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 informácií a tvorbu poznámok z textu, ktoré sú dôležité z pohľadu výučby, a umožní študentom vytvárať personalizované poznámky. Využívame hlavne syntaktickú analýzu textu. Poznámky sa vytvárajú pomocou využitia značiek slovných druhov a závislostí medzi slovami vo vetách. Výsledkom je interaktívny systém na tvorbu poznámok, ktoré sa vytvárajú 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.

## POĎAKOVANIE

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.....

## ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že bakalársku prácu s názvom: Spracovanie učebných textov som vypracoval samostatne, na základe konzultácií a štúdia odbornej literatúry. Neporušila som autorský zákon a zoznam použitej literatúry som uviedol na príslušnom mieste.

.....

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. V procese vzdelávania je problém vytvoriť poznámky zo zdroja, ktoré by zahrňovali podmnožinu dôležitých informácií.

Väčšina učebných 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 disciplína, ktorá sa zaoberá spracovaním prirodzeného jazyka strojmi. V našom systéme využívame niekoľko úloh tejto disciplíny.

Náš navrhovaný systém sa zameriava na tvorbu poznámok z viet pomocou extrakcie relevantných informácií z nich. Na tvorbu poznámok sa používa najmä 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 časovo náročný. Pomocou nami navrhnutého systému by si vedeli z dát vytvoriť poznámky a tak získať podmnožinu, pre nich relevantných informácií, rýchlejšie. Vytvorené poznámky by boli personalizované, teda priamo prispôsobené tvorbe poznámok používateľa a tým väčšmi zredukujú potrebný čas, keďže používateľ nebude musieť získané relevantné informácie ešte následne upravovať.

## 1.2 Prehľad práce

Práca začína časťou analýzy 2 Spracovanie prirodzeného jazyka, v ktorej je spracovanie prirodzeného jazyka, jeho úlohy a nástroje na podporu spracovania priro-

dzeného jazyka. Analýza pokračuje opisom a rozborom nástrojov na spracovanie paralelných textov v časti 3 Analýza nástrojov na správu paralelných textov. V časti 4 Návrh systému na tvorbu poznámok sú rozpísané návrhové rozhodnutie systému na tvorbu poznámok. Implementačné rozhodnutie sú definované v časti 5 Systém na tvorbu poznámok. Nasleduje časť 6 Experimenty, v ktorej sa nachádzajú vyhodnotenia naším experimentov. Práca je ukončená časťou 7 Zhodnotenie.

# 2 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 typickými charakteristikami. Navyše, každý človek je odlišný, 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.

## 2.1 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) 2.1.1,
- Rozdelenie vety na menšie časti (angl. Chunking),
- Rozpoznávanie názvoslovných entít (angl. Named Entity Recognition) 2.1.2,
- Označovanie sémantického postavenie (angl. Semantic Role Labeling),
- Rozpoznanie koreferencií (angl. Coreference resolution) 2.1.3,
- Morfologické segmentovanie (angl. Morphological Segmentation),
- Generovanie prirodzeného jazyka (angl. Natural Language Generation),

– Optické rozoznávanie textu (angl. Optical Character Recognition),
– Rozloženie vzťahov (angl. Dependency parsing) 2.1.4,
– 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. 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.

### 2.1.1 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*.

### 2.1.2 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].

Problémy 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.

### 2.1.3 Identifikácia koreferencií

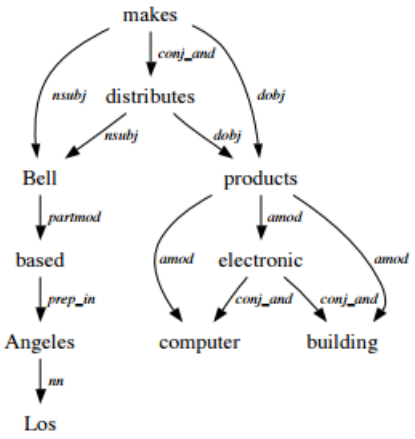
Identifikácia koreferencií (angl. Coreference resolution) predstavuje nájdenie alternatívnych pomenovaní rovnakej entity na základe referencií a kontextu. V texte sa často používajú zámena (angl. pronouns) *to*, *tí*, *on*, anglicky *it*, *those*, *he* alebo menné frázy (angl. noun phrase). Tieto referencie sa odkazujú na iné podstatné mená alebo mená a názvy. Rozpoznávanie koreferencií identifikuje entitu, väčšinou z reálneho sveta, na ktorú sa referencia odkazuje. Táto úloha spracovania prirodzeného jazyka sa využíva v aplikáciách spracovania prirodzeného jazyka, ako sú extrakcia informácií (viď. 2.1.5 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*.

### 2.1.4 Identifikácia gramatických závislostí

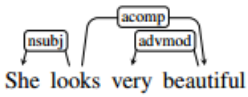
Identifikácia vzťahov nám poskytuje jednoduchý opis gramatických vzťahov slov vo vete. Aplikovaním identifikácie vzťahov na vetu *Bell, based in Los Angeles, makes and distributes electronic, computer and building products.* vznikne strom vzťahov (angl. dependency tree) (viď. Obr. 1) [4].




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 Obr. 2 je vidno, 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 Obr. 2 vidno, okrem iných aj závislosť, ktorej nadradený token je slovo *looks*, podradený token je slovo *She* a vzťah je *nsubj*.

### 2.1.5 Extrakcia informácií

Systémy a aplikácie zamerané na extrakciu informácií vyhľadávajú a extrahujú informácie z textov, článkov a dokumentov, pričom reagujú na používateľove

požiadavky. Výstup 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].

Vyhľadanie informácií a extrakcia informácií spolu úzko súvisia. Prvé využitie slúži na vyhľadávanie relevantných zdrojov informácií v databázach textov, článkov a dokumentov podľa používateľových potrieb. Na vyhľadaných zdrojoch následne prebehne extrakcia informácií.

## 2.2 Nástroje na spracovanie prirodzeného jazyka

V súčasnosti je vyvinutých, alebo sú vo vývoji, viacero nástrojov, ktoré sa dajú použiť pri spracovávaní prirodzeného jazyka. Vývoj takýchto nástrojov je podporovaný na známych univerzitách ako sú napríklad Princeton, Stanford alebo Camridge, ale aj mimo univerzít napríklad v Google.

### 2.2.1 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.

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. 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

[1] www.wordnet.princeton.edu

nie je typické pre nábytok. Prídavné mená obsahujú aj vzťah antonymity, takže slovo *suchý* bude prepojené so slovom *mokrý* ako so svojím antonymom.

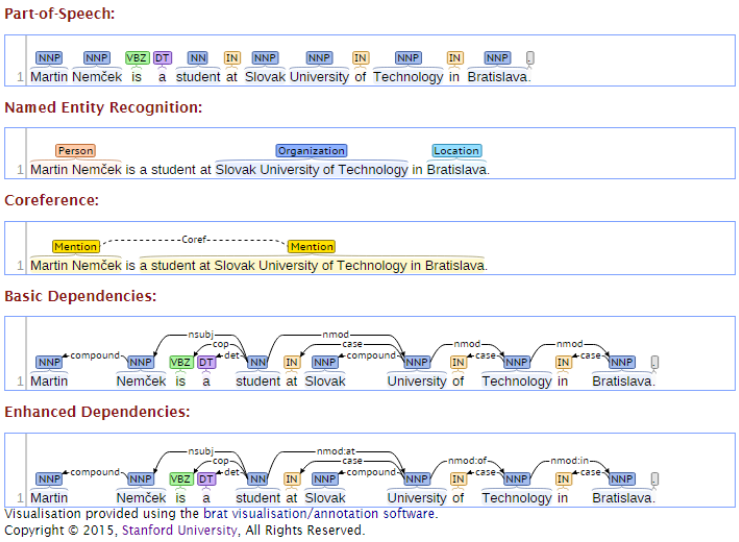
Tento nástroj je dostupný vo webovej verzií, ale ponúka aj stiahnutie jeho databázových súborov.

### 2.2.2 StanfordNLP

Nástroj StanfordNLP[2] je vyvíjaný na Stanfordskej univerzite. Skladá sa z niekoľkých softvérov, ktoré sa zameriavajú na úlohy spracovania prirodzeného jazyka. Sú to *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 Obr. 3 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.”.




Obr. 3: *StanfordNLP online demo*

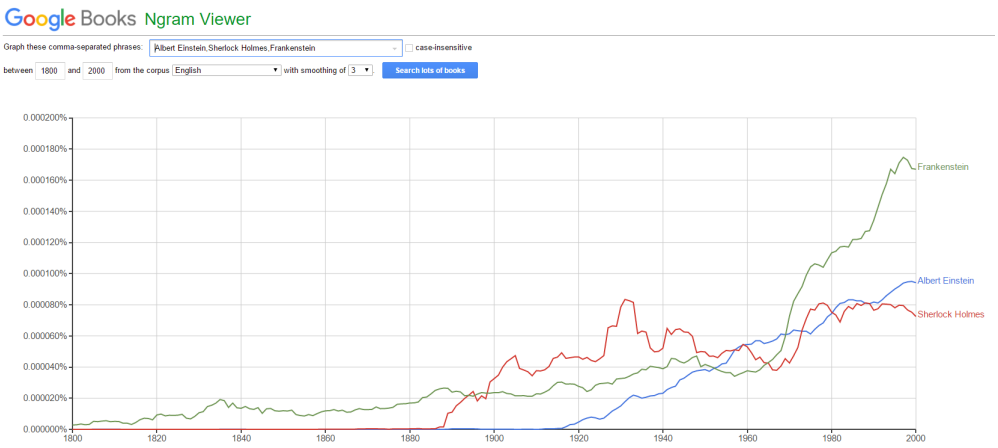
[2]www.nlp.stanford.edu

### 2.2.3 Google Ngram

Google Ngram[3] je vyvinutý spoločnosťou Google. 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, 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[4] 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 Obr. 4 vidno vizualizáciu výskytu mien *Albert Einstein, Sherlock Holmes, Frankenstein* v knihách od roku 1800 do roku 2000.




Obr. 4: *Google Ngram Viewer*

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

[3] www.books.google.com/ngrams
[4] http://www.oxforddictionaries.com/

### 2.2.4 AlchemyAPI

AlchemiAPI[5] je súbor dvanástich nástrojov, z ktorých sú niektoré zamerané na úlohy spracovania prirodzeného jazyka popísané v časti 2.1 Úlohy spracovania 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 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. Dostupné je aj online webové 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.

## 2.3 Zhrnutie

Spracovanie prirodzeného jazyka napomáha spracovávaniu, pochopeniu alebo generovaniu ľudského jazyka strojmi. Rieši viacero úloh, ktoré sú využívané v niekoľkých smeroch. Z pohľadu spracovania učebného textu sú dôležité hlavne úlohy *značkovanie slovných druhov*, *rozpoznávanie názvoslovných entít*, *rozloženie vzťahov* využívané v smere *extrakcií informácií*. Existuje viacero nástrojov na podporu spracovania prirodzených jazykov. Veľká časť je vyvíjaná na univerzitách, ako aj nástroj *StanfordNLP*, ktorý ponúka informácie ako sú značky slovných druhov, typy názvoslovných entít, závislosti a ďalšie.

[5] www.alchemyapi.com

# 3 Analýza nástrojov na správu paralelných textov

Dostupnosť aplikácií na spracovanie prirodzeného jazyka je veľká a široká. My sa zameriavame na aplikácie, ktoré umožňujú editovať paralelný text.

Nástroje na správu paralelných textov uľahčujú spracovanie viacerých druhov a verzií toho istého 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 následujúcich častiach je zanalyzovaných niekoľko nástrojov tohto typu.

## 3.1 InterText

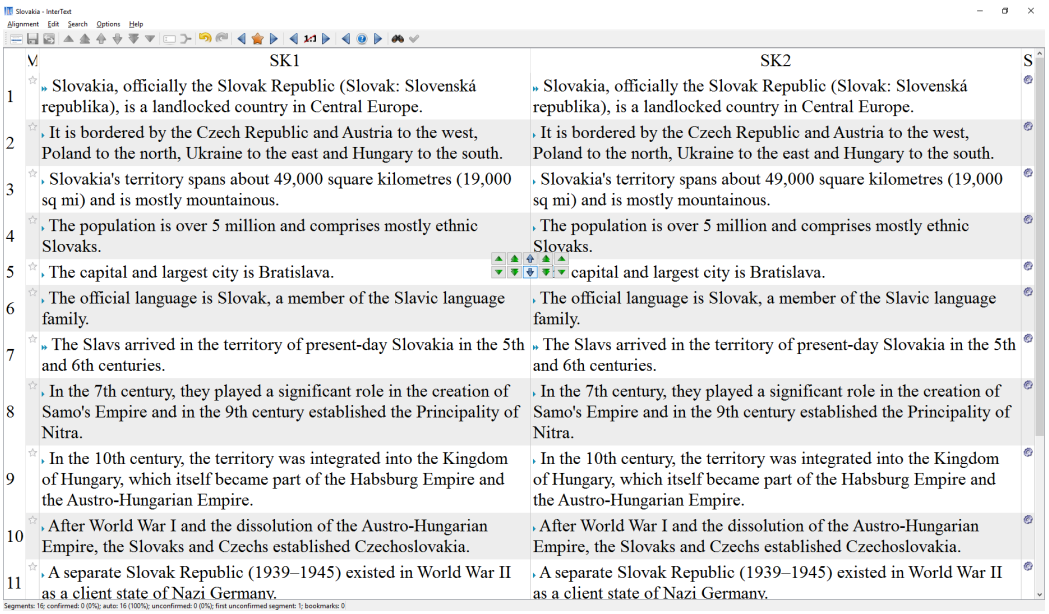
InterText[6] je textov, využívaný na správu viacerých paralelne zarovnaných verzií textu rôznych jazykov na úrovni viet. Táto aplikácia je dostupná vo verzií pre desktop a server.

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 Obr. 5 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. 5: *Aplikácia InterText*

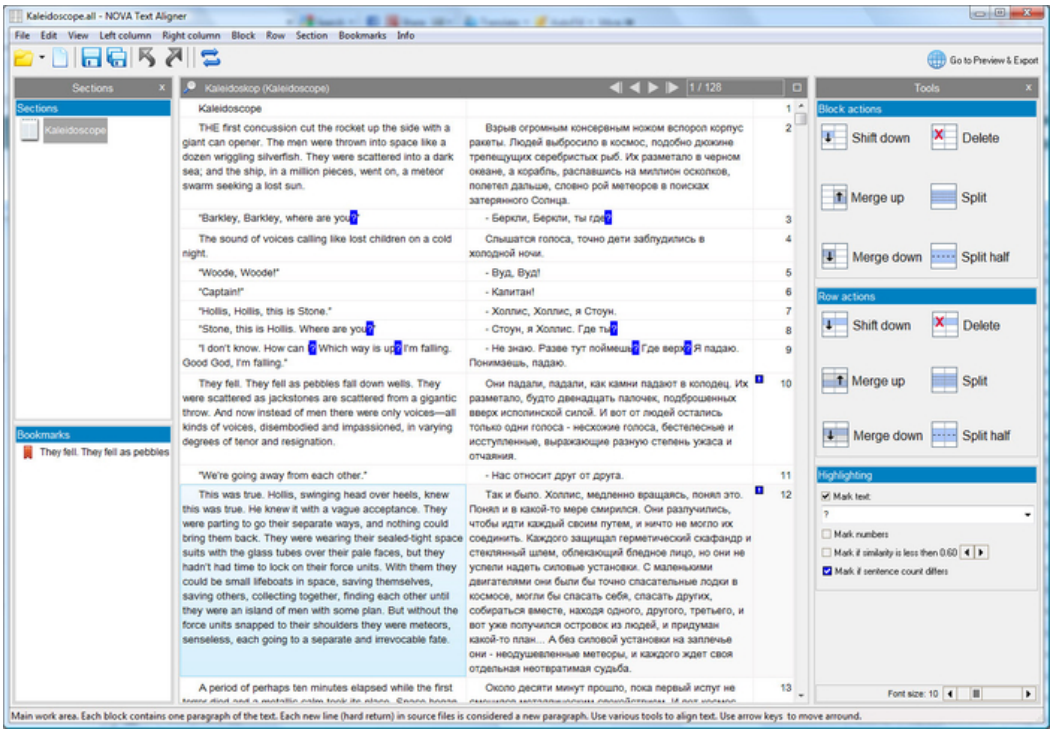
## 3.2 NOVA Text Aligner

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 Obr. 6, 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.

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

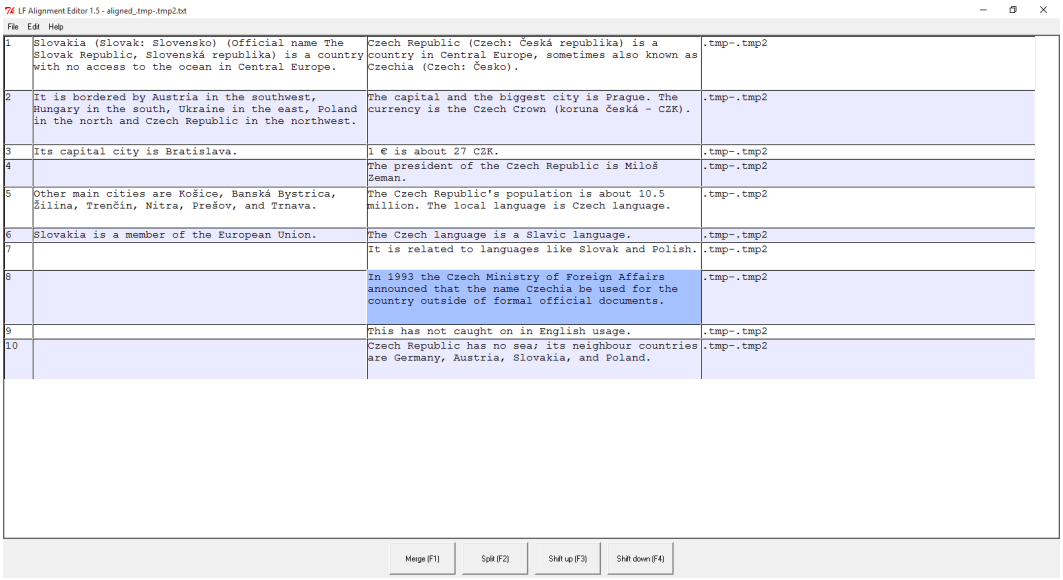



Obr. 6: *Aplikácia NOVA Text Aligner*[8]

## 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 Obr. 7 vidno grafické rozhranie tejto aplikácie, ktoré ponúka mnohé vymoženości. 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



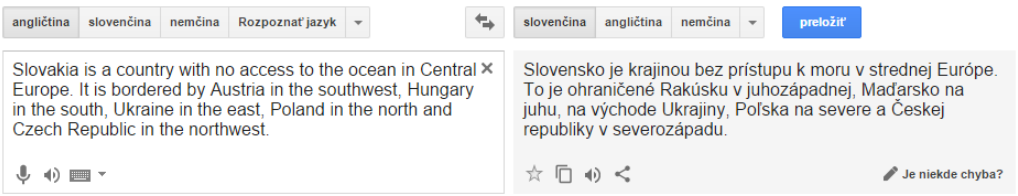

Obr. 7: *Aplikácia LF Aligner*

## 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 Obr. 8 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. 8: *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, ako systém spracováva 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 systému na tvorbu poznámok

Systém na tvorbu poznámok vyžaduje splnenie viacerých požiadaviek. Hlavné z nich sú vytvorenie poznámky a možnosť jej následnej interaktívnej úpravy. Ďalej sú to tvorba viacnásobných poznámok, vytváranie poznámok na základe pravidiel s využitím závislostí slov, ktoré môžu byť aplikované na viacero viet, výber vhodnej databázy s adekvátne vytvoreným databázovým modelom. Spomenuté požiadavky sú opísané v nasledujúcich častiach.

## 4.1 Tvorba poznámok

Pri spracovávaní viet potrebné 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, ktoré informácie vo vete sú relevantné a majú byť použité v poznámke. 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 najmä zo závislostí. Viac informácií o štruktúre je v časti 4.3 Štruktúra viet a pravidiel. Okrem štruktúry obsahuje aj informácie o pozíciách, na ktorých sa má výsledná poznámka rozdeliť na menšie časti - vety.

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

### 4.1.2 Určenie relevantných informácií

Určenie relevantnosti informácií pri učebných textoch a poznámkach je subjektívne. 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 závislé 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 pravidlá napríklad, iba na základe 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

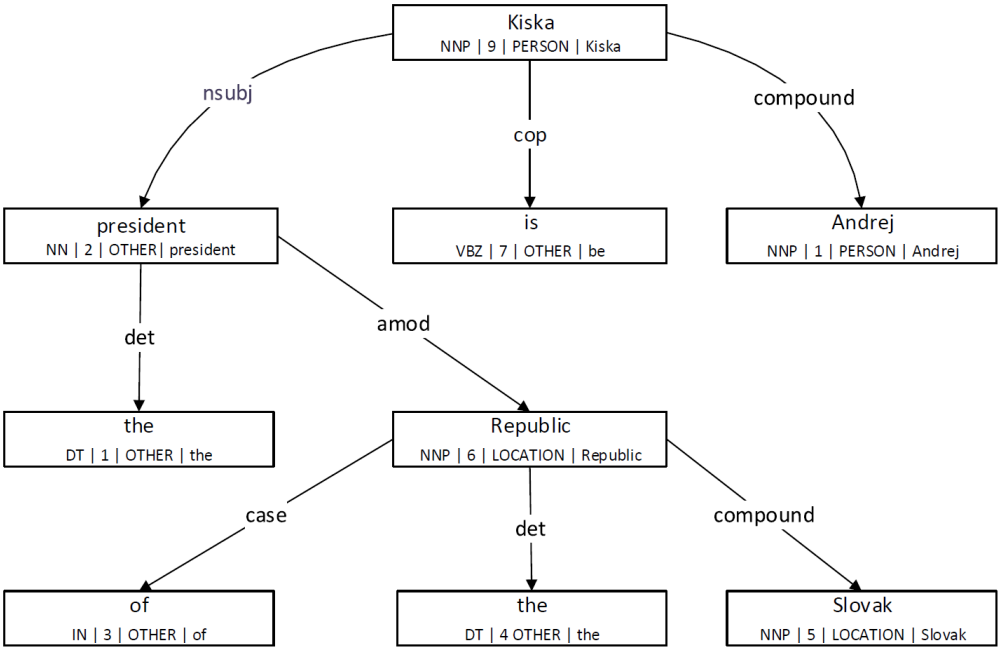
Všeobecná štruktúra je definovaná závislosťami, vzťahmi závislostí, pozíciami závislostí, a informáciami o nadradenom tokene a podradenom tokene závislostí. Tieto informácie sa skladajú zo značky slovného druhu, indexu, názvoslovnej entity a lemy. Štruktúra pravidla je oproti všeobecnej štruktúre, ktorú využíva veta, rozšírená o 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

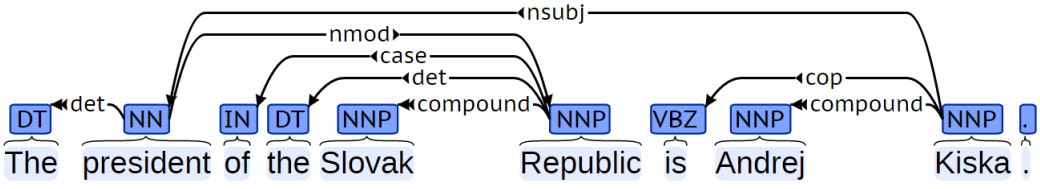
Stromová reprezentácia uľahčuje reprezentáciu vzťahov a prepojení jednotlivých slov vo vete. Táto reprezentácia môže byť pre vetu *„The president of the Slovak Republic is Andrej Kiska."* vyjadrená Obr. 9. Reprezentácie je podobná reprezentácií na Obr. 1 z časti 2.1.4 Identifikácia gramatických závislostí, 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. 9: *Stromová reprezentácia štruktúry*

### 4.3.2 Reprezentácia vo vete

Reprezentácia vo vete taktiež reprezentuje vzťahy a prepojenia slov vo vete. Zo zobrazenia tejto reprezentácie sa dá jednoduchšie pochopiť celá štruktúra, keďže zobrazuje slová v pôvodnom poradí a celkovo vetu v originálnom formátovaní. Pre vetu *„The president of the Slovak Republic is Andrej Kiska."* vyzerá *reprezentácia vo vete* ako na Obr. 10.




Obr. 10: *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.

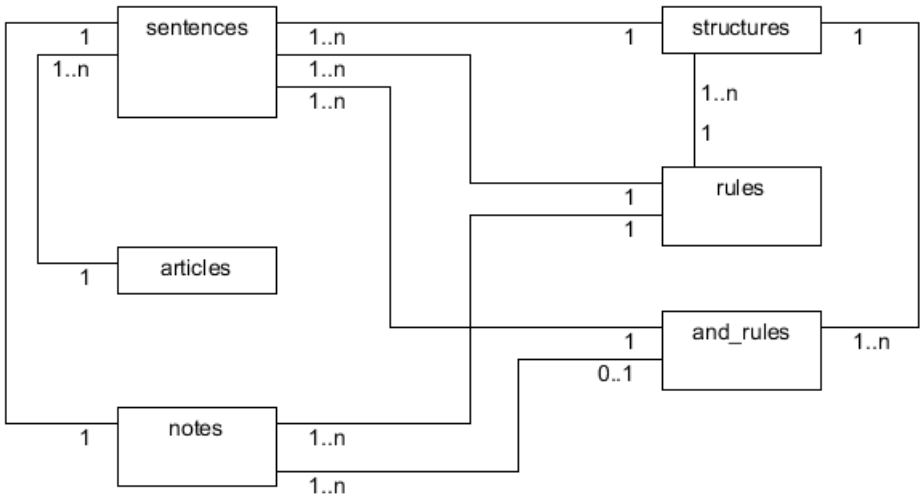
## 4.5 Návrh uchovávania textov v databázach

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 Obr. 11. 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. 11: *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 Obr. 12.

| Pole | Typ |
|---|---|
| _id | ObjectId |
| text | String |
| created_at | DateTime |
| updated_at | DateTime |

Obr. 12: *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 Obr. 25 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. 13: *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 Obr. 26.

| 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. 14: *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 Obr. 27.

| 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. 15: *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 Obr. 11 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 Obr. 28.

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. 16: *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 Obr. 29.

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

Obr. 17: *Model kolekcie and rules*

### 4.5.7 Zhrnutie kolekcií

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 kolekcií nám uľahčuje aj prípadne neskoršie rozšírenie databázového modelu alebo zmenu konkrétnych kolekcií. Taktiež uľahčuje prípadné klastrovanie databázy, ak by bolo nutné, keďže každá kolekcia by mohla byť na samostatnom serveri.

## 4.6 Zhrnutie

Systém dokáže vytvoriť poznámku z viet na základe pravidla. Pravidlo je tvorené primárne zo závislostí, značiek slovných druhov a typov názvoslovných entít. Závislostí majú viacero využití pri tvorbe poznámok, hlavné z nich sú jednoznačná identifikácia informácie a pravidlo závisle od štruktúry vety. Z toho vyplýva, že pravidlo je aplikovateľné na viacero viet s rovnakou štruktúrou. Štruktúru si vieme reprezentovať stromom alebo priamo vo vete. Informácie o spracovaných článkoch, vetách a vytvorených poznámkach a pravidlách sa ukladajú do textovej databázy s dôkladne navrhnutým modelom.

# 5 Systém na tvorbu poznámok

Pri implementácií systému na tvorbu poznámok vyberáme konkrétnu textovú databázu a hlavný nástroj na podporu spracovania prirodzeného jazyka. Riešime fundamentálne princípy používania pravidiel na tvorbu poznámok, ako vyhľadanie, aplikovanie, vytvorenie a úprava pravidla.

## 5.1 Použité technológie

Systém je implementovaný v programovacom jazyku C# a na grafické rozhranie je použitý framework WinForms. Postavený je nad nástrojom StanfordNLP (bližšie opísaný v časti 2.2.2 StanfordNLP), ktorý umožňuje jednoduchým spôsobom získať esenciálne informácie pre náš systém ohľadne textov, ako závislostí slov vo vetách, značky slovných druhov, názvoslovné entity, indexy a lemy slov. StanfordNLP je pomocou knižnice IKVM sprístupnený z jazyka Java do balíka Stanford.NLP.CoreNLP v jazyku C#. V systéme je použitá verzia 3.5.2 tohto nástroja aj knižnice Stanford.NLP.CoreNLP, spolu so zdrojovými kódmi modelov z originálneho nástroja. Databázová vrstva systému je implementovaná pomocou textovej databázy MongoDB, ktorá uľahčuje narábanie s textom a využíva vlastný dopytovací jazyk [8]. Na komunikáciu medzi databázou a systémom sa používa knižnica MongoDB.Driver a na prácu s databázovými štruktúrami, ako sú dokumenty, je potrebná knižnica MongoDB.Bson. Knižnica HtmlAgilityPack je využívaná na spracovávanie HTML kódu pri získavaní vstupných článkov priamo z webovej stránky.

## 5.2 Manažment dát

Pre vytvorenie poznámky z vety potrebujeme 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 dostato-

čnej 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 35. 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.

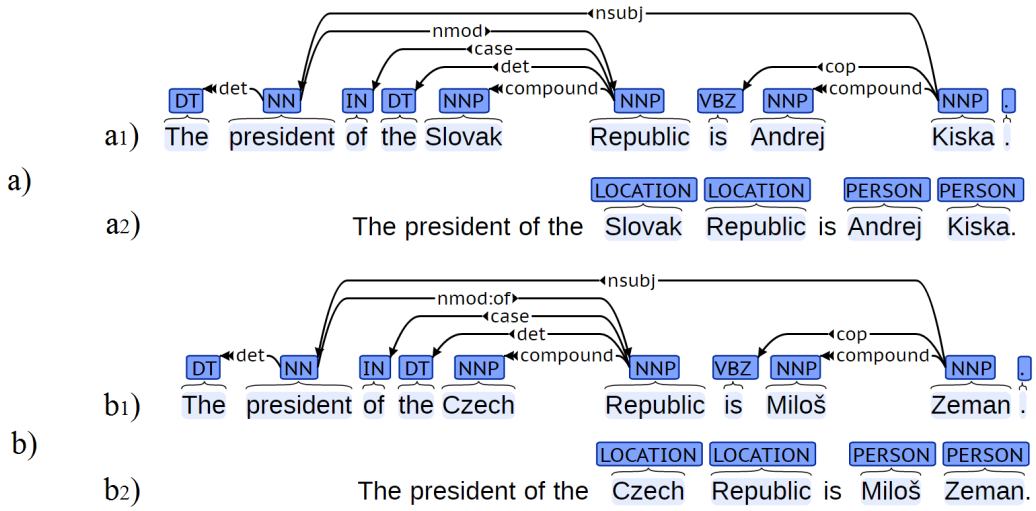
Posledná časť zhody, hodnotová časť zhoda, reprezentuje úplnú zhodu dvoch viet. Veta má hodnotovú časť zhody len s totožnou vetou.

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 18 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, závislosť) then
7:                     do zhody na type porovnania pripočítaj ohodnotenie kroku
       return zhoda
```




Obr. 18: *Príklad určenia zhody viet*

Predpokladajme situáciu z obrázka 18 Príklad určenia zhody viet. V databáze máme uloženú spracovanú vetu *a* aj s pravidlom pre túto vetu. Spracovávame vetu *b*. Na časti *a1* obrázka 18 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 35) 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 37). 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.

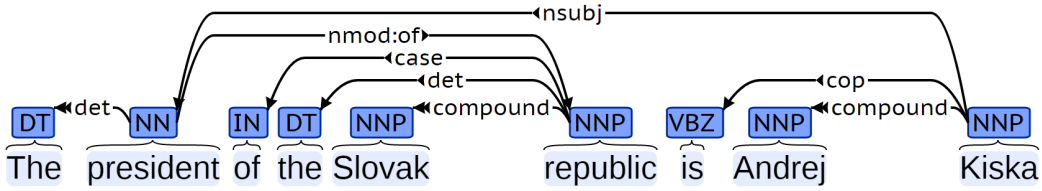
**Algoritmus 2** Aplikovanie pravidla

```
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
```

Pre vetu *„The president of the Slovak republic is Andrej Kiska."* nám nástroj Stanford CoreNLP poskytne závislostí vyobrazené na obrázku 19 Závislostí jednoduchej vety. Ak na túto vetu aplikujeme pravidlo so štruktúrou v tvare zobrazenej na obrázku 20 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 19

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. 19: *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. 20: *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 ohod-

notený 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 Možnosti rozšírenia systému

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

Databázová vrstva nášho systému je implementovaná pomocou textovej databázy *MongoDB*. Ako hlavný nástroj na podporu spracovania prirodzeného jazyka využíva nástroj *StanfordNLP*, pre jednoduché získanie esenciálnych informácií o textoch. V procese vyhľadania pravidla je potrebné určiť zhodu štruktúry spracovávanej vety so štruktúrami viet v databáze, z dôvodu vyhľadania najvyhovujúcejšieho pravidla. Určenie zhody sa delí na tri časti, a to štrukturálna, obsahová

a hodnotová. Pri aplikovaní pravidla sa počíta zhoda závislostí, kvôli udržaniu dostatočnej všeobecnosti pravidla. Po vytvorení poznámky je používateľ schopný si danú poznámku upraviť podľa vlastných predstáv a systém sa naučí nové, alebo upraví existujúce pravidlo, ako spracovávať vetu, z ktorej poznámka vznikla. Systém má priestor na rozšírenie a to najmä v predspracovaní textu v zmysle eliminácie nepodstatných viet.

# 6 Experimenty

...

# 7 Zhodnotenie

...

# Zoznam použitej literatúry


[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 Používateľská príručka

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

## A.1 Inštalácia

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.

## A.2 Spustenie aplikácie

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

# B 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) |

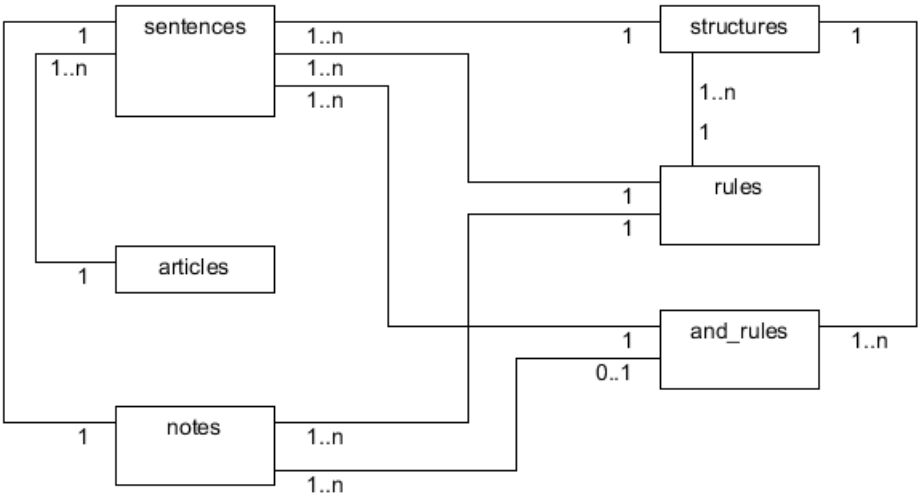
Obr. 21: *Zoznam závislostí*

# C 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. |
| rule_ref_id | Odkaz do kolekcie *Rules.* |
| and_rule_ref_id | Odkaz do kolekcie *And Rules.* |
| article_ref_id | Odkaz do kolekcie *Articles.* |
| structure_ref_id | Odkaz do kolekcie *Structures.* |
| note_ref_id | Odkaz do kolekcie *Notes.* |
| text | Spracovaný text. |
| structure_data | Zoznam dát o štruktúre. |
| relation_name | Názov vzťahu závislosti. |
| dependencies | Zoznam závislostí. |
| governor | Nadradený token. |
| dependent | Podradený token. |
| position | Pozícia závislosti v štruktúre. |
| comparison_type | Interná hodnota – typ porovania závislostí. |
| token_type | Interná hodnota – typ zodpovedajúceho tokenu informácie. |
| POS | Značka slového druhu. |
| index | Pozícia tokenu vo vete. |
| ner | Typ názvoslovnej entity. |
| lemma | Lema tvar slova. |
| sentence_terminators | Pozície ukončení viet v poznámke. |
| set_position | Pozícia množiny viacnásobnej poznámky. |

Obr. 22: *Legenda diagramov kolekcií*




Obr. 23: *Databázový model*

| Pole | Typ |
|---|---|
| _id | ObjectId |
| text | String |
| created_at | DateTime |
| updated_at | DateTime |

Obr. 24: *Model kolekcie articles*

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

Obr. 25: *Model kolekcie notes*

| 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. 26: *Model kolekcie sentences*

<table>
<tr><th>Pole</th><th>Typ</th></tr>
<tr><td>_id</td><td>ObjectId</td></tr>
<tr><td>structure_data</td><td>Array of Documents
<table>
<tr><th>Pole</th><th>Typ</th></tr>
<tr><td>relation_name</td><td>String</td></tr>
<tr><td>dependencies</td><td>Array of Documents
<table>
<tr><th>Pole</th><th>Typ</th></tr>
<tr><td>governor</td><td>Document
<table>
<tr><th>Pole</th><th>Typ</th></tr>
<tr><td>POS</td><td>String</td></tr>
<tr><td>index</td><td>Integer</td></tr>
<tr><td>ner</td><td>String</td></tr>
<tr><td>lemma</td><td>String</td></tr>
</table></td></tr>
<tr><td>dependent</td><td>Document
<table>
<tr><th>Pole</th><th>Typ</th></tr>
<tr><td>POS</td><td>String</td></tr>
<tr><td>index</td><td>Integer</td></tr>
<tr><td>ner</td><td>String</td></tr>
<tr><td>lemma</td><td>String</td></tr>
</table></td></tr>
<tr><td>position</td><td>Integer</td></tr>
<tr><td>comparison_type</td><td>Integer</td></tr>
<tr><td>token_type</td><td>Integer</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>created_at</td><td>DateTime</td></tr>
<tr><td>updated_at</td><td>DateTime</td></tr>
</table>

Obr. 27: *Model kolekcie structures*

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

Obr. 28: *Model kolekcie rules*

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

Obr. 29: *Model kolekcie and rules*

# D Ukážka článkov použitých v experimente

*Slovensko* - „Slovakia (Slovak: Slovensko) (Official name The Slovak Republic, Slovenská republika) is a country with no access to the ocean in Central Europe. 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. Its capital city is Bratislava. Other main cities are Košice, Banská Bystrica, Žilina, Trenčín, Nitra, Prešov, and Trnava. Slovakia is a member of the European Union."

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

*Maďarsko* - „Hungary is a country in Central Europe. Its capital city is Budapest. Hungary is slightly bigger than its western neighbour Austria and has about 10 million inhabitants. Other countries that border Hungary are Slovakia, Ukraine, Romania, Serbia, Croatia and Slovenia. Hungary's official language is the Hungarian language. It has been a member of the European Union (EU) since 2004. In Hungarian the country is called Magyarország (Hungary) or Magyar Köztársaság (Hungarian Republic). This is named after the Magyar tribes who came to Hungary in the late 9th century."

*Poľsko* - „Poland is a country in Eastern Europe. It is next to Germany to the west (along Oder and Lusatian Neisse), the Czech Republic and Slovakia to the south, Ukraine and Belarus to the east, and the Baltic Sea, Lithuania, and Russia to the north. The total land area of Poland is about 312,679 km2 (120,728 mi2). This

makes Poland the 77th largest country in the world with over 38.5 million people. Most Polish people live in large cities, including the capital, Warsaw (Polish: Warszawa), Lódź, Cracow (Polish: Kraków), the second capital of Poland (first was Gniezno), Szczecin, Gdańsk, Wroc law and Poznań. The word ”Poland”was written officially for the first time in 966. In 1569, Poland formed a strong union with Lithuania called the Polish-Lithuanian Commonwealth. At some point in its history, it was the largest state in Europe and became very influential. Much of the territory that now makes up Central European states used to belong to the Commonwealth. Eventually, after entering a somewhat sudden yet steady decline, the Commonwealth collapsed in 1795. Poland regained its independence in 1918 after World War I. In 1921, Poland defeated Soviet Russia in the Polish-Soviet War that started in 1919.”

*Nemecko* - „The Federal Republic of Germany, also called Germany (German: Bundesrepublik Deutschland or just Deutschland), is a country in Central Europe. The country’s full name is sometimes shortened to the FRG (or the BRD, in German). To the north of Germany are the North Sea, the Baltic Sea, and the country of Denmark. To the east of Germany are the countries of Poland and the Czech Republic. To the south of Germany are the countries of Austria and Switzerland. To the west of Germany are the countries of France, Luxembourg, Belgium, and the Netherlands. The total area of Germany is 137,847 square miles. The large majority of Germany has warm summers and cool or cold winters. In June 2013, Germany had a population of 80.6 million people. After the United States, Germany is the second most popular country for migration in the world. Before it was called Germany, it was called Germania. In the years A.D. 900 until 1806, Germany was part of the Holy Roman Empire. From 1949 to 1990, Germany was made up of two countries called the Federal Republic of Germany (inf. West Germany) and the German Democratic Republic (inf. East Germany). During this time, the capital city of Berlin was divided into a west and an east part. On 13 August 1961, East Germany started building the Berlin Wall between the two parts of Berlin. West Germany was one of the countries that started the European Union.”

*Taliansko* - „Italy is a country in Europe and a member of the European Union.

Its official name is Repubblica Italiana. Italy is a democratic republic and is a founding member of the European Union. Italy is also a member of the G8, as it has the 8th largest Gross Domestic Product in the world. Its President is Sergio Mattarella and its Prime Minister is Matteo Renzi. Before 1861, it was made up of smaller kingdoms and city-states."

*Francúzsko* - „France (French: France), officially the French Republic (French: République française), is a country in Western Europe. Its capital city is Paris. It is a member of the European Union. It is known for its culture, its many monuments and structures, and places such as the Louvre, the Eiffel Tower, the Arc de Triomphe, Giverny, Mont Saint Michel, Versailles, and Notre Dame de Paris. France is divided into 22 régions that are further subdivided départements. The country has been one of the great powers since the end of the 17th century. In the 18th and 19th centuries, it built a big colonial empire across West Africa and Southeast Asia. Nowadays, this does not exist. It is the most visited country in the world. About 82 million foreign tourists visit it every year. France is a founding member of the European Union. It has the largest land area of any member. France is also a founding member of the United Nations, and a member of the G8 and NATO. It is one of the five permanent members of the United Nations Security Council. It has nuclear weapons, including active warheads, and also has nuclear power plants. Some well-known cities in France include Paris, Lyon, Marseille, Bordeaux, Lille, Toulouse, Nice, Strasbourg, Rennes and Nantes."

*Spojené kráľovstvo* - „The United Kingdom of Great Britain and Northern Ireland, called the United Kingdom, GB or UK, is a sovereign state in Western Europe. It unites England, Northern Ireland, Scotland and Wales as one Kingdom.[10] It is a member of the European Union, the United Nations, the Commonwealth, NATO and the G8. It has the sixth largest economy in the world. Around 65 million people live in the UK. Most people in the UK speak English. There are five native languages other than English. They are Welsh in Wales, Gaelic and Scots in Scotland and Northern Ireland, Irish in Northern Ireland, and Cornish in Cornwall. Between the 17th and mid 20th-centuries Britain was an important world power. It became a colonial empire that controlled large areas of Africa, Asia, North America and

Oceania. Today this empire does not exist, although Britain keeps links with most countries of its former empire. Some well-known cities in the UK are London, Edinburgh, Cardiff, Belfast, Manchester, Bristol, Liverpool, Birmingham, York and Glasgow."

*Španielsko* - „Spain is a country in Southern Europe. It is in the Iberian Peninsula near Portugal and Gibraltar. France and the country of Andorra are on its northeast side, where the Pyrenees mountains are. The people of Spain are called Spaniards. Most people there speak Spanish (in Spanish, "Castellano", from Castilla, or Ëspañol") but there are other languages in different parts of the country. They are Catalan, Basque, and Galician, Leonese, Aragonese, Aranese Occitan and even Portuguese. The religion of most of the people in Spain is Roman Catholic. Since 1975, Spain has had a king who only does what the constitution allows him to. For example, the king can declare a war, but only if the Government asks him to do so. The parliament is called Las Cortes Generales, and has two bodies: Ël Congreso"(The Congress) and Ël Senado"(The Senate) and it is chosen by the Spanish people by voting. This kind of government is called a constitutional monarchy. The King of Spain is Felipe VI. The Prime minister is Mariano Rajoy. The government and the king's palace are in Madrid, the capital of Spain. Spain has more than five hundred thousand square kilometres of land. It is smaller than France, but it is bigger than Sweden or Germany. Almost fifty million people live in Spain. Spain has 17 parts called autonomous communities (this means that they can decide upon some affairs themselves). Each part has its own government."

*Ukrajina* - „Ukraine is a country in Eastern Europe. Russia is to the north-east of Ukraine, Belarus is to the Northwest, Poland and Slovakia are to the West, Hungary, Romania, Moldova and self-proclaimed Transnistria are to the South West and the Black Sea is to the Southwest. Ukraine is a republic. The capital of Ukraine is Kiev. It was a part of the Soviet Union from 1922 until 1991."

*Rakúsko* - „Austria (German: Österreich; officially called Republic of Austria), is a country in Central Europe. Around Austria there are Germany, Czech Republic, Slovakia, Hungary, Slovenia, Italy, Switzerland, and Liechtenstein. Currently, the

chancellor is Werner Faymann. Austria has been a member-state of the EU since 1995. The people in Austria speak German, a few also Hungarian, Slovenian and Croatian. The capital of Austria is Vienna (Wien). Austria is more than a thousand years old. Its history can be followed to the ninth century. At that time the first people moved to the land now known as Austria. The name Östarrichiïs first written in an official document from 996. Since then this word has developed into the Modern German word Österreich, which literally means Ëast Kingdom.Äustria is a democratic state and has nine federal states (German: 'Bundesländer'): Vorarlberg, the Tyrol, Salzburg, Carinthia, Styria, Upper Austria, Lower Austria, Vienna and Burgenland. It is a neutral state, that means it does not take part in wars with other countries. Austria has been in the United Nations since 1955 and in the European Union since 1995."

*Švédsko* - „Sweden (Swedish: Sverige) is a Nordic country in the part of Europe called Scandinavia. Its neighbors are Finland and Norway. Sweden is also connected to Denmark in the south by a bridge. It is a developed country. It is famous for its welfare state. People who live in Sweden are called Swedes. Sweden's capital city is Stockholm. Sweden is a constitutional monarchy because it has a king, Carl XVI Gustaf, but he does not have any real power. Sweden is a parliamentary state meaning that the government is elected by the parliament which is appointed by the people. The country is democratically ruled by a government headed by an elected prime minister. Stefan Löfven was elected Prime Minister in September 2014. He took office in October 2014. The population of Sweden is almost 9.5 million people. Sweden has an official majority language, (called svenska in Swedish). Sweden has five official minority languages, Finnish, Yiddish, Sami, Meänkieli and Romani. Sweden became a member of the European Union in 1995. It is not a member of the Eurozone. Sweden has not begun to use the euro as currency. This is because the people have voted against this. The currency remains the Swedish krona (Swedish crown). Sweden has 25 provinces (landskap). They are found in three different regions: Norrland in the North, Svealand, the central region, and Götaland in the South."

# E IIT.SRC článok

# Interactive System for Creation of Notes

Martin NEMČEK*

*Slovak University of Technology in Bratislava*
*Faculty of Informatics and Information Technologies*
*Ilkovičova 2, 842 16 Bratislava, Slovakia*
xnemcekm@stuba.sk

**Abstract.** 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 article 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 will be an interactive system for creating notes based on learned rules from user.

## 1 Introduction

Computers are not able to understand information in natural language. In our proposed system the notes are created from sentences by extracting relevant information from them. We use syntactic analysis of sentences and extract relations and dependencies between words from these sentences. The final result of our proposed method are personalized notes. The user will be able to modify the automatically created notes. The system will then learn new rules for sentence to note transformation from these changes and takes them into account for the next time.

## 2 Our proposed system

A rule consists mainly from two parts - *list of data of original sentence* and *list of data of note*. Each entry in *list of data of original sentence* and *list of data of note* contains these parts: relation name and list of grouped dependencies with the same relation name. Each dependency contains a governor token, a dependent token and its position considering all dependencies. The governor and dependent token consists of Part-Of-Speech (POS) tag and index of word in sentence to which is the token connected. Index of the token is bounded with a position of its word in sentence.

Dependencies from the second list are applied to sentence to create a new note. The rule may order to create a compound note from a sentence. The compound note is composited of some simple sentences. The positions in sentence on which the note should be split into smaller sentences are

* Bachelor study programme in field: Informatics
Supervisor: Miroslav Blšták, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

kept within the rule.

When processing a sentence an applicable rule has to be looked up in database before creation of the note. Dependencies of rule and dependencies of the sentence being processed has to correspond to each other. Evaluation is based upon two conditions. The sentence that is being processed has to have the exact amount of entries in list of data of original sentence while these entries contain exactly the same relation names as the rule's relation names.

The applicable rule is found if these two conditions are met. However, the conditions can cause a situation that more than one rule is found. In this case we have to calculate the match probability of this sentence and the original sentence obtained from the rule. The rule with the highest probability of the match is applied.

Calculating the match consists of several steps. First, the POS tags match of governor and dependent tokens is calculated separately. Indices of governor and dependent tokens are calculated also separately. These first steps determine if the sentence contains arbitrary dependency with same value of POS tag or index. In followed step is determined a half-match of dependencies. Half-match of dependency is match of POS tag and index at the same time at governor or dependent token of dependency. We calculate matches of POS tags and index of governor or dependent token for every dependency. Finally, in the last step we calculate the number of absolute-matched dependencies. Absolute-match dependencies is the total match of POS tags and indices in governor and dependent tokens. Every step has assigned a rating. If a condition in the step is evaluated as true, the rating of the step is added to the final result. The final result is a percentage value of the match. The rating is based on importance of the step in calculating a precise match, while depending on the number of steps and dependencies, so the final result cannot exceed a limit of 100%. A pseudo code for an algorithm calculating the match is outlined in Algorithm 1 and specific example is shown on Figure 1.

**Algorithm 1** Calculating match

```
1:  procedure CALCULATEMATCH(sentence, originalDependencies)
2:      oneCompareTypeRating ← calculate percentage rating of one comparison
3:      for all originalDependencies do
4:          if count(sentence, dependency) = count(originalDependencies, dependency) then
5:              match ← match + oneCompareTypeRating
6:          counter ← counter + count(originalDependencies, dependency)
7:      oneCompareTypeRating ← oneCompareType/counter
8:      for all originalDependency do
9:          for all dependency do
10:             for all comparison do
11:                 if applyComparison(sentence, comparison, dependency) then
12:                     match ← match + oneCompareTypeRating
        return match
```

If rule look up does not find any applicable rule, it means that the system have not processed the same or similar sentence yet. A manual rules of parser are used in this case. The output of the parser is a note. A new rule is created based on the note. Dependencies of original sentences are taken and used to create a *list of data of original sentence*. This list is then assigned to the rule. Dependencies of note are used to create a *list of data of note* which is then also assigned to the rule. The sentence ends are determined depending on how many sentences the note contains. POS tags and indices of tokens are stated by the corresponding words of the original sentence and the newly created note.

By the principle of rule look up, the sentence being processed has to contain dependencies from the *list of data of original sentence* and also dependencies from the *list of data of note*.

The process of applying a rule has several steps. For each dependency in the list of data of original sentence, the respective dependency is looked up in sentence that is being processed. The word corresponding with dependent token from the looked up dependency is taken and added to the note on its index position. In case of dependency relation *nominal subject* the word corresponding with governor is also added. After processing all dependencies the last minor changes are done such as capitalization of the first letter of the note, splitting note into more sentences if rule defined so. Algorithm 2 shows pseudo code of the process of applying rule on sentence.

**Algorithm 2** Applying rule

```
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
```

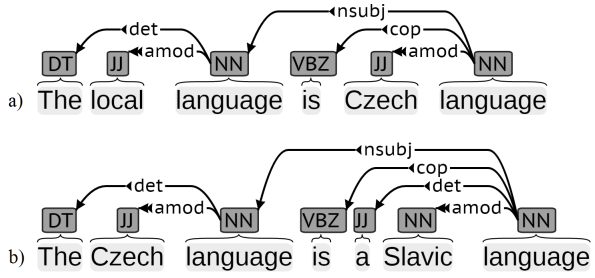



*Figure 1. Example sentences*

Lets consider example from Figure 1. We have rules for two sentences and we are processing the first one. In this situation, there are at least two rules which are applicable for the sentence *a*. Assume that we are calculating match with the sentence *b*. We iterate over all dependencies of processed sentence *a*. The first dependency is *det* with the governor token NN (noun) and index 3 and the dependent token DT (determiner) and index 1. First, we find out, if the sentence *b* contains any dependency with tokens NN or DT and index equals to 1 or 3. This is the separate calculation of POS tags and indices. Then, we try to find in the sentence *b* any dependency, which has dependent or governor token tag of type NN and index equal to 3 or tag of type DT and index equal to 1. This is only the half-match step. As the last step, we check if sentence b contains dependency, where the governor token is NN and index is equal to 3 and the dependent token is DT and its index is 1. If any of these step were matched, the rating of that particular step is added to the final result and iteration continues with following dependency until all dependencies were iterated over.

*Acknowledgement:* This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG 1/0646/15.