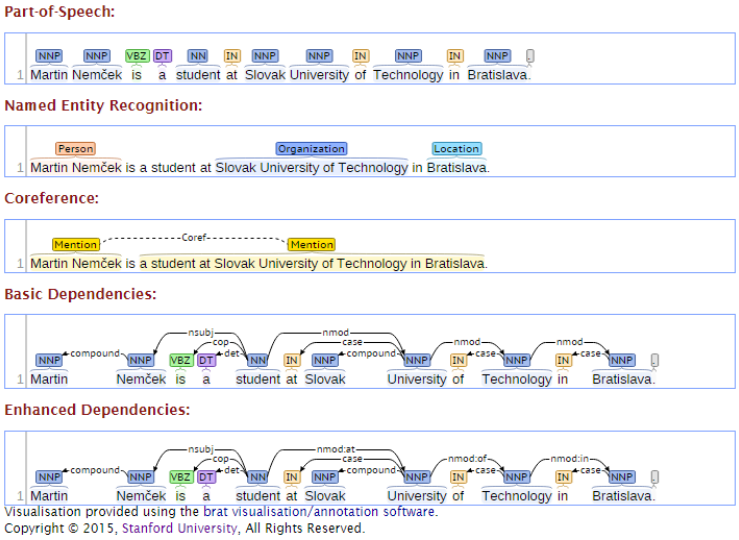
2.3.2 StanfordNLP

Nástroj StanfordNLP[2] je vyvíjaný na Stanfordskej univerzite. Skladá sa z niekoľkých softvérov, ktoré sa zameriavajú na úlohy spracovania prirodzeného jazyka popísané v sekcií 2.2 Úlohy spracovania prirodzeného jazyka. Sú to softvéry *Stanford Parser*, *Stanford POS Tagger*, *Stanford EnglishTokenizer*, *Stanford Relation Extractor* a mnoho ďalších. *Stanford CoreNLP* zahŕňa viacero zo spomenutých softvérov.

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

Dostupné je aj online webové demo. Na obrázku 5 StanfordNLP online demo vidno 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.".

[2] www.nlp.stanford.edu



Obr. 5: *StanfordNLP online demo*

2.3.3 CambridgeAPI

CambridgeAPI[3] je nástroj vytvorený na Cambridge univerzite. Umožňuje prístup k viacerým slovníkom. Momentálne (r. 2015) tento nástroj ponúka prístup k pätnástim prekladovým slovníkom, ako napríklad anglicko-čínsky, anglicko-ruský, anglicko-arabský, anglicko-japonský a ďalšie. Všetky prekladové slovníky majú primárny jazyk angličtinu. Slovenčinu v súčastnosti nepodporuje.

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

2.3.4 Google Ngram

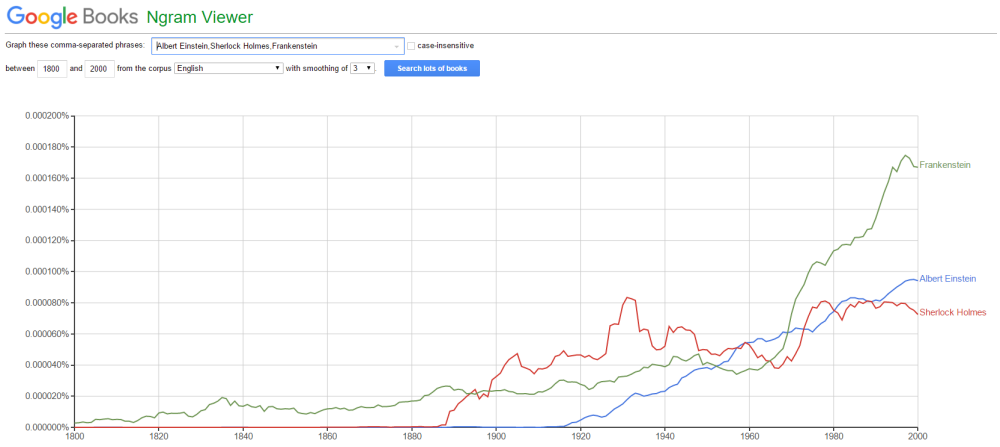
Google Ngram[4] je postavený na ďalšom softvéri od spoločnosti Google, Google Books. V knihách, napísaných od roku 1500 až do súčastnosti, vyhľadáva výskyty n-gramov. Podporuje len niektoré jazyky, ako angličtina, francúzština, ruština alebo čínština. Na vyhľadávanie v knihách využíva optické rozpoznávanie textu,

[3] www.dictionary-api.cambridge.org
[4] www.books.google.com/ngrams

pričom dokáže spracovať aj regulárne výrazy, avšak tie môžu byť použité iba ako náhrada celého slova, ale nie uprostred slova. Regulárny výraz *„* Einstein"* spracuje, pričom *„Albert Einste*n"* nie.

N-gram je podľa oxfordského slovníka[5] definovaný ako postupnosť *n* za sebou idúcich slov alebo znakov. *Martin* je n-gram veľkosti jedna, teda 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 a je dostupný vo webovom rozhraní. Na obrázku 6 Google Ngram Viewer vidno vizualizáciu výskytu mien *Albert Einstein, Sherlock Holmes, Frankenstein* v knihách od roku 1800 do roku 2000.



Obr. 6: *Google Ngram Viewer*

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

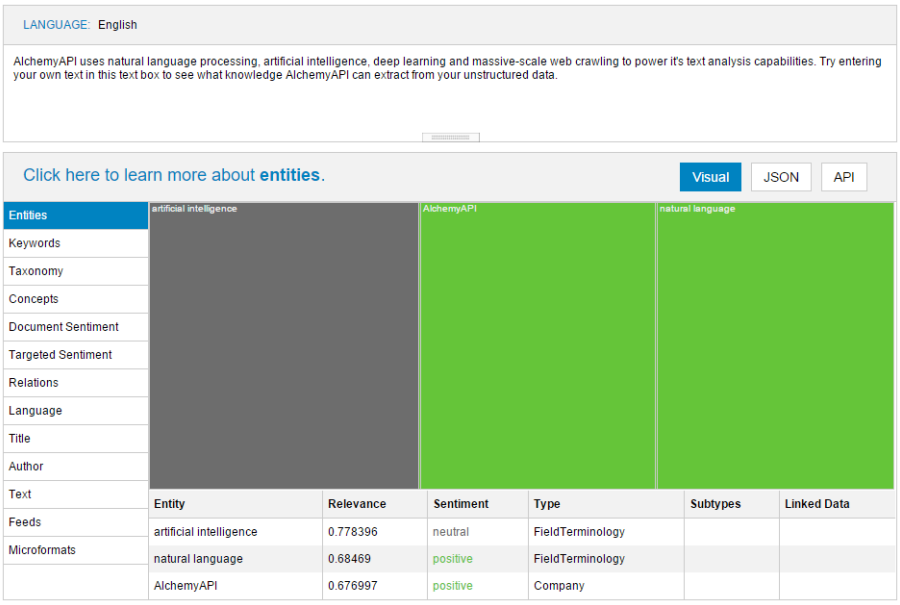
2.3.5 AlchemyAPI

AlchemiAPI[6] je súbor dvanástich nástrojov, z ktorých sú niektoré zamerané na úlohy spracovania prirodzeného jazyka popísané v sekcií 2.2 Úlohy spracovania prirodzeného jazyka, ako napríklad extrakcia entít, extrakcia kľúčových slov, extrakcia vzťahov, ale aj iné zaujímave funkcie, napríklad extrakcia autora z textu.

[5] http://www.oxforddictionaries.com/
[6] www.alchemyapi.com

Na používanie tohto nástroja je potrebná registrácia 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.



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.

2.4 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. V jednej časti nástroja je zdrojový text alebo súbor a v druhej časti výsledný text alebo súbor. Hlavný dôraz sa kladie práve na transformáciu zo zdrojového textu na cieľový. Transformácia môže mať viacero podôb, ako preklad, zarovnanie, zjednodušenie textu, a mnoho ďalších. Texty sú zväčša, pre zjednodušenie transformácie, rozdelené podľa viet, pričom vety na jednej úrovni zvyčajne spolu súvisia podľa určitej vlastnosti.

V nasledujúcich častiach sú predstavení niektorí z predstaviteľov tohto typu nástrojov.

3.1 InterText

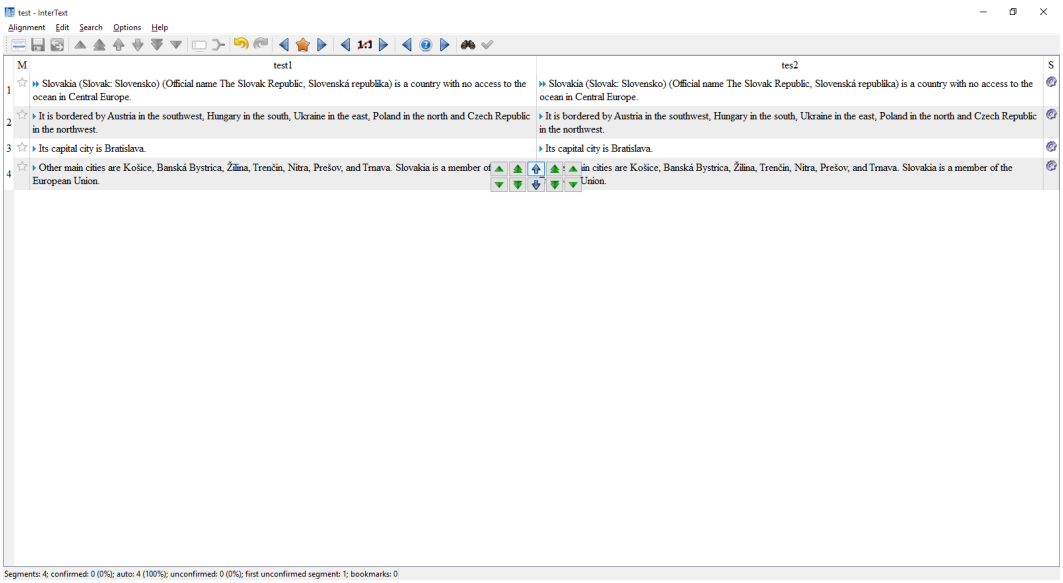
InterText[7] je editor paralelne zarovnaných textov, využívaný na správu viacerých paralelne zarovnaných verzií textu rôznych jazykov na úrovni viet. Táto aplikácia je dostupná vo verzií pre desktop a server.

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.

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



Obr. 8: *Aplikácia InterText*

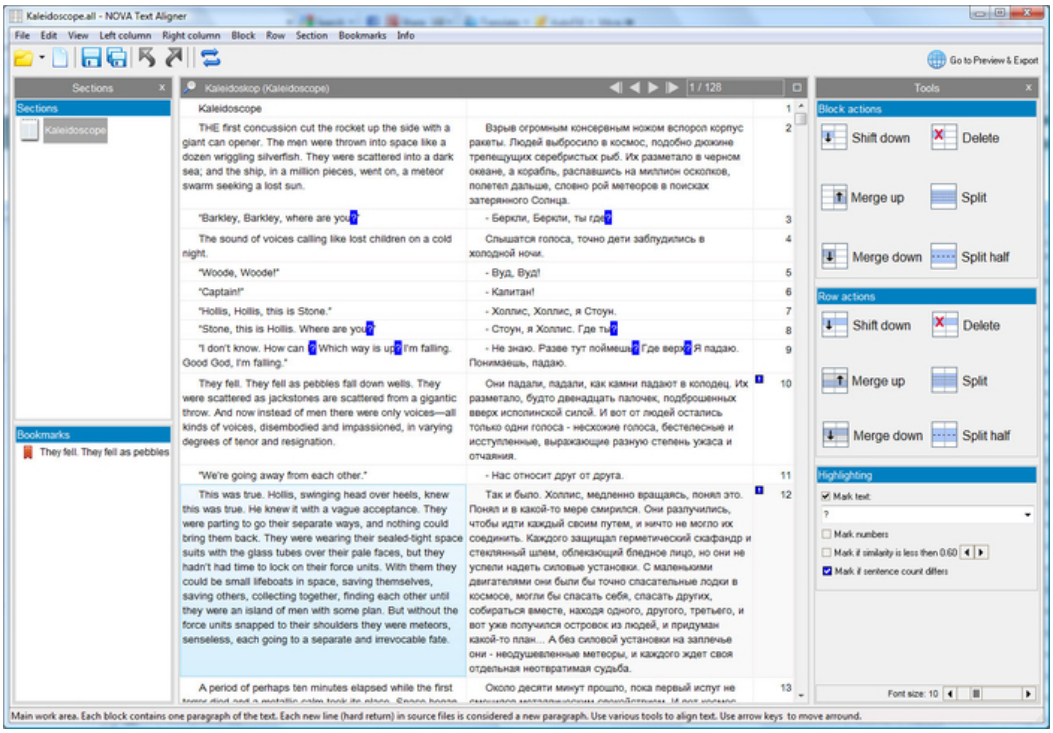
3.2 NOVA Text Aligner

NOVA Text Aligner[8] 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 rozdelená 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.

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

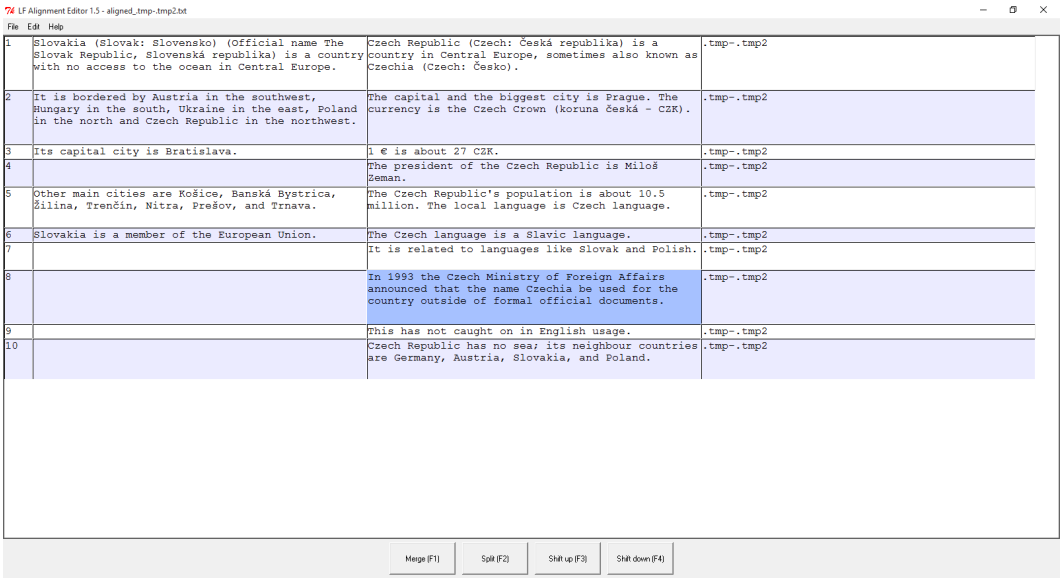


Obr. 9: *Aplikácia NOVA Text Aligner*[9]

3.3 LF Aligner

Aplikácia LF Aligner[10] 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.

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



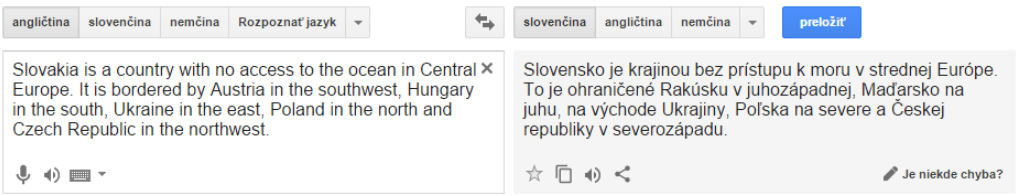
Obr. 10: *Aplikácia LF Aligner*

3.4 Google Translate

Za najznámejšieho zástupcu webových nástrojov na spracovanie paralelných textov sa dá pokladať nástroj Google Translate[11]. 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.

[10] http://parallel-text-aligner.en.softonic.com/
[11] 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 musia byť uložené na externom úložisku, ako napríklad v databáze na serveri.

4 Návrh

4.1 Tvorba poznámok

Pri spracovávaní viet je potreba extrahovať dôležité a konkrétne informácie, v našom prípade slová, a z extrahovaných informácií vytvoriť poznámku. Poznámka je zložená výhradne zo slov obsiahnutých vo vete, z ktorej bola vytvorená. Tým zabezpečíme, že poznámka nebude obsahovať irelevantné informácie z hľadiska vzťahu ku vete.

Spracovávané vety sa môžu opakovať, a preto je dôležité, aby sa rovnaké a podobné vety spracovali vždy takým istým spôsobom. Pre tento účel je vhodné mať definované pravidlo (viď. 4.1.1 Pravidlo na spracovanie), ktoré bude určovať, ako sa má daná veta spracovať. Aplikovaním pravidla na vetu vytvoríme poznámku. Používateľ má možnosť interaktívne upraviť poznámku, a tým pádom aj pravidlo, ktoré bolo použité na vytvorenie danej poznámku, v takom rozsahu, že sa vytvorená poznámka zmení podľa vykonaných zmien. Zmeny v pravidle sa musia prejaviť aj pri nasledujúcom aplikovaní pravidla na rovnakú alebo podobnú vetu. Používateľovi to poskytne kontrolou nad spôsobom, akým sa vytvárajú poznámky a bude si ich vedieť prispôsobiť - personalizovať.

4.1.1 Pravidlo na spracovanie

Pravidlo na spracovanie určuje, ktorá informácie vo vete sú relevantné a majú byť použité v poznámke. V časti 4.1.2 Určenie relevantných informácií je podrobnejšie opísaný proces určenia relevantných informácií. Okrem určenia relevantnosti slov pravidlo taktiež určuje, na ktorej pozícií sa má konkrétna informácia vo výslednej poznámke nachádzať. Výsledná poznámka sa nemusí skladať iba z jednej vety, ale môže pozostávať z ľubovolného počtu viet. Počet viet, z ktorých bude poznámka pozostávať a miesta, na ktorých má byť pôvodná poznámka rozdelená na vety určuje pravidlo.

Skladá sa primárne zo štruktúry, ktorá je zložená hlavne zo závislostí. Viac informácií o štruktúre je v časti 4.3 Štruktúra viet a pravidiel. Okrem štruktúry obsahuje aj informácie o pozíciach, na ktorých sa má výsledná poznámka rozdeliť na menšie časti - vety.

Pravidlo je aplikovateľné nie len na totožné vety, ale na vety s rovnakou štruktúrou a je nezávislé od obsahovej časti vety.

4.1.2 Určenie relevantných informácií

Určenie relevantnosti informácií hlavne pri učebných textoch a poznámkach je subjektívna vec. Informáciu, ktorú niekto považuje za relevantnú môže byť pre iného irelevantná. Relevantnosť informácií preto určuje používateľ pomocou vytvárania nových a úpravy existujúcich pravidiel cez spracovávanie článkov, viet a úpravou vytvorených poznámok. Tým si používateľ prispôsobuje tvorbu poznámok, aby zodpovedala jeho predstave relevantnosti a celkovo tvorbe poznámok. Na tento účel mu systém ponúka interaktívne a intuitívne rozhranie.

4.1.3 Viacnásobné poznámky

Viacnásobná poznámka je poznámka, ktorá sa skladá z viacerých viet, ktoré majú rovnaký základ a sú doplnené o prvok z množiny spojky *a* (angl. and). V praxi to vyzerá tak, že z vety *„It is bordered by Austria in the southwest, Hungary in the south, Ukraine in the east, Poland in the north and Czech Republic in the northwest."* sa dá vytvoriť viacnásobná poznámka v tvare zobrazenej v tabuľke 1 Viacnásobná poznámka.

Tabuľka 1: *Viacnásobná poznámka*

It is bordered by Austria in the southwest.
It is bordered by Hungary in the south.
It is bordered by Ukraine in the east.
It is bordered by Poland in the north.
It is bordered by Czech Republic in the northwest.

Z tabuľky vidno, že rovnaký základ viet vo viacnásobnej poznámke je *It is border by*, ktoré je rozšírené o prvky z množiny {*Austria in the southwest, Hungary in the south, Ukraine in the east, Poland in the north, Czech Republic in the northwest*}.

Množina sa určuje pomocou dvoch závislostí. Prvá je závislosť so vzťahom *conjuction* a špecifikom *and*, ktorá nám určuje, ktoré slová sú prepojené spojkou

a. Druhá využívaná závislosť, je závislosť so vzťahom *appositional modifier*, ktorá určuje slova prepojené čiarkou, keďže spojka *a* môže byť nahradená čiarkou. Pomocou týchto závislostí vieme extrahovať z vety jednotlivé prvky množiny a množinu vytvoriť.

Základ viet vo viacnásobnej poznámke je relevantná informácia z pôvodnej vety určená pravidlom, okrem prvkov množín. Po aplikovaní pravidla a získaní základu viet sa následne namnoží a spojí s jednotlivými prvkami množiny a vytvoria tak viacnásobnú poznámku.

4.2 Použitie závislostí pri tvorbe poznámok

Závislosti majú viacero využití pri tvorbe poznámok. Hlavné z nich sú:

- jednoznačná identifikácia informácie,
- pravidlo závisle iba od štruktúry,
- menší počet potrebných pravidiel.

Závislosť nám zoskupuje informácie o jej tokenoch a vzťahu medzi nimi. Na základe nich vieme, bez potreby iných vstupov, jednoznačne identifikovať konkrétnu informáciu vo vete alebo poznámke.

Pravidlo, ktoré je tvorené prevažne zo závislostí, nie je viazané na konkrétnu vetu, pri spracovávaní ktorej vzniklo. Viaže sa na štruktúru vety (viď. 4.3 Štruktúra viet a pravidiel). Táto vlastnosť nám umožňuje použiť jedno a to isté pravidlo na spracovanie viacerých viet.

Keby sme vytvárali pravidla iba na základe, napríklad značiek názvoslovných druhov a ukladali by sme si v akom poradí boli značky v pôvodnej vete a v akom poradí v poznámke, potrebovali by sme pre každú obmenu vety nové pravidlo. Keďže je pravidlo naviazané na štruktúru vety a nie na obsahovú časť, teda aj poradie slov, vieme jedným pravidlom spracovať viacero obmien jednej vety, za podmienky, že sa obmenou nezmenila štruktúra. Aplikovateľnosťou pravidla na viacero viet sa nám redukuje počet všetkých možných potrebných pravidiel.

4.3 Štruktúra viet a pravidiel

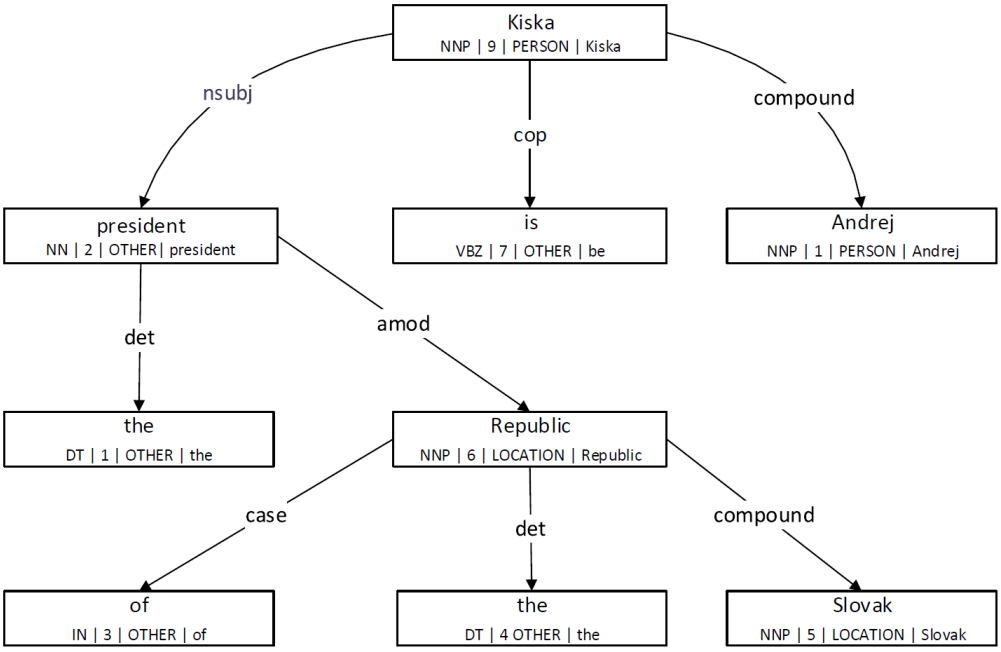
Všeobecná štruktúra je definovaná závislosťami, vzťahmi závislostí, pozíciami závislostí, a informáciami o nadradenom tokene a podradenom tokene závislostí. Tieto informácie sa skladajú zo značky slovného druhu, indexu, názvoslovnej entity a lemy. Štruktúra pravidla je oproti všeobecnej štruktúre, ktorú využíva veta, rozšírená a doplňujúce informácie. Sú to informácie potrebné pre správny chod systému. Skladajú sa z typu porovnania a typu tokenu. V náčrtoch sa nezobrazujú keďže sú využívané interne a nie sú nevyhnutné na pochopenie danej časti.

Štruktúru viet prezentujeme dvoma spôsobmi:

- stromom (viď. 4.3.1 Reprezentácia stromom),
- vo vete (viď. 4.3.2 Reprezentácia vo vete)

4.3.1 Reprezentácia stromom

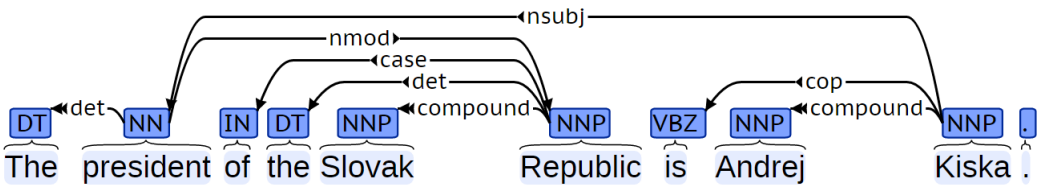
Stromová reprezentácia uľahčuje reprezentáciu vzťahov a prepojení jednotlivých slov vo vete. Táto reprezentácia môže byť pre vetu *„The president of the Slovak Republic is Andrej Kiska."* vyjadrená obrázkom 12 Stromová reprezentácia štruktúry. Reprezentácie je podobná reprezentácií na obrázku 1 Strom vzťahov z kapitoly 2.2.4 Identifikácia gramatických závislostí, avšak je ešte rozšírená o značky slovných druhov, indexy, názvoslovné entity a lemy. V tomto poradí sú tieto informácie zobrazene v stromovej reprezentácia pod príslušným slovom.



Obr. 12: *Stromová reprezentácia štruktúry*

4.3.2 Reprezentácia vo vete

Reprezentácia vo vete taktiež reprezentuje vzťahy a prepojenia slov vo vete. Zobrazenie tejto reprezentácie lepšie opisuje reálne uloženie štruktúry v databáze. Pre vetu *„The president of the Slovak Republic is Andrej Kiska."* vyzerá *reprezentácia vo vete* ako na obrázku 13 Reprezentácia štruktúry vo vete.



Obr. 13: *Reprezentácia štruktúry vo vete*

4.4 Uchovávanie textov v databázach

Text je špecifický údajový model s variabilnou štruktúrou, ktorý potrebujeme efektívne ukladať v databáze. Je nevyhnutné použiť vhodnú databázu, ktorá je prispôsobená na ukladanie textov. Databáza musí obsahovať vlastnosti, ktoré umožňujú jednoduché narábanie s dátami s variabilnou štruktúrou a nemajú prebytočnú spotrebu pamäti pri ich uchovávaní. Taktiež je potrebná podpora bezproblémového ukladanie, získavania, vyhľadávania a spracovania textov na úrovni databázy. Pri výbere vhodnej databázy sme prešli viacero alternatív.

4.4.1 Relačné databázy

Relačné databázy pracujú spoľahlivo pri obmedzenom množstve dát [6]. Pri potrebe aplikácie s obrovským množstvom dát sa rozšíriteľnosť (angl. scalability) a schéma stávajú problémom [8]. Tento typ databáz ukladá dáta v tabuľkách zložených z riadkov a stĺpcov. Výhodou relačných databáz je možnosť jednoduchého vytvorenia prispôsobeného pohľadu na dáta [7]. Z pohľadu ukladania textu, a celkovo dát s variabilnou štruktúrou, je nevýhoda relačných databáz v ťažko prispôsobiteľnej štruktúre.

4.4.2 Textové databázy

Textové databázy ukladajú dáta vo forme dokumentov, vďaka čomu ponúkajú vysoký výkon, horizontálnu rozšíriteľnosť a podporu pre ukladanie dát s variabilnou štruktúrou [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 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. Dátovy typ dokument v textových databázach vyhovuje požiadavkám na databázu pre náš systém. Dokáže jednoducho narábať s dátami s variabilnou štruktúrou, bez nároku na prebytočnú pamäť a s dobrou rozšíriteľnosťou.

4.4.3 Porovnanie a výber typu databázy

Porovnávame textové a relačné databázy. V oboch kategóriách sme vybrali jedného z najväčších predstaviteľov danej kategórie. Za relačné databázy to je *MySQL* a za textové databázy *MongoDB*. Porovnanie poskytovaných prvkov porovnávaných databáz je v tabuľke 2 Porovnanie poskytovaných prvkov.

Tabuľka 2: *Porovnanie poskytovaných prvkov*

	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) je poskytovanie rozšírenej funkcionality dátovým modelom. 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* zabezpečuje zabránenie poklesu priepustnosti operácií čítania a zapisovania pri náraste dát ukladaním dát 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 3 Porovnanie používaných pojmov [6].

Tabuľka 3: *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

Pri ukladaní textov a iných časti systému v databáze potrebujeme dynamickú štruktúru z dôvodu variability štruktúry ukladaných dát. Namapovanie dát na pevne stanovenú štruktúru a vzťahy relačných databáz by bolo veľmi náročné. Preto je výhodnejšie uchovávať dáta pokope v dynamickej štruktúre. Z bodov *dynamická štruktúra* a *lokálnosť dát* z tabuľky 2 Porovnanie poskytovaných prvkov jasne vidno, že pre tento účel je textová databáza vhodnejšia. *Bohatý dátový model* je rovnako veľkou výhodou. V neposlednom rade *MongoDB* automaticky podporuje *sharding*, ktorý sa dá spraviť aj v databáze *MySQL*, ale za cenu výkonnosti. Z týchto dôvodov sme sa rozhodli pre použitie textovej databázy v našom systéme.

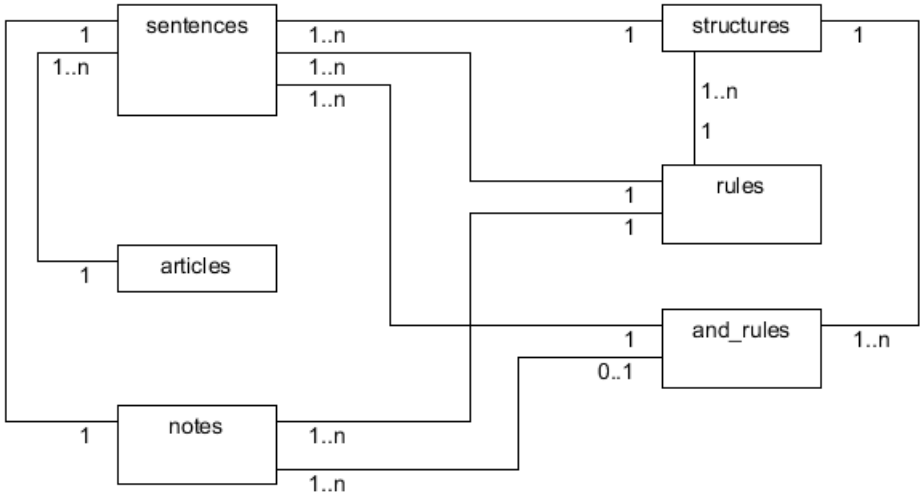
4.5 Návrh uchovávania textov v databázach

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 14 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. 14: *Databázový model*

4.5.1 Kolekcia articles

V kolekcii *articles* sa ukladajú spracovávané texty.

Kolekcia obsahuje textové pole *text* na uloženie textu v pôvodnom tvare. Model kolekcie *articles* je zobrazený na obrázku 15 Model kolekcie articles.

Pole	Typ
_id	ObjectId
text	String
created_at	DateTime
updated_at	DateTime

Obr. 15: *Model kolekcie articles*

4.5.2 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 16 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. 16: *Model kolekcie notes*

4.5.3 Kolekcia sentences

V následujúcej kolekcii *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_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 17 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. 17: *Model kolekcie sentences*

4.5.4 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 *dependent* 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 18 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. 18: *Model kolekcie structures*

4.5.5 Kolekcia rules

Rules je kolekcia, do ktorej sa ukladajú pravidla na vytvorenie poznámok z viet. Vďaka databázovému modelu na obrázku 14 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 19 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, lebo koniec vety, v tomto prípade bodka, sa nachádza na tretej pozícií spomedzi tokenov vo vete. Číslovanie pozícií začína od nuly. 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. 19: *Model kolekcie rules*

4.5.6 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 vo viacnásobnej poznámke. Model kolekcie je vyobrazený na obrázku 20 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. 20: *Model kolekcie and rules*

4.5.7 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.

4.6 Zhrnutie

Dummy text..

5 Systém

dummy text..

5.1 Vybrané nástroje

Do nášho systému sme zakomponovali nástroje, ktoré sú jedným z najlepších vo svojej kategórií a najviac vyhovujú potrebám systému. Pre databázovú vrstvu sme zvolili databázu MongoDB (viď. 5.1.1 MongoDB) a nástroj na spracovanie prirodzeného jazyka sme vybrali StanfordNLP (viď. 5.1.2 StanfordNLP).

5.1.1 MongoDB

Pre databázovú vrstvu nášho systému používame popredného predstaviteľa textových databáz MongoDB[12]. Ukladá dáta v dokumentoch vo formáte BSON (Binary JSON), ktorých štruktúra sa môže meniť. Využíva dynamickú štruktúru, 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].

5.1.2 StanfordNLP

Nástroj StanfordNLP je všeobecne popísaný v časti 2.3.2 StanfordNLP. Vybrali sme ho, lebo umožňuje jednoduchým spôsobom získať esenciálne informácie pre náš systém ohľadne textov. Sú to závislostí slov vo vetách, značky slovných druhov, názvoslovné entity, indexy a lemy slov.

[12] www.mongodb.org

5.2 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 daným 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.

5.2.1 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 časť,
- obsahová časť,
- hodnotová časť.

Štrukturálna časť zhody odzrkadľuje percentuálnu zhodu dvoch štruktúr. Pri tejto zhode zisťujeme, či štruktúry obsahujú tokeny závislostí s nadradenými značkami slovných druhov a ich konkrétny počet. Štrukturálna časť zhody 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 tokenmi s rovnakými nadradenými značkami slovných druhov. Definícia nadradenej značky slovného druhu je v časti Nadradená značka slovného druhu na strane 37. 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.

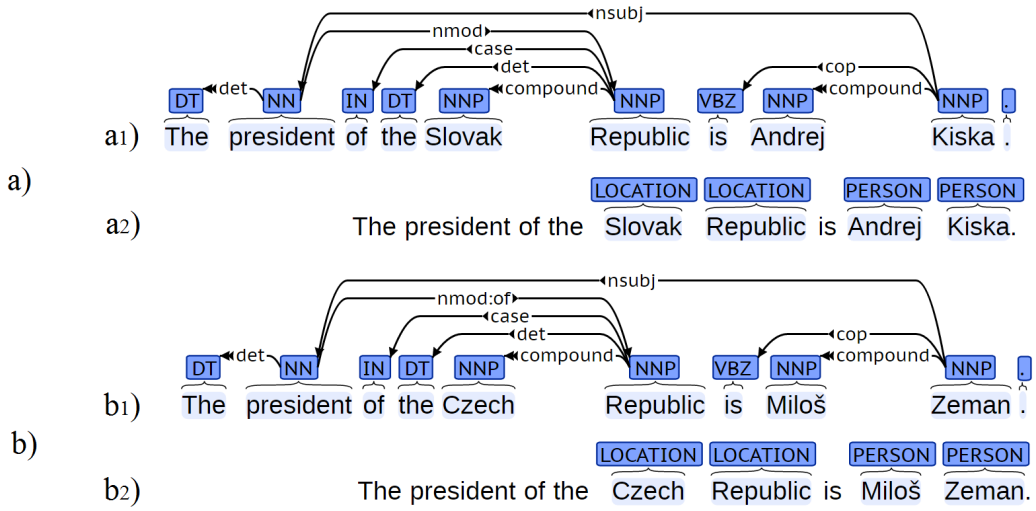
Obsahová časť zhody zodpovedá percentuálnej zhode obsahu dvoch viet. Kontrolujú sa indexy slov, konkrétne značky slovných druhov a názvoslovné entity. Pri obsahovej zhode majú vety *„The president of the Slovak Republic is Andrej Kiska."* a *„The president of the Czech Republic is Miloš Zeman."* obsahovú 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 sú 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 časť zhoda, reprezentuje úplnú zhodu dvoch viet. Veta má hodnotovú časť zhody 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 Výpočet zhody viet zobrazuje pseudo algoritmus výpočtu zhody, konkrétny príklad je zobrazený na obrázku 21 Príklad určenia zhody viet

Algoritmus 1 Výpočet zhody viet

```
1: procedure VYPOCETZHODYVIET(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, závislosż) then
7:                     do zhody na type porovnania pripočítaj ohodnotenie kroku
       return zhoda
```



Obr. 21: *Príklad určenia zhody viet*

Predpokladajme situáciu z obrázka 21 Príklad určenia zhody viet. V databáze máme uloženú spracovanú vetu *a* aj s pravidlom pre túto vetu. Spracovávame vetu *b*. Na časti *a1* obrázka 21 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.

Následne sa určuje obsahová časť zhody. 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%.

Nadradená značka slovného druhu

Pod nadradenou značkou slovného druhu sa chápe značka slovného druhu zoskupujúca množinu značiek slovných druhov, do ktorej značka slovného druhu patrí.

Napríklad značka slovného druhu VBD (Verb, past tense - sloveso v minulom čase) patrí medzi skupinu značiek slovných druhov slovies {*VB, VBD, VBG, VBN, VBP, VBZ*}. Z toho vyplýva, že nadradená značka slovného druhu *VBD* je VB (Verb - sloveso).

5.2.2 Aplikovanie pravidla

Procesom vyhľadania pravidla (viď. 5.2.1 Vyhľadanie pravidla) sme získali pravidlo na spracovanie vety. Aplikáciou pravidla na vetu vytvoríme poznámku.

Proces aplikovania pravidla na vetu s cieľom vytvorenia poznámky má viacero krokov. Pre všetky závislosti zo *zoznamu dát štruktúry* pravidla, príslušná závislosť je vyhľadaná v spracovávanej vete. Pri vyhľadávaní príslušnej závislosti sa závislosti neporovnávajú, okrem iného, na základe značiek slovných druhov svojich tokenov, ale podľa nadradených značiek slovných druhov (viď. Nadradená značka slovného druhu na strane 37) svojich tokenov. Tento spôsob vyhľadávania nám umožňuje aplikovať jedno pravidlo na viacero viet, ako už bolo spomenuté v časti 4.2 Použitie závislostí pri tvorbe poznámok. Avšak, môže to spôsobiť vyhľadanie viac ako jednej príslušnej závislosti. Preto musí byť vypočítaná zhoda závislostí (viď. Výpočet zhody závislostí na strane 39). Po vypočítaní zhody závislostí a získaní závislosti s najväčšou zhodou, slovo korešpondujúce s tokenom, ktorý sa má z danej závislosti vybrať, sa pridá do poznámky na pozíciu pozície závislosti. Po spracovaní všetkých závislostí, posledné minoritné úpravy sú vykonané nad poznámkou, ako 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 Aplikovanie pravidla.

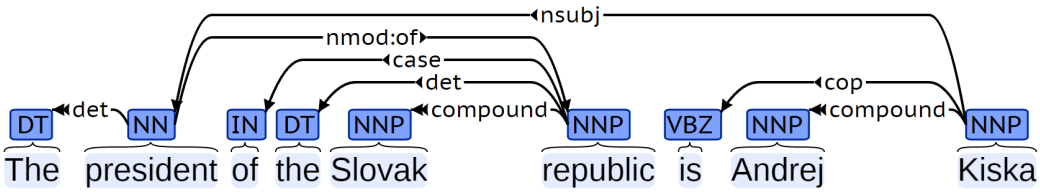
Pre vetu *„The president of the Slovak republic is Andrej Kiska."* nám nástroj

Algoritmus 2 Aplikovanie pravidla

```
1: procedure APLIKUJPRAVIDLO(veta, pravidlo)
2:     poznámka ← vytvor prázdnu poznámku
3:     for all závislostí v pravidle do
4:         závislosż ← nájdiZávislosť(veta, závislosż pravidla)
5:         if závislosż existuje then
6:             pridaj závislosż do poznámky
7:     rozdeľ poznámku na vety podľa pravidla
       return poznámka
```

Stanford CoreNLP poskytne závislostí vyobrazené na obrázku 22 Závislostí jednoduchej vety. Ak na túto vetu aplikujeme pravidlo so štruktúrou v tvare zobrazenej na obrázku 23 Príklad štruktúry pravidla, výsledná poznámka bude *„President is Kiska."*.

Aplikovanie pravidla prebieha nasledovným spôsobom. Prechádzajú sa všetky závislosti v štruktúre pravidla. Prvá závislosť v štruktúre pravidla je závislosť so vzťahom *nsubj* na pozícií jedna a podradeným korešpondujúcim tokenom. Má podradený token so značkou slovného druhu *NN*, typom názvoslovnej entity *OTHER - ostatné*, lemou *President* a indexom dva. Nadradený token má značku slovného druhu *NNP*, názvoslovnú entitu *PERSON - osoba*, lemu *Kiska* a index deväť. Takáto závislosť sa vyhľadá v štruktúre vety medzi závislosťami na obrázku 22 Závislostí jednoduchej vety. Závislosti sa vyhľadávajú podľa zhody všetkých informácií o ich tokenoch a vzťahu medzi nimi. Ak pre danú závislosť vyhovuje viacero závislostí, pomocou výpočtu zhody závislostí vyberieme tú s najväčšou zhodou. V tomto prípade vidíme, že vyhovujúca závislosť je len jedna a to prvá závislosť *nsubj* medzi slovom na druhej pozícií *president* a posledným slovom, na deviatej pozícií *Kiska*. Z tejto závislosti sa zoberie podradený token, keďže tak určuje pravidlo v stĺpci *typ tokenu*. Slovo *president* sa pridá do poznámky na pozíciu jedna. Rovnakým spôsobom sa prechádzajú a spracujú všetky závislosti v štruktúre pravidla a podľa nich sa extrahuje slovo z vety a pridá do poznámky.



Obr. 22: *Závislostí 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. 23: *Príklad štruktúry pravidla*

Výpočet zhody závislostí

Princíp výpočtu zhody závislostí je veľmi podobný s výpočtom zhody viet zo sekcie 5.2.1 Vyhľadanie pravidla. Porovnávajú sa vždy nadradené aj podradené tokeny. Porovnanie má niekoľko krokov. Začína sa s porovnaním značiek slovných druhov. Pokračuje sa názvoslovnou entitou, indexom, lemou a nakoniec sa porovnáva vzdialenosť pozícií tokenov vo vetách. Každý krok je príslušne ohodnotený a ak porovnanie bolo úspešné, ohodnotenie sa pripočíta k finálnej hodnote reprezentujúcej percentuálnej zhody závislostí.

5.2.3 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 poznámka.

Zo závislostí pôvodnej vety a informácií o ich tokenoch sa vytvorí nový záznam o štruktúre pôvodnej vety. Tak isto sa vytvorí aj nový záznam o štruktúre poznámky. Z poznámky sa vytvorí záznam o novej poznámke a následne sa z nej

vytvorí zoznam koncov viet. Tento zoznam spolu so štruktúrou poznámky vytvorí záznam nového pravidla. Z pôvodnej vety sa vytvorí záznam o vete a prepojí sa so štruktúrou pôvodnej vety, článkom, ktorý obsahoval danú vetu a pravidlom, ktoré bolo vytvorené a použité a s poznámkou, ktorá vznikla z vety. Týmto vznikne nové pravidlo na spracovanie takej istej alebo podobnej vety ako sme práve spracovali.

5.2.4 Úprava pravidla

Systém umožňuje používateľovi upraviť vytvorenú poznámku. Tá sa da upraviť z množiny slov pôvodnej vety, pričom sa dajú ľubovoľne usporiadať a definovať ľubovoľný počet viet, na ktoré poznámku rozdeliť. Úpravou poznámky sa systém „naučí" nové pravidlo alebo upraví existujúce pravidlo. Ktorá z týchto dvoch akcií sa vykoná sa rozhoduje podľa zhody, ktorá bola určená pri spracovávaní pôvodnej vety poznámky.

Ak je štrukturálna časť zhody menšia ako 100%, tým pádom aj obsahová a hodnotová zhoda je pod 100%, tak sa systém naučí nové pravidlo. Znamená to, že systém doposiaľ nespracovával vetu s rovnakou štruktúrou, a tak použil pravidlo podľa vety s najpodobnejšou štruktúrou a po úprave poznámky sa naučí, ako spracovať vetu s danou štruktúrou.

V prípade, že štrukturálna časť zhody je 100%, ale obsahová nie je úplná a tým pádom ani hodnotová, znamená to, že systém spracoval vetu s rovnakou štruktúrou, ale iným obsahom. V tomto prípade systém pozná ako spracovať vetu s danou štruktúrou, takže sa štruktúra pravidla upraví podľa vykonaných zmien. Vytvorí sa nový záznam o poznámke a aj o pôvodnej vete a prepoja sa na upravené pravidlo, pomocou ktorého bude systém vedieť spracovať viacero viet.

Pri 100% štrukturálnej, obsahovej aj hodnotovej časti zhody, systém spracoval identickú vetu, akú už v minulosti spracoval. Preto sa zmeny na vytvorenej poznámke prejavia pri upravení pravidla. Tak isto sa upraví aj vytvorená poznámka, aby reflektovala vykonané zmeny.

5.3 Priestor na rozvoj

Systém ponúka priestor na jeho ďalší vývoj. Dostupných je niekoľko oblastí, v ktorých by sa dal ďalej rozvíjať.

Aby bolo vytváranie poznámok čo najefektívnejšie, je potrebné vstupný text predspracovať. Elimináciou nepodstatných častí textu, konkrétne viet, by sa výrazne zredukoval počet viet, ktoré musí systém alebo používateľ spracovať. Eliminácia nepodstatných viet by nemusela vykonávať známymi algoritmami, ale mohla by sa vykonávať napríklad na základe kľúčových slov v poznámkach. Podľa toho ako by si používateľ upravoval poznámky a tým definoval pravidla, by sa systém vedel naučiť podľa výskytu kľúčových slov v poznámkach, ktoré informácie sú pre používateľa dôležité. Na základe toho by vedel určiť dôležitosť vety pre používateľa a eliminovať ju ak by nevyhovovala rozmedziu, ktoré by stanovovalo, ktoré vety eliminovať, a ktoré nie.

Ďalšou oblasťou rozvoja by mohlo byť definovanie si vlastných závislostí. Vďaka vlastným závislostiam by nebol systém odkázaný na závislostí slov vo vete a hľadaní vzoru vo vetách na vytvorenie poznámky. Mohol by byť modifikovaný na hľadanie iných vzorov, napríklad vo vetách.

5.4 Zhrnutie

Dummy text..

6 Výsledky

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

6.1 Podčasť

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

7 Záver

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

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

Literatúra

[1] 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. 24: *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. 25: *Legenda diagramov kolekcií*