Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 812 19 Bratislava

Štatistický preklad textu

Dokumentácia k inžinierskemu dielu

Tím č. 4 – aDictIT

Bc. Róbert Horváth
Bc. Peter Jurčík
Bc. Peter Macko
Bc. Vladimír Ruman
Bc. Peter Sládeček
Bc Maroš Ubreži
Bc. Matúš Vacula

Vedúci: Ing. Dušan Zeleník
Akademicky rok:2010/2011
Predmet: Tímový projekt

Obsah

1 Účel dokumentu

Obsahom predkladaného dokumentu je dokumentácia k vývoju softvérového systému. Úlohou vyvíjaného systému je prekladať používateľom zadaný text z jedného jazyka do iného jazyka a späť. Text sa prekladá po vetách alebo skupinách slov. Spôsob prekladania je založený na štatistickej podobnosti výskytu prekladanej časti v korpuse. Tento softvérový systém je výsledkom študentského tímového projektu z predmetu Tímový projekt. Vedúcim a zároveň zadávateľom témy je Ing. Dušan Zeleník.

1.1 Forma a rozsah dokumentu

Projekt je vyvíjaný pomocou agilnej metodiky SCRUM, ktorá má viaceré špecifiká. Vývoj v nej prebieha v diskrétne oddelených častiach nazývaných šprinty, ktoré trvajú spravidla dva týždne. Táto skutočnosť sa odráža aj na dokumentácii, ktorej obsahom je podrobný opis vyvinutej práce v prvých dvoch šprintoch. Každý šprint pozostáva z používateľských príbehov (User story), ktoré majú jasne formulované rozdelenie na analýzu, návrh, implementáciu a testovanie. V rámci príbehov sú identifikované úlohy, ktoré v dokumente štruktúrujú jednotlivé časti príbehov.

Na konci dokumentu sa vo forme príloh nachádzajú výstupy analýzy prekladu viet a odstavcov.

1.2 Použité technológie

Pre prácu na projekte sme sa rozhodli pre využitie nasledujúcich technológií:

- Java
- MySQL
- PHP, HTML5
- Tomcat
- Apache
- Systém pre verziovanie: Git
- Systém pre správu úloh: Redmine
- IDE: Eclipse, NetBeans

1.3 Slovník pojmov

Korpus – súhrn všetkých skúmaných textov, komunikátov daného jazyka

Paralelný korpus – korpus obsahujúci totožné texty v rôznych jazykoch

Webová služba – je to služba prístupná na webe podľa zadanej požiadavky, prostredníctvom XML súboru vracia odpoveď s výstupom v XML formáte

1.4 Zoznam skratiek

WS – webová služba

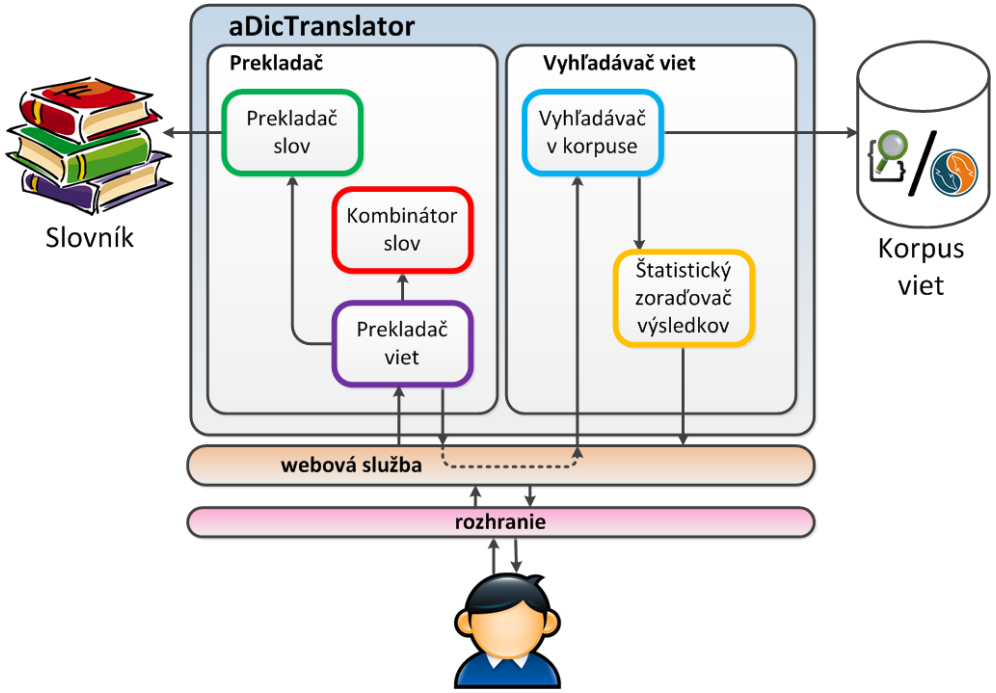
SQL - Structured Query Language, je počítačový jazyk na manipuláciu (DML) (výber, vkladanie, úpravu a mazanie) a definíciu dát (DDL).

2 Základná štruktúra systému

Náš systém na preklad textu sa skladá z niekoľkých vrstiev. Tieto vrstvy oddeľujú jednotlivé funkčné celky a vytvárajú celkovú funkcionalitu produktu ako je zobrazené na obrázku Obr. 1 Štruktúra prekladača.

Týmito vrstvami sú:

1. Rozhranie prekladača
2. Webová služba
3. Prekladač



Obr. 1 Štruktúra prekladača

O samotnej štruktúre nášho prekladača hovorí aj kapitola 7.1 Nová štruktúra prekladača.

2.1 Rozhranie prekladača

Rozhranie prekladača je stránka v jazyku PHP, ktorá zabezpečuje komunikáciu medzi koncovým používateľom a webovou službou. Táto vrstva systému môže byť nahradená ľubovoľným iným rozhraním prekladača 3tej strany.

Technické detaily o rozhraní sa spomínajú v tejto dokumentácii v nasledujúcich kapitolách: Návrh používateľského rozhrania, 9.5 Nové webové rozhranie prekladača.

2.2 Webová služba

Aby sme poskytli svoje služby používateľom vytvorili sme webovú službu. Na ňu sa pripája naše rozhranie, ale dokážu ju využívať aj tretie strany. Na implementáciu sme použili typ služby REST. Samotná služba je naprogramovaná v prostredí Java. Služba beží na aplikačnom servery Tomcat vo verzií 7.

Samotná implementácia služby je spomenutá v kapitole 5.1 Služba typu REST, pričom následné úpravy sa nachádzajú v časti 8.4 Optimalizácia služby typu REST a ďalšie pomocné možnosti služby sú v časti 8.2 Pomocné funkcie webovej služby.

2.3 Prekladač

Jadrom nášho systému na preklad viet je samotný prekladač. Tak, ako webová služba aj on je naprogramovaný v jazyku Java. Okrem toho využíva zdroje v podobe MySQL databázy slovníka na preklad a ElasticSearch vyhľadávača na indexáciu viet.

Samotný prekladač sa skladá z dvoch základných častí:

- Translatror
- SentenceFinder

Obidve tieto časti využívajú návrhový vzor strategy, ako je spomenuté v kapitole 8.3 Vytvorenie Stratégií. Umožňuje to jednoduché vytváranie a prácu s viacerými verziami prekladača.

2.3.1 Translator

Translator je časť riešenia, ktorá získava preklady jednotlivých slov zo slovníka a následne ich spája do možných viet. Týmto vzniká veľká množina viet, ktoré sa dostávajú do SentenceFindera. Na preklad sme získali a vytvorili databázu prekladov slov, ktorá je opísaná v kapitole 3.2.2 Návrh slovníka. Spôsob kombinovania slov do výsledných viet je opísaný v kapitole 4.1 Preklad vety. Slovník sme okrem toho prispôsobili tak, aby vedel vyhľadávať aj preklady slovných spojení, čo je opísané v kapitole 7.3 Skupiny slov v korpuse.

2.3.2 SentenceFinder

SentenceFinder je skutočným jadrom celej metódy. Táto časť dostáva možné vety a snaží sa ich vyhľadať v databáze možných viet, pričom počas tímového projektu sme vytvorili niekoľko prístupov, ako to spraviť.

Základom našej metódy bola rozsiahla analýza prekladu vety v kapitole 4.1 Preklad vety. Tu sa vyjadrujeme aj k základnému korpusu viet ktoré na preklad používame z databázy SME článkov. Ďalším zdrojom viet sa stali titulky z filmov, ktorých získanie je popísané v časti 5.3 Titulky.

V tejto časti sme sa zaoberali aj možnosťami na rýchlejšie vyhľadávanie viet v korpuse čo sme rozobrali v časti 5.4 Zrýchlenie vyhľadávania. Výstupom tejto kapitoly bolo použitie nástroja

ElasticSearch na hľadanie v množine viet. Neskôr sme museli vety čistiť, čo je zachytené v kapitole 10.3 Čistenie viet.

Pre možnosť rýchlejšieho testovania zmien parametrov sme vytvorili konfiguračný súbor nášho riešenia. Táto činnosť je zachytená v časti 8.1 Konfigurovateľnosť prekladača.

Prekladač sme prispôsobili tak, aby dokázal vo vetách v korpuse nahrádzať číslovku za číslovku, ktorú zadal používateľ. Podobný postup bol zvolený aj pre menné entity. Tieto postupy sú opísané v kapitole 6.2 Preklad čísel a vlastných mien, v ktorej sme sa snažili využiť existujúce riešenia. V častiach 7.2 Extraktor faktov z korpusu viet a 10.4 Nahrádzanie čísloviek sme opísali konkrétny spôsob, ktorý sme na nahrádzanie použili.

Vyhľadávač prekladov je zhotovený vo viacerých verziách. Ako progresívna metóda bola rozobraná verzia n-gramov v časti 9.1 Vyhľadávanie cez n-gramy. Metódu sme aj ďalej upravovali a snažili sme sa zapojiť term frequency do procesu hľadania najlepšej zhody, čo je opísné v kapitole 10.1 Pridanie vplyvu Term frequency na preklad cez N-gramy.

Ďalšou progresívnou metódu bolo vyhľadávanie cez synonymá v časti 9.2 Vyhľadávanie pomocou synoným a 9.3 Reindexácia údajov z indexu sentences do indexov synoným, ktoré sa však neosvedčilo z dôvodu nevhodného chovania databázy ElasticSearch.

Ďalšou metódou bolo upravovanie a zohľadňovanie poradia viet zo slovníka pri hodnotení viet. To je opísané v kapitole 10.5 Verzia poradie slov v slovníku.

2.4 Ďalšie podporne nástroje

Na to, aby sme vedeli ako sa naša metóda zlepšuje sme využili možnosti testovania úspešnosti prekladu. To je detailne opísané v časti 5.2 Úspešnosť prekladov. Následne sme vytvorili aj testovacie prostredie pre rýchle overenie toho, či náš prekladač zvláda vety ktoré vedel preložiť v predchádzajúcej verzii. Spôsob vytvorenia tohto testovacieho prostredia je opísaný v kapitole 6.1 Automatický test, ktorý bol neskôr rozšírený o podporu viacerých verzií prekladača (7.4 Automatický tester).

Aby sme dokázali pracovať súčasne s databázou MySQL a ElasticSearch, potrebovali sme aj nástroj, ktorý by vedel upravovať dáta v oboch databázach naraz. Ten je rozobraný v časti 7.5 Návrh nástroja pre automatické pridávanie zmien do ElasticSearch a MySQL.

Počas testovania prekladov bolo potrebné sledovať logy. Preto sme si vytvorili upravený formát zápisu cez knižnicu Log4J, ktorý je opísaný v časti 9.4 Logovanie v prekladači.

3 Prvý šprint – čajíček

V súvislosti s názvom nášho tímu sme sa jednotlivé šprinty rozhodli pomenovávať návykovými látkami. V prvom šprinte je to káva, ako jedna z najneškodnejších návykových látok.V tomto šprinte sme riešili nasledujúce úlohy:

1. Analýza existujúcich prekladačov
2. Navrhnutie používateľského prostredia
3. Výber slovníka pre preklad
4. Vytvorenie webovej služby

Výsledkom prvého šprintu je teda funkčné prostredie pre prácu s prekladačom, takisto slovník, ktorý obsahuje dostatočné množstvo slov pre preklad. V konečnej fázy sme spustili službu a prepojili ju s grafickým rozhraním tak, aby si náš zákazník dokázal prekladať slová.

3.1 Návrh používateľského rozhrania

Ako používateľ chcem *pracovať* s *používateľským rozhraním*, pomocou ktorého budem *prekladať text*.

3.1.1 Analýza existujúcich rozhraní

V kapitole sa venujeme rozboru rozhraní populárnych webových aplikácií a to konkrétne Google Translate a Yahoo! Babel Fish.

Google Translate

Google Translate poskytuje používateľom prehľadné a elegantné rozhranie, ktorého použitie je veľmi intuitívne. Používateľ má možnosť nastavenia zdrojového a cieľového jazyka prekladu pomocou dvoch tlačidiel, ktoré sa kliknutím rozvinú do zoznamov ponúkajúcich kompletnú jazykovú ponuku.

Do výrazného textového poľa v ľavej časti obrazovky máme možnosť vpisovať text, ktorý si želáme prekladať. Preklad je ponúknutý v podobnom elemente na opačnej strane obrazovky, avšak bez možnosti editácie používateľom (Obr. 2).

Celé rozhranie sa výborne prispôsobuje používateľom nadstavenému rozlíšeniu a veľkosti jeho obrazovky. Základnou myšlienkou je vložiť textové pole pre vpisovanie do elementu, ktorého CSS atribút width (šírka) je zadaný pomerovo v percentách. Následne pomocou JavaScriptu je získaná dĺžka tohto elementu. Od tejto hodnoty je odčítaná určitá hodnota. Následne je takto modifikovaná hodnota pripísaná CSS elementu width (šírka) textového poľa. Tým sa zabezpečí efekt rovnakého odstupu písaného textu od okraja elementu, bez ohľadu na parametre používateľovej obrazovky. Rovnako sa týmto predchádza aj odlišnostiam pri interpretácii vlastného box modelu jednotlivými webovými prehliadačmi. Tento postup je použitý aj pri nadstavení okna pre zobrazovanie výsledkov prekladaného textu.

Obr. 2 – Rozhranie Google Translate. Bledomodrá časť ukazuje textové pole vložené do elementu s pomerovo nadstavenou šírkou

Yahoo! Babel Fish

Zanalyzujme si ešte webové rozhranie prekladača od spoločnosti Yahoo!. Pri prvotnom zobrazení stránky sa používateľovi zobrazuje iba textové pole, do ktorého vkladá text, ktorý chce prekladať. Následný výsledok z procesu prekladania sa zobrazuje v okne, ktoré sa zobrazí nad oknom textového poľa so zadávaným textom (Obr. 2).

Používateľ má možnosť nastavenia jazykov prekladu z natívneho HTML zoznamu možností, umiestneného priamo pod textovým poľom so zadávaným textom.

Na rozdiel od Google Translate nie je rozhranie prekladača od spoločnosti Yahoo! natoľko modifikovateľné. Textové pole pre vpisovanie prekladaného textu je vložené v kontajneri s pevnou šírkou. Optimalizácia pre rozličné webové prehliadače je samozrejmosť.

Obr. 3 – Rozhranie prekladača Yahoo! Babel Fish od spoločnosti Yahoo!

Záver

Prekladač od Yahoo! ponúka svojim používateľom klasické rozhranie s využitím dostupných HTML elementov v kombinácii s CSS. Google Translate pri budovaní rozhrania navyše využíva aj možnosti technológie na strane klienta a tým ponúka svojim užívateľom veľmi elegantné, moderné a na používanie intuitívne rozhranie. Rovnako tak aj celá aplikácia od Googlu pôsobí veľmi interaktívne a poskytuje používateľovi väčší komfort pri práci.

Výber implementačného prostredia

Prvou časťou nášho implementačného prostredia sa stal operačný systém. V tejto oblasti sme sa priklonili k operačnému systému Linux Fedora. Hlavným dôvodom bola jednoduchá práca, možnosť jednoduchého inštalovania ďalších programov a pravidelné aktualizácie systému.

V oblasti programovacieho jazyka sme zvolili programovací jazyk Java. Tento jazyk nám ponúka hlavne možnosti jednoduchého programovania a možnosti práce s webovými technológiami. Okrem toho budeme využívať aj programovací jazyk PHP pre tvorbu webového rozhrania. Pri práci s webovým rozhraním okrem toho využijeme aj JavaScript a framework JQuery.

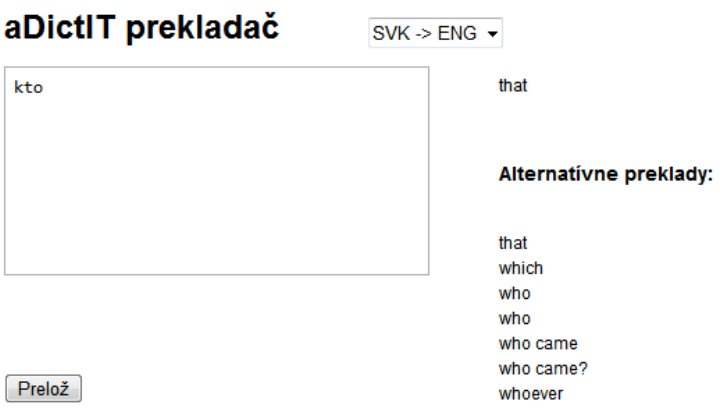
Na ukladanie dát využijeme databázový systém MySQL, ktorá nám zabezpečí rýchly a jednoduchý prístup k dátam. Oproti ostatným systémom pre nás ponúka najlepšie možnosti rozšírení.

3.1.2 Návrh

Návrh grafického rozhrania

Na základe vykonanej analýzy existujúcich rozhraní sme sa rozhodli navrhnúť elegantné a pre používateľa intuitívne rozhranie webovej aplikácie. Rozmery jednotlivých elementov by sa mali prispôsobiť veľkosti používateľovej obrazovky ako aj nadstavenému rozlíšeniu.

Zrealizované rozhranie je viditeľné na Obr. 3. Používateľ má možnosť si vybrať zadávaný a cieľový jazyk prekladu z ponukového menu nachádzajúceho sa v strede obrazovky. Prekladaný text používateľ zadáva do textového poľa v ľavej časti obrazovky. Celá akcia prekladu sa spustí stlačením tlačidla Prelož. Následný výstup z prekladu je zobrazený v elemente na pravej strane obrazovky. Pod týmto elementom sa zobrazí aj nadpis Alternatívne preklady a pod ním je vždy na novom riadku ponúknutá alternatíva prekladu. Elementy s prekladaným a preloženým textom sa automaticky prispôsobia šírke používateľovej obrazovky pomocou CSS atribútu width (šírka) zadaného v percentách.



Obr. 4 - Ukážka webového rozhrania služby na štatistický preklad textu

Návrh webovej služby

Keďže náš prekladač má poskytovať služby nielen pre našu aplikáciu, ale má pomáhať aj ďalším programátorom chceme ho poskytnúť ako webovú službu. Táto služba by mala v prvej fáze poskytovať služby na preklad textu a získanie všetkých dostupných prekladov. V budúcnosti ju plánujeme rozšíriť o ďalšie možnosti, ako napríklad nastavenie prekladu a podobne.

3.1.3 Implementácia

Implementácia grafického rozhrania

Používateľ aplikácie na štatistický preklad textu zadáva požadované údaje pomocou grafického používateľského rozhrania (angl. GUI). Používateľ má možnosť zadať voľbu zdrojového a cieľového jazyka prekladu spolu s prekladaným textom. Tieto elementy, do ktorých používateľ zadáva hodnoty, sú umiestnené v HTML formulári. Potvrdením HTML formulára tlačidlom Prelož, sú hodnoty odoslané na server v hodnotách globálnych PHP premenných $_POST[‘type_here’] a $_POST[‘language’]. Po ich spracovaní na strane servera je stránka s webovým rozhraním aplikácie znovu načítaná a používateľovi je zobrazený výsledok z celého procesu prekladu.

Implementácia webovej služby

Keďže sme si ako implementačné prostredie vybrali programovací jazyk Java, aj naša webová služba bude implementovaná v tomto jazyku. Tento fakt si vyžiadal nainštalovanie servera Tomcat na náš server.

Následne sme vytvorili webovú službu s dvoma metódami.Webová služba sídli na adrese:

http://147.175.159.145:8080/aDictItTranslator/services/TranslatorService?wsdl

V nej je možné volať metódy:

getFirstTranslate

je to metóda na získanie prvého a teda najpravdepodobnejšieho prekladu daného slova. Vracia teda objekt typu *string* v podobe preloženého slova. Táto metóda vyžaduje parametre:

word – typu *string* – je slovo, ktoré chceme preložiť
lang_from – typu *string* – trojpísmenové označenie jazyku v ktorom sa nachádza slovo
lang_to – typu *string* – trojpísmenové označenie jazyku v ktorom chceme mať preložené slovo

getTranslates

metóda získava všetky preklady daného slova. Vracia pole objektov *string*, v ktorom sa tieto preklady nachádzajú.

word – typu *string* – je slovo, ktoré chceme preložiť
lang_from – typu *string* – trojpísmenové označenie jazyku v ktorom sa nachádza slovo
lang_to – typu *string* – trojpísmenové označenie jazyku v ktorom chceme mať preložené slovo

3.1.4 Testovanie

Zobrazenie rozhrania

Pre testovanie viditeľnosti rozhrania sme použili nasledovný testovací scenár, podľa ktorého sme otestovali funkčnosť implementácie. Tento scenár sme opakovali v prehliadačoch IE6, IE7, IE8, IE9, Mozilla Firefox a Google Chrome všetky na platforme Windows.

Názov	Zobrazenie rozhrania	ID Testu	01-01-01
Rozhranie	Domovská stránka aplikácie	ID UC	01
Účel	Zobrazenie rozhrania webovej aplikácie		
Vstupné podmienky	Žiadne		
Výstupné podmienky	Používateľ má zobrazené rozhranie		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**
1.	Zadanie adresy prekladača do internetového prehliadača	Zobrazenie všetkých elementov rozhrania na očakávaných pozíciách	Všetky elementy rozhrania sa zobrazili na očakávaných pozíciách

Schopnosť odpovede

Pre overenie odosielania odpovede služby sme použili nasledovný testovací scenár, podľa ktorého sme otestovali funkčnosť implementácie.

Názov	Odpoveď služby	ID Testu	01-01-02

Rozhranie	Domovská stránka aplikácie	ID UC	01
Účel	Overenie odpovede služby		
Vstupné podmienky	Zadaný výraz		
Výstupné podmienky	Používateľ vidí na obrazovke preložený výraz		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**
1.	Získanie odpovede služby	Zobrazenie výsledku prekladu a alternatív prekladu	Výsledky prekladu vrátane alternatívnych výsledkov sa zobrazili v požadovaných HTML elementoch

3.2 Preklad slova

Ako používateľ chcem preložiť slovo.

3.2.1 Analýza existujúcich slovníkov

Na preklad slova z jedného jazyka do druhého je potrebné, mať slovník s dostatočne veľkým počtom hesiel. Veľká časť koncových používateľov používa webové slovníky. Na internete je ich dostatok a väčšinou poskytujú podobné rozhrania a funkcionalitu. Na účely nášho projektu sa však nemôžeme spoliehať na webové služby externých entít. Lepším riešením je disponovať kompletným slovníkom, čím získame flexibilitu manipulácie s ním a hlavne efektívnosť. Pre tento účel sme sa snažili nájsť dostatočne veľký slovník, teda taký, ktorý by pokryl čo najväčšiu časť jazyka. Zo začiatku sme sa rozhodli implementovať preklad z anglického do slovenského jazyka.

Wiktionary - Wikislovník

Wiktionary je projekt, ktorý vznikol za účelom vytvorenia voľného viacjazykového slovníka. Princíp je podobný ako u Wikipedie. Heslá do slovníka pridávajú používatelia - dobrovoľníci. Pridávajú aj definície jeho významu, preklady do viacerých jazykov a slovné tvary vo všetkých pádoch. Slovenský slovník však v súčasnosti obsahuje len približne 1500 hesiel. O niečo lepšie je na tom český slovník, ktorý má viac než 29000 hesiel, ale aj to pokrýva len veľmi malú časť jazyka na to, aby vytvoril plnohodnotný prekladač.

Freelang

Freelang je voľne dostupný softvér, ktorý slúži na precvičovanie cudzieho jazyka a na preklad slov. Poskytuje možnosť stiahnuť prekladový slovník z veľkého množstva dvojíc jazykov. Slovensko-anglický slovník obsahuje len o niečo viac než 3000 hesiel, teda len tie najčastejšie slová jazyka, čo pre prekladač textu, ktorý plánujeme vytvoriť rozhodne nemôže stačiť.

PC Translator

PC Translator je desktopová aplikácia, ktorá slúži ako prekladač. Je veľmi populárny na území Českej a Slovenskej republiky. Je dostupný v ôsmych jazykových verziách, pričom slovenský slovník obsahuje takmer 800 000 hesiel. Pri hlbšej analýze sme zistili, že každý prekladový slovník v aplikácii je uložený v súbore formátu DBF. Tento formát súboru sme schopní spracovať a

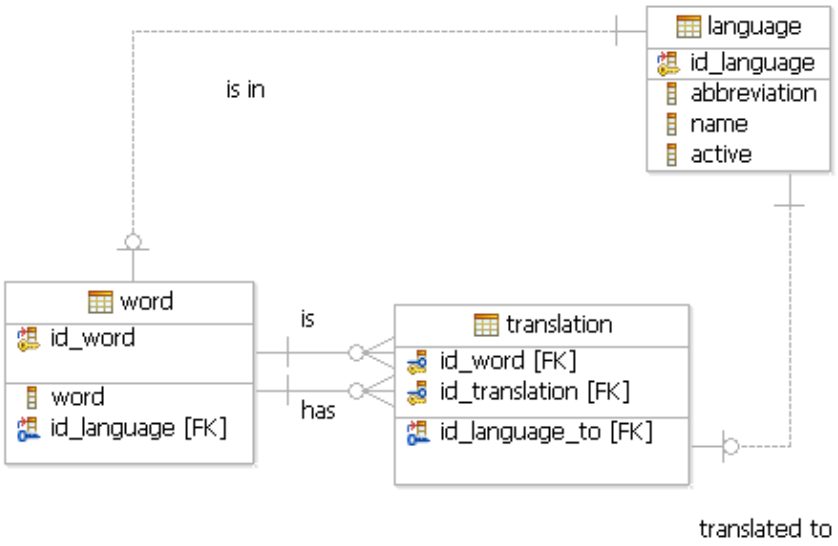
prispôsobiť pre naše potreby. V neskoršej fáze projektu by bolo možné pridať aj slovníky iných jazykov.

Zhodnotenie

Rozhodli sme sa použiť slovník z aplikácie PC Translator. Dôvodom je hlavne dobrá slovná zásoba a spracovateľnosť jeho dát. Ostatné dostupné slovníky neposkytujú dostatočné pokrytie jazyka a tým pádom nemôžu slúžiť k vytvoreniu plnohodnotného prekladača.

3.2.2 Návrh slovníka

Na obrázku Obr. 4 je zobrazený fyzický model slovníka. Pozostáva z tabuliek word,language a translation. Tabuľka language predstavuje jednotlivé jazyky ktorých slová sú v slovníku obsiahnuté. V tabuľke word je definované slovo a jazyk do ktorého patrí. Tabuľka translation je väzobnou tabuľkou ktorá určuje, že ktoré slovo je prekladom inému slovu v určenom jazyku.



Obr. 5 - Fyzický model slovníka

Detailný popis tabuliek

Tabuľka 1 - Entita fyzického modelu - language

Názov	language		
Kľúč	***Názov***	***Typ***	***Opis***
PK	id_language	int(11)	Primárny kľúč tabuľky
	abbreviation	varchar(5)	Skratka názvu jazyka
	name	varchar(100)	Názov jazyka
	Active	int(1)	Príznak aktívnosti jazyka, či je možné použiť pri preklade tento jazyk

Tabuľka 2 - Entita fyzického modelu - word

Názov	word		
Kľúč	***Názov***	***Typ***	***Opis***

PK	id_word	int(11)	Primárny kľúč tabuľky
FK	id_language	int(11)	Referencia na jazyk, do ktorého slovnej zásoby dané slovo patrí
	word	varchar(255)	Slovo v slovníku

Tabuľka 3 - Entita fyzického modelu - translation

Názov	translation		
Kľúč	***Názov***	***Typ***	***Opis***
PK	id_word	int(11)	Časť primárneho kľúča tabuľky, referencia na slovo, ku ktorému je definovaný preklad
PK	id_translation	int(11)	Časť primárneho kľúča tabuľky, referencia na slovo, ktoré je prekladom pôvodného slova v danom jazyku
FK	id_language_to	int(11)	Referencia na jazyk, do ktorého slovnej zásoby preklad patrí

3.2.3 Implementácia

Slovník prekladača je implementovaný ako databáza v MySQL. Pre získanie prekladu bola implementovaná metóda v jazyku Java. Na komunikáciu s databázou bolo použité rozhranie JDBC.

Princíp je veľmi jednoduchý. Vstupnými parametrami sú slovo, ktoré má byť preložené, jazyk, z ktorého slovo pochádza a výsledný jazyk. Výstupom metódy je zoznam slov, nakoľko jedno slovo môže mať viacero prekladov a synoným.

Zdrojový kód funkcie:

```
PreparedStatementselWord = cm.prepareStatement(
                "SELECT " +
                        "W_TO.word " +
                "FROM " +
                        "word W_FROM " +
                        "JOIN translation T ON W_FROM.id_word = T.id_word " +
                        "JOIN word W_TO ON W_TO.id_word = T.id_translation " +
                "WHERE " +
                        "T.id_language_to = ? AND " +
                        "W_FROM.word = ? AND " +
                        "W_FROM.id_language = ?"
);

selWord.setInt(1,id_language_to);
selWord.setString(2, wordToTranslate);
selWord.setInt(3, id_language_from);

ResultSetrs = selWord.executeQuery();

List<String>translations = newArrayList<String>();

while(rs.next())
{
        translations.add(rs.getString("word"));
```

}

Testovanie

Pre overenie funkcionality slovníka, sme vytvorili nasledujúce testovacie scenáre ktoré realizujú preklad slov z angličtiny do slovenčiny a zo slovenčiny do angličtiny.

Tabuľka 4 - Akceptačný test prekladu do slovenčiny

Názov	Preklad slova eng->svk		ID testu	01-02-01
Rozhranie	GUI služby (možno premenovať)		ID UC	01
Účel	Preklad slova z angličtiny do slovenčiny			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**	
1.	Zo selectu na výber prekladu zvolíme možnosť eng->svk	V selectesazvolí možnosť eng->svk	V selectesazvolila možnosť eng->svk	
2.	Do ľavého poľa napíšeme slovo car	V ľavom poli zostane napísané zadané slovo	V ľavom poli zostalo napísané zadané slovo	
3	Klikneme na tlačidlo prelož	Slovo (car) sa preloží a jeho preklad (auto) sa zobrazí v pravom poli	Slovo (car) sa preložílo a jeho preklad (auto) sa zobrazí v pravom poli	

Tabuľka 5 - Akceptačný test prekladu do angličtiny

Názov	Preklad slova svk->eng		ID testu	01-02-02
Rozhranie	GUI služby (možno premenovať)		ID UC	01
Účel	Preklad slova zo slovenčiny do angličtiny			
Vstupné podmienky	Žiadne			
Výstupné podmienky	Žiadne			
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**	
1.	Zo selectu na výber prekladu zvolíme možnosť svk->eng	V selectesazvolí možnosť svk->eng	V selectesazvolila možnosť svk->eng	
2.	Do ľavého poľa napíšeme slovo voda	V ľavom poli zostane napísané zadané slovo	V ľavom poli zostalo napísané zadané slovo	
3	Klikneme na tlačidlo prelož	Slovo (voda) sa preloží a jeho preklad (water) sa zobrazí v pravom poli	Slovo (voda) sa preložilo a jeho preklad (water) sa zobrazil v pravom poli	

4 Druhý šprint – kávička

Druhý šprint sme pracovne nazvali pivo, ako mierne silnú návykovú látku. V tomto šprinte sme sa zamerali na nasledujúce fakty:

1. Preklad vety zadanej v jazyku
2. Zapojenie článkov z databázy sme do procesu
3. Pridanie tvaroslovia

Výstupom druhého šprintu je prekladač, ktorý dokáže preložiť vetu a v prípade že ju nájde v našom korpuse sme článkov tak vetu dokáže aj správne vyskloňovať.

4.1 Preklad vety

Ako *používateľ* chcem preložiť *vetu*.

4.1.1 Analýza

Analýza existujúcich riešení na preklad textu

V súčasnosti je dostupných niekoľko možností ako rýchlo získať preklad textu. Používateľ má k dispozícii veľké množstvo webových služieb ale aj prekladačov, ktoré môžu byť distribuované i ako inštalácie do osobného počítača alebo notebooku. Väčšina z nich vytvára potenciálne preklady, ktoré následné ohodnotí percentom úspešnosti. Niektoré tiež poskytujú interakciu na overenie prekladu, či možnosť jeho korektného napísania.

V analýze sa budeme podrobnejšie zaoberať nasledujúcimi prekladačmi:

- ***Google Translator***
  - najznámejší a najpoužívanejší
  - aktuálne zvláda preklad do 64 jazykov
  - nevýhodou je obmedzené používanie, keďže Google Translate API v2 bolo spoplatnené
  - medzi výhody patrí overenie prekladu používateľom
- ***Bing***
  - vyvinutý z Live Search Translator a Windows Live Translator
  - druhá najpoužívanejšia webová služba slúžia na preklad textu
  - podporuje 37 jazykov
- ***World Lingo***
  - platená verzia rozpoznáva viac ako 141 jazykov
  - neplatená verzia rozpoznáva iba 32 jazykov a absentuje slovenčina
- ***MOSES***
  - prekladač, vyvíjaný niekoľkými vysokoškolskými študentmi a pedagógmi niekoľkých krajín sveta
  - automatické učenie sa prekladov
  - nezávislosť od jazyka – stačí množina korpusov v dvoch rôznych jazykoch
  - absencia slovenčiny

- ***PC Translator 2010***
  - produkt spoločnosti LangSoft s.r.o., ktorá sa jeho zdokonaľovaním zaoberá už od roku 1988
  - výhodou je modul, ktorý používateľa naučí cudzí jazyk
  - nevýhoda je menší počet jazykov

Preklad viet

Skúmanie výsledkov prekladu nemusí byť vždy objektívne. Posudzovať ich správnosť napríklad podľa korektnosti celej vety, nie je príliš vhodné. Preto je potrebné zadefinovať systém ohodnocovania viet, ktorý bude brať do úvahy aj iné aspekty. Pri jazykoch ako je slovenčina a čeština to môže byť napríklad skloňovanie a časovanie, pri angličtine zase vhodnosť použitia daného slova alebo výsledný slovosled vo vete. V nasledujúcej tabuľke je uvedené bodové skóre, podľa ktorého boli vety pri prekladoch vyhodnocované.

Tabuľka 6 - Hodnotenie viet podľa úspešnosti prekladu.

Preklad	Bodov
správny preklad	*5*
nesprávny preklad	*0*
časovanie	*3*
skloňovanie	*3*
slovosled	*4*
iné alebo zle preložené slovo	*1*
kombinácia chýb	***počet / 2***

Majme konečnú množinu, ktorá obsahuje vety na preloženie. Postupne spravíme ohodnotenie každej z nich – hodnota i-tého ohodnotenia je $body_i \in <0,5>$. Ich súčet tvorí čitateľa zlomku.

Menovateľ je zložený zo súčinu celkového počtu viet N a maximálneho počtu bodov – max_body (v našom prípade 5). Ich podiel vynásobený číslom sto tvorí percento úspešnosti daného prekladača.

Rovnica 1 - Výpočet úspešnosti prekladača.

$$úspešnosť = \left[\frac{\sum_{i=1}^{N} body_i}{N * \max\_body} \right] * 100;$$

Zhodnotenie prekladu viet

Z daného prieskumu (Príloha A) jednoznačne vyplýva, že najväčším problémom pre prekladače je jazyk obsahujúci veľa skloňovacích pravidiel. V prípade jednoduchších jazykov (ako napríklad angličtina) je výsledok zobrazený používateľovi buď celý správny alebo sa nedá vôbec pou-

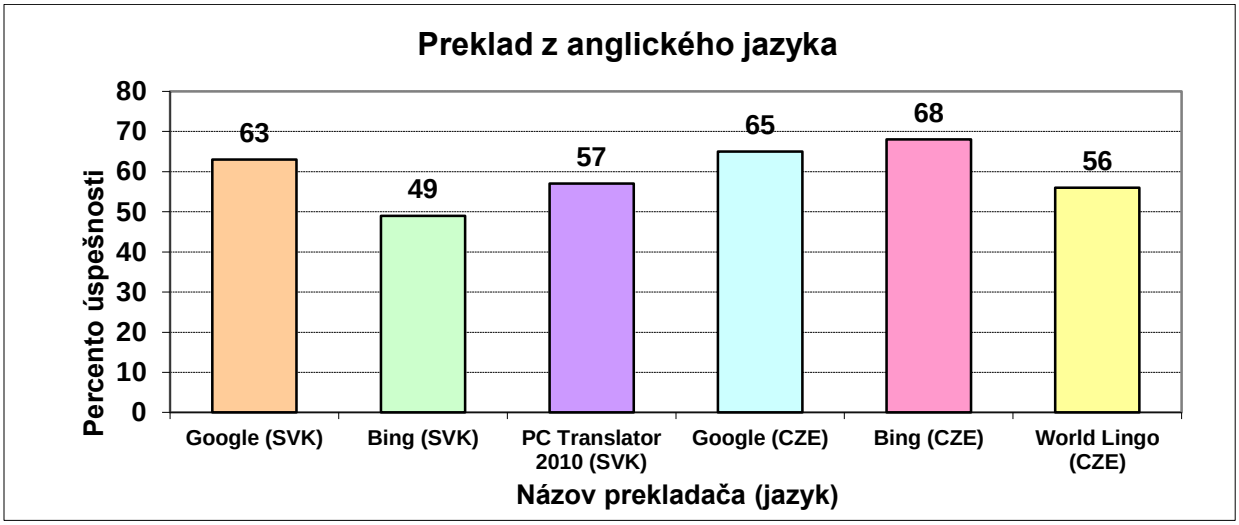
žiť. Jediným z problémom je aj časovanie slovies. Správny tvar sa však nedá zistiť, keďže vety sú zväčša vytrhnuté z kontextu.

Celkové výsledky:

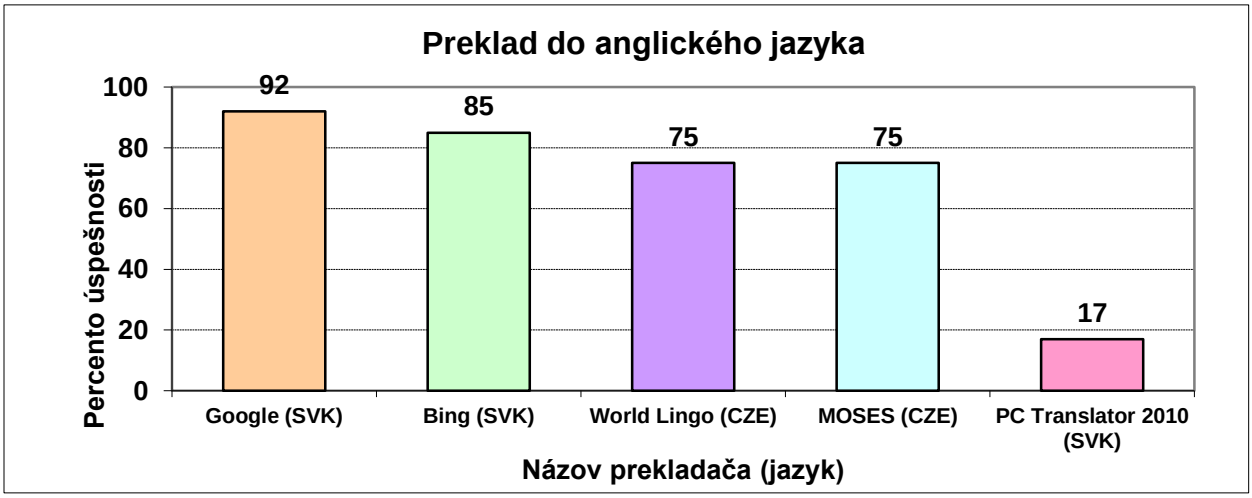
1. Google Translator – 74%
2. Bing – 67 %
3. World Lingo – 52 %
4. PC Translator – 37 %

Poznámka:

- MOSES dokáže prekladať iba jedným smerom a preto nebol zahrnutý do celkových výsledkov



Obr. 6 - Graf úspešnosti prekladu vety z anglického jazyka.



Obr. 7 - Graf úspešnosti prekladu vety do anglického jazyka.

Preklad častí textu - paragrafov

Predchádzajúca kapitola (+ Príloha A) rozoberala preklad jednej vety medzi dvoma jazykmi. Používateľ ale zväčša potrebuje získať ucelený pohľad na určitú časť kontextu, ktorej nerozumie. Preto je nevyhnuté, aby prekladače vedeli spracovávať zadané údaje nielen po slovách či vetách, ale aby aj odvodzovali vzťahy medzi jednotlivými časťami kontextu. Jednotlivo preložené vety by mohli byť z rôznych tém, vyskytovalo by sa miešanie časov, či mužských a ženských osôb. Preložená časť by pôsobila rušilo.

Na výpočet ohodnotenia viet použijeme Rovnica 1 z kapitoly „Preklad viet“ a bodovanie zTabuľka 7.

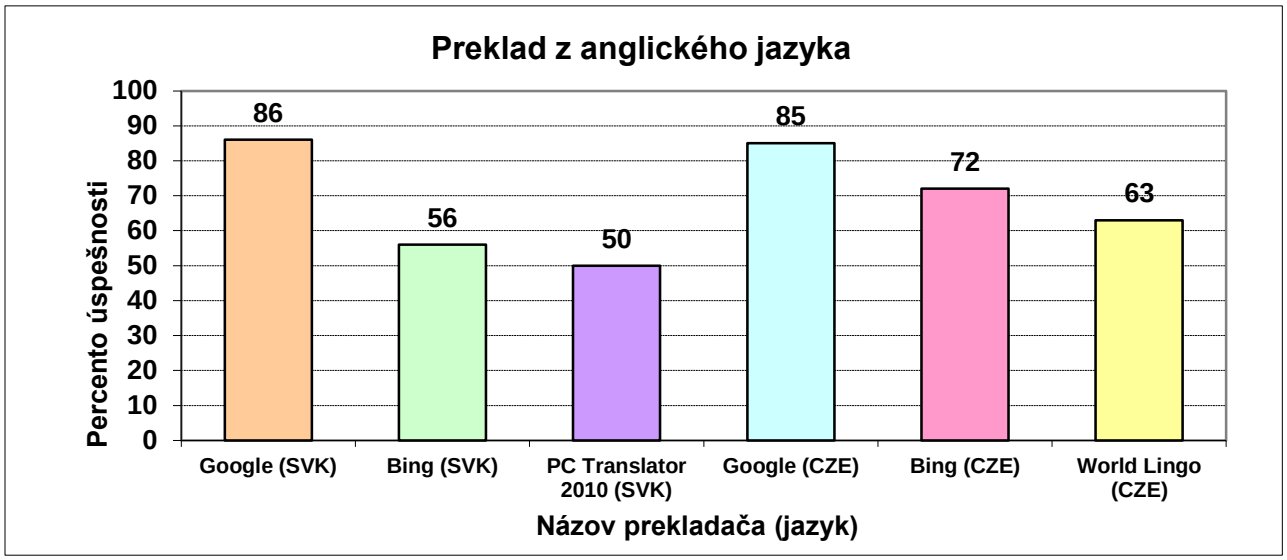
Tabuľka 7 - Hodnotenie viet podľa úspešnosti prekladu.

Preklad vety v kontexte	Bodov / slovo
správny preklad	*5*
nesprávny preklad	*0*
časovanie	*3*
skloňovanie	*3*
slovosled	*4*
~~*slovo navyše*~~	~~*2*~~
iné alebo zle preložené slovo	*1*

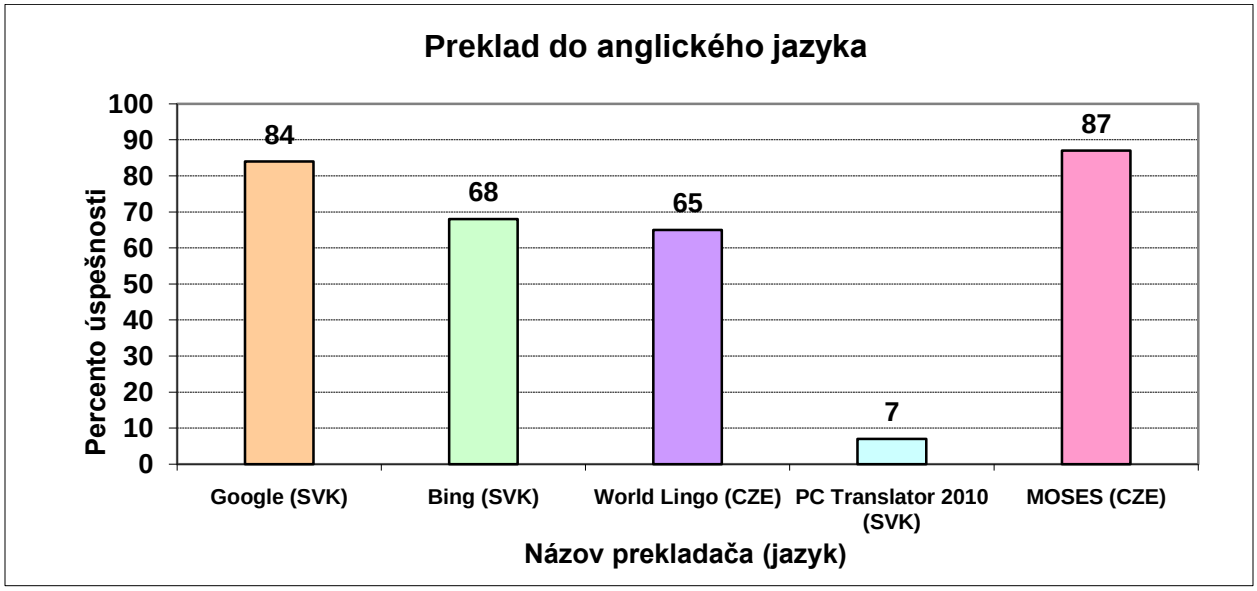
Zhodnotenie prekladu paragrafov

Prekladanie ucelených častí textu (Príloha B) je oveľa zložitejší proces ako preklad samostatnej vety. Existujúca dejová línia, kumulovanie prídavných mien, striedanie časov, či dialógy osôb. To všetko je potrebné zapracovať do algoritmu prekladu. Postupné prekladanie slov a ich vkladanie do vety môže mať úspech iba pri jazykoch, ktoré nepoznajú skloňovanie. V opačnom prípade však narážame na problém. Príkladom takéhoto prekladača je napríklad PC Translator 2010.

Percentuálna úspešnosť bola znázornená pomocou dvoch stĺpcových grafov. Príjemným prekvapením je MOSES, ktorého schopnosti je možné porovnávať s takým gigantom ako je Google Translate. Škoda len, že sa špecializuje iba na preklad z češtiny do angličtiny.



Obr. 8 - Graf úspešnosti prekladu textu z anglického jazyka.



Obr. 9 - Graf úspešnosti prekladu textu do anglického jazyka.

Celkové zhodnotenie

Analýza problémovej oblasti bola zameraná na porovnanie prekladu medzi anglickým, českým a slovenským jazykom. Bola zložená z dvoch častí, ktoré jednoznačne potvrdili, že skloňovanie je najväčším „nepriateľom“ prekladača.

Počítanie percentuálnej úspešnosti bolo vykonané s pomocou presne zadefinovaného modelu ohodnocovania. Výsledky však stále nie sú uspokojivé. Je potrebné ich neustále zdokonaľovať. To možno dosiahnuť napríklad aplikovaním novej funkcie, ktorá určí pravdepodobnosti výskytu za pomoci iných kauzálnych vzťahov.

V súčasnosti na trhu absentuje nástroj, ktorý by bol schopný konkurovať, či prípadne nahradiť najznámejšiu aplikáciu slúžiacu na preklad textov. Ním je Google Translate. Ten však „uštedril" programátorom tvrdý úder pod pás – spoplatnil Translate API. Nastal teda čas pokúsiť sa vymyslieť niečo nové.

Analýza slovenského jazyka a spojených problémov

Slovenský jazyk

Pre preklad textu do slovenského jazyka je nevyhnutné analyzovať problémy spojené s jazykom. Slovenčina patrí medzi západoslovanské jazyky a je používaná viac ako šiestimi miliónmi ľudí. Zaraduje sa medzi tzv. flexné jazyky a je charakteristická bohatou morfológiou. Slová v slovenčine majú viacero tvarov a pravidlá pre tvorbu tvarov nie sú striktne definované. Tento fakt predstavuje problém pre tvorbu algoritmov, ktorých cieľom je úprava slova na konkrétny tvar. Na zápis slovenčiny sa používa slovenská abeceda pozostávajúca z modifikovaných písmen latinky a je rozšírená o diakritické znamienka. Písanie diakritických znamienok sa v neformálnom texte alebo komunikácií prostredníctvom internetu často nevyužíva [1, 2].

Získanie konkrétneho tvaru slova

Zmeniť slovo do požadovaného tvaru je dôležitý krok pri preklade. Bez neho by nebolo možné zo slov vyskladať spisovnú slovenskú vetu. V praxi sa najčastejšie využíva opačný proces, a to získanie základného tvaru slova – lematizácia. Keďže by stačilo tento proces otočiť, aby sme získali ľubovoľný tvar slova, analyzujeme možné prístupy lematizácie. Vo všeobecnosti sa delia do dvoch skupín a to [3]:

- slovníkové
- pravidlové

Slovníkové prístupy využívajú databázu slov jazyka a všetkých prislúchajúcich tvarov. Aktuálna databáza slovenských slov od Jazykovedného ústavu Ľudovíta Štúra pozostáva z 2 472 051 záznamov vo forme základný tvar a odvodený tvar. Jedinou nevýhodou tejto databázy je nepoužiteľnosť pre nové slova jazyka, ktoré v nej nie sú obsiahnuté. Úspešnosť získavania základného tvarou slovníkovým prístupom dosahuje veľmi dobré výsledky. Databáza slov od jazykovedného ústavu upravená na lematizátor dosiahla v experimente na bežných a odborných článkoch (3209 slov) úspešnosť lematizácie až 96,1% [2].

Pravidlové prístupy sa opierajú o pravidlá tvorby slovných tvarov a snažia sa ich aplikovať pre získavanie základného tvaru. Medzi nástroje využívajúce takýto prístup patrí nástroj *Tvaroslovník*[1]. Autori vytvorili slová, ktoré predstavujú šablóny ohýbania pre iné slová a pomocou pravidiel podvojnej výmeny je možné získavať ľubovoľné tvary slov. Výhodou v porovnaní so slovníkovým prístupom je možnosť získať základný tvar aj pre nové slová, ktoré sa v databáze slov nenachádzajú. Potrebné je iba nájsť vhodnú šablónu pre ohýbanie požadovaného slova. Nástroj

[1] http://nazou.fiit.stuba.sk/home/index.php
[2] Napríklad www.titulky.sk, www.opensubtitles.org

Tvaroslovník v rovnakom experimente, na akom bola testovaná databáza slov od Jazykovedného ústavu Ľ. Štúra, dosiahol úspešnosť lematizácie 84,2% [2, 3].

Vidíme, že slovníkový prístupje pre úpravu slova na základný tvar efektívnejší, a preto predpokladáme, že bude efektívnejší aj pri získavaní odvodených slovných tvarov.

Analýza databázy SME článkov

Databázu SME článkov tvorí 99164 článkov, ktoré boli získané na fakulte v rámci výskumného projektu. Tie bolo zabalené v SQL súbore, ktorého spustením sa vytvorí a naplní tabuľka articles.

#	Stĺpce	Typ	Zotriedenie	Atribúty	Nulový	Predvolené	Extra	Akcia
1	id	int(11)			Nie	None	AUTO_INCREMENT	Zmeniť Odstrániť More
2	title	varchar(255)	utf8_general_ci		Nie	None		Zmeniť Odstrániť More
3	body	longtext	utf8_general_ci		Nie	None		Zmeniť Odstrániť More
4	author	varchar(255)	utf8_general_ci		Áno	NULL		Zmeniť Odstrániť More
5	published_at	datetime			Nie	None		Zmeniť Odstrániť More
6	created_at	datetime			Nie	None		Zmeniť Odstrániť More
7	updated_at	datetime			Nie	None		Zmeniť Odstrániť More
8	logger_url	varchar(255)	utf8_general_ci		Áno	NULL		Zmeniť Odstrániť More
9	perex	longtext	utf8_general_ci		Nie	None		Zmeniť Odstrániť More
10	section	varchar(255)	utf8_general_ci		Nie	None		Zmeniť Odstrániť More
11	category	varchar(255)	utf8_general_ci		Nie	None		Zmeniť Odstrániť More
12	url	varchar(255)	utf8_general_ci		Nie	None		Zmeniť Odstrániť More
13	ari	decimal(8,3)			Áno	NULL		Zmeniť Odstrániť More
14	type	varchar(15)	utf8_unicode_ci		Nie	None		Zmeniť Odstrániť More

Označiť všetko / Odznačiť všetko *Výber:* Prechádzať Zmeniť Odstrániť Primárny Unikátny Index Celý text

Obr. 10 - Štruktúra tabuľky articles.

Primárnym kľúčom je „id“. V stĺpci body je uložený text celých článkov. Ostatné stĺpce aktuálne nie sú pre náš projekt podstatné.

Problémy:

1. text článkov obsahuje nepotrebné html značky
2. hľadanie slov alebo viet v celých článkov je pomalé – potreba rozdelenia na menšie časti
3. vhodný návrh a prepojenie tabuliek

4.1.2 Návrh

Metóda pre preklad textu

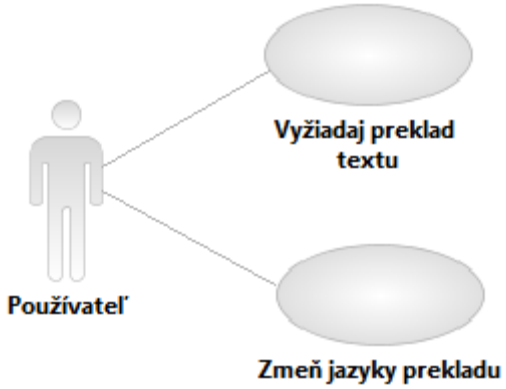
Preklad textu z jedného jazyka do druhého je komplikovaný proces, na ktorý samotné preloženie jednotlivých slov nie je postačujúce. Pre úspešný preklad je dôležité aby preložená veta dodržala sémantiku jazyka a vznikla tak zmysluplná veta. Analýza Slovenského jazyka ukázala, že vytvorenie algoritmu pre generovanie vhodných slov je veľmi náročný proces a kvôli nie striktne defi-

novaným pravidlám tvorby slov nemôže dosahovať veľkú úspešnosť. Preto sme snahu o vytvorenie pravidiel tvorby viet zavrhli hneď na začiatku.

Jednou alternatívou riešenia je použitie prístupu spomínaného u nástroja Moses. Pre vytvorenie takéhoto riešenia je nevyhnutné získať paralelne preložené korpusy, ktoré budú slúžiť ako množina príkladov pre preklad textu. Avšak existujúce paralelné texty pre slovenčinu sú nevhodné pre preklad obyčajného textu a vytvorenie vlastných korpusov je časovo príliš náročný proces. Navrhujeme preto použiť prístup založený na štatistike,ktorý bude kombinovať preklad jednotlivých slov, získanie ich odvodených tvarov a generovanie potenciálnych prekladov. Ich správnosť sa overí pomocou štatistických metód na korpuse slovenských textov.

Prípadypoužitia

V našom systéme sme identifikovali nasledovné prípady použitia (Obr. 10):



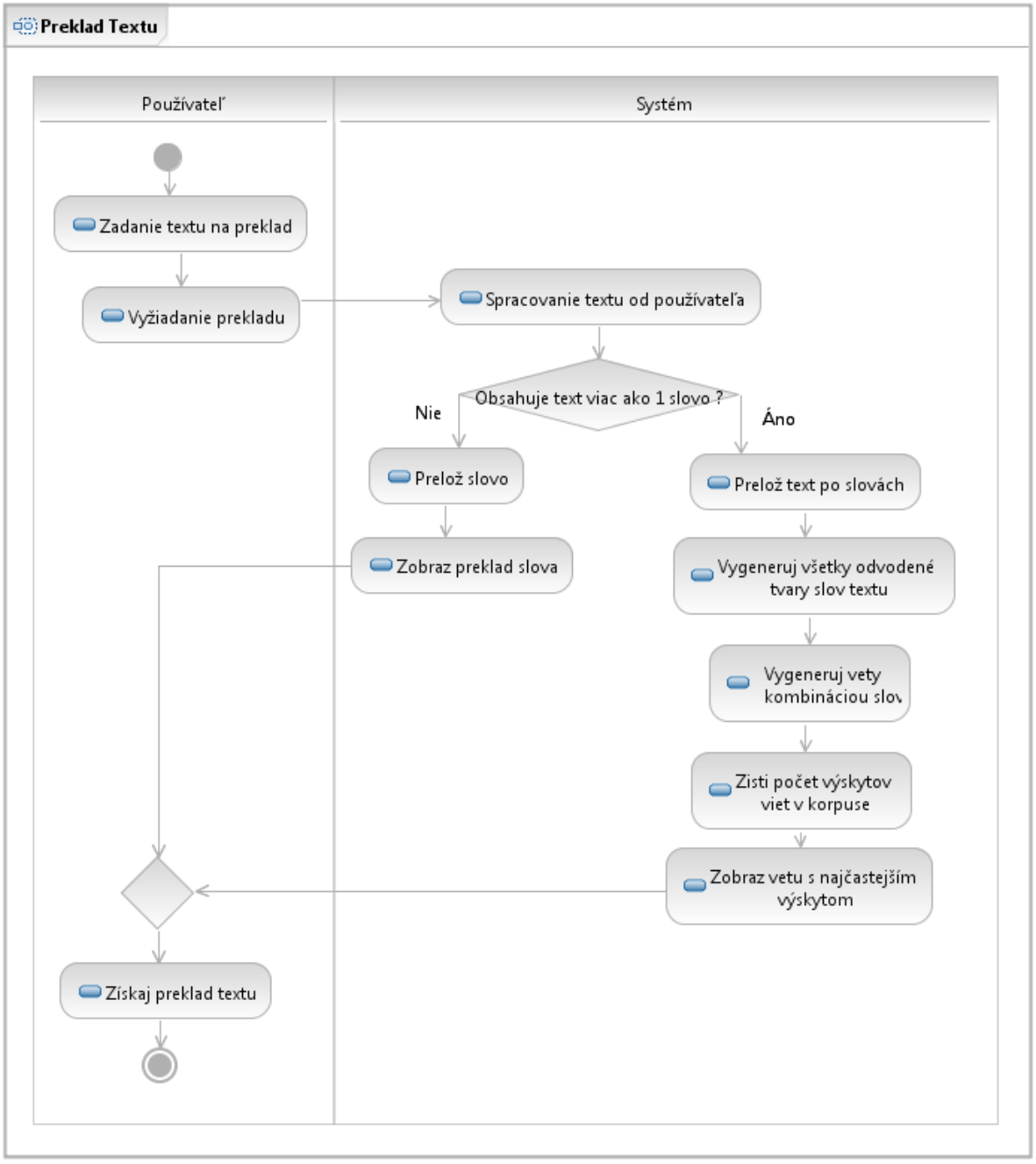
Obr. 11 - Diagram prípadov použitia.

Aby bolo možné získať preklad textu je nevyhnutné, aby súčasťou systému boli nasledujúce časti:

- slovník pre preklad jednotlivých slov
- generátor slovných tvarov pracujúci nad databázou slovenských slov od Jazykovedného ústavu Ľudovíta Štúra
- generátor viet zo všetkých kombinácií slov
- korpus slovenských textov
- vyhľadávač potenciálnych prekladov v korpuse
- webová stránka poskytujúca jednoduché rozhranie
- webová služba pre prepojenie používateľského rozhrania s jadrom aplikácie

Diagram aktivít

Ako je vidieť na Obr. 11, proces začína vložením textu na preklad do systému od používateľa. Prvým krokomje rozdelenie textu na slová, ktoré sú po jednom preložené pomocou slovníka. Tým dostaneme všetky významy preložených slov. Ďalším krokom je získanie všetkých odvodených tvarov preložených slov prostredníctvom databázy tvarov slovenských slov. To by malo

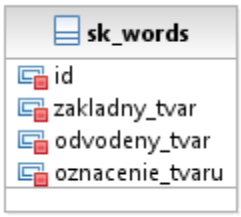


Obr. 12 - Diagram aktivít.

zabezpečiť, že sa vygenerujú správne vyskloňované slová. Nasleduje tvorenie viet kombinovaním vygenerovaných tvarov, čím vznikne množina potenciálnych prekladov. Posledným krokom je hľadanie týchto viet v korpuse. V prípade, že sa podarí nájsť presnú zhodu, hľadanie je zastavené a používateľovi sa zobrazí príslušná veta ako preklad. Zvyšné vety sú zobrazené len ako alternatívne preklady.

Fyzický model generátora slovných tvarov a viet

Na Obr. 12je zobrazený fyzický model databázy slov z Jazykovedného ústavu Ľudovíta Štúra. Obsahuje len jednu tabuľku, v ktorej sú uložené všetky tvary slovenských slov, označenie týchto tvarov a prislúchajúce základné tvary.



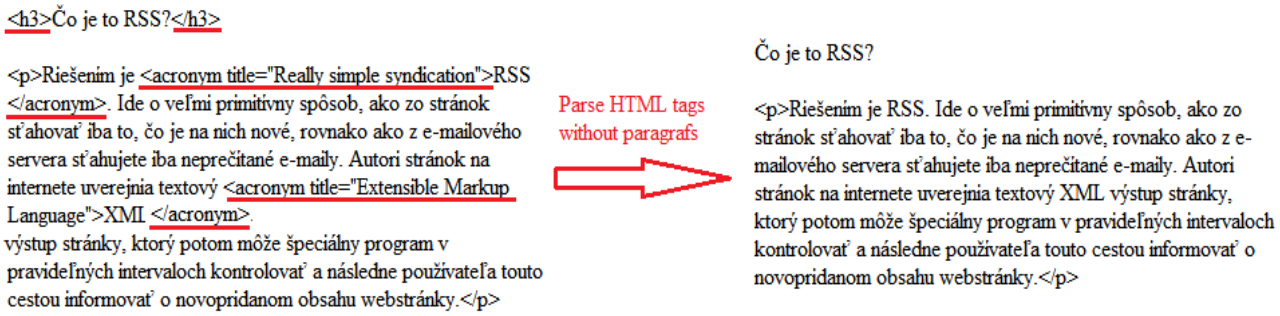
Obr. 13 - Fyzický model generátora slovných tvarov.

V tabuľke sa bude často hľadať buď zakladaný tvar slova alebo jeho odvodený tvar.Keďže obsahuje približne 2,4 milióna záznamov je dôležité, aby tieto atribúty boli indexované.

Návrh metódy využitia sme článkov

Problém č.1: Text článkov obsahuje nepotrebné html značky.

Každý z článkov obsahuje konkrétny text, ktorý je zapísaný vo forme vhodného výstupu na webovú stránku týždenníka SME. Obsahuje teda rôzne HTML značky, ktoré slúžia na oddelenie paragrafov, na zvýraznenie určitých častí, či na presun na inú stránku. Často krát sa v strede článku nachádzajú i reklamy. Pre štatistickú prácu a spracovanie dát je potrebné, aby tieto články neobsahovali HTML značky. Jedinou výnimkou je označenie paragrafu (odseku), ktoré neskôr poslúži ako tzv. „okolie“ slova. To znamená, že za pomoci kontextu budeme vedieť určiť konkrétny význam slova. V slovenčine, ale i v iných jazykoch, existujú slová, ktoré majú niekoľko významov. Medzi takéto slovo patrí napríklad slovo hlava.



Obr. 14 - Odstránenie nepotrebných HTML značiek.

Problém č.2: Hľadanie slov alebo viet v celých článkov je pomalé.

Vyhľadávanie kľúčových slov, prekladov, či viet nad veľkým korpusom prác je neefektívne. Text článku je preto potrebné rozdeliť na menšie časti – vety. Každá veta musí končiť bodkou,

otáznikom alebo výkričníkom. Tu však je potrebné zdôrazniť, že aj číslovania alebo iniciálky mien obsahujú bodku.

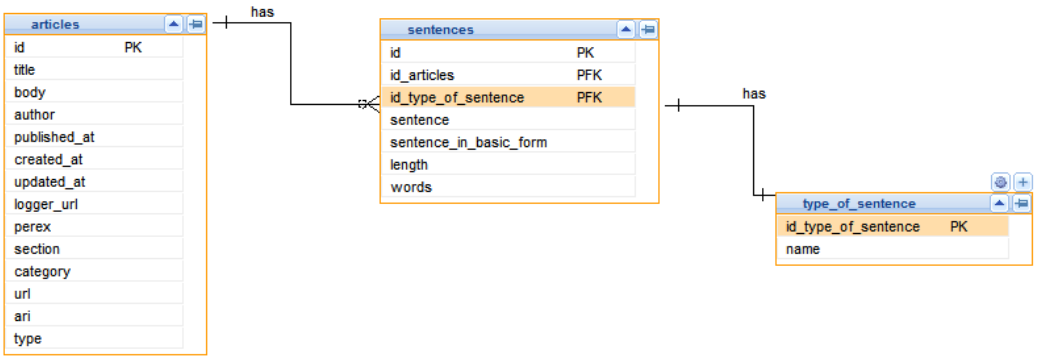
Algoritmus navrhnutého riešenia:

Krok č.1.: načíta text článku z tabuľky articles
Krok č.2.: odstráni značky paragrafov
Krok č.3.: odstráni medzery za bodkou, dvojbodkou, otáznikom a výkričníkom
Krok č.4.: odstráni v texte všetky dve a viac medzery za sebou
Krok č.5.: nastaví začiatok vety na nulu
Krok č.6.: hľadá symbol bodky, dvojbodky, otáznika alebo výkričníka a za ním veľké písmeno – nastaví koniec vety a nový začiatok druhej vety
Krok č.7.: pomocou známeho začiatku a konca vytiahne danú vetu z kontextu a uloží do pola
Krok č.8.: ak už nenájde symbol bodky, dvojbodky, otáznika alebo výkričníka a za ním veľké písmeno – ide o poslednú vetu, ktorú uloží do pola
Krok č.9.: overí vytvorené pole
  a.: zmazanie príliš krátkych viet (napr. číslovanie, skratka mena a pod.)
  b.: zmazanie medzery, ak ňou veta začína alebo končí
Krok č.10.: zisti typ vety pomocou posledného znaku
Krok č.11.: zisti dĺžku a počet slov vo vete
Krok č.12.: ulož vetu do tabuľky sentences

Problém č.3: Vhodný návrh a prepojenie tabuliek.

Tabuľka articles jednoznačne identifikuje daný článok pomocou svojho parametru id. Ten sa prenáša aj do tabuľky viet (sentences), ktoré z neho vznikli. Vystupuje tu ako cudzí kľúč s názvom id_articles. Dané vety môžu byť určitého typu (oznamovacie, opytovacie, rozkazovacie), čo je reprezentované pomocou ďalšieho cudzieho kľúča id_type_of_sentences. Ten pochádza z tabuľky type_of_sentence.

Platia nasledujúce vzťahy:
- článok má niekoľko viet
- veta je práve jedného typu



Obr. 15 - Návrh databázového modelu SME článkov.

4.1.3 Implementácia

Generátor slovných tvarov a viet

Generátor je implementovaný formou metódy v jazyku Java a komunikuje s MySQL databázou slovných tvarov. Pre komunikáciu s databázou používa rozhranie JDBC. Prostredníctvom generátora je možné získať:

- základný tvar slova,
- všetky odvodené tvary slova,
- všetky kombinácie všetkých slovných tvarov zo zadanej množiny slov.

Ukážka zdrojového kódu pre získanie základného tvaru slova:

```
publicArrayList<String>getLemma(finalStringword, finalStringtableName) throwsSQLException{
	finalStringquery = "SELECT * FROM " + tableName +" WHERE odvodeny_tvar='" + word + "'";
	finalStatementstatement = connection.createStatement();
	finalResultSetresultSet = statement.executeQuery(query);
	finalArrayList<String>output = newArrayList<String>();
	int index = 0;
	try {
		while (resultSet.next()){
			output.add(index, resultSet.getString("zakladny_tvar"));
			index++;
		}
	} finally {
		statement.close();
		resultSet.close();
	}
	returnoutput;
}
```

Ukážka zdrojového kódu pre získanie odvodených tvarov slova:

```
publicArrayList<String>getWordForms(finalStringword, finalStringtableName) throwsSQLException{
	finalStringquery = "SELECT * FROM " + tableName + " WHERE zakladny_tvar =
	binary'"+word+"'";
	finalPreparedStatementstatement = connection.prepareStatement(query);
	finalResultSetresultSet = statement.executeQuery(query);
	finalArrayList<String>output = newArrayList<String>();
	int index = 0;
```

```
        try {
                while (resultSet.next()){
                        output.add(index, resultSet.getString("odvodeny_tvar"));
                        output.add(index+1, resultSet.getString("oznacenie_tvaru"));
                        index += 2;
                }
        }finally {
                statement.close();
                resultSet.close();
        }
        returnoutput;
}
```

Generátor kombinácií slovných tvarov, využíva metódu pre získavanie slovných tvarov. Vďaka rekurzívnemu volaniu tak dokáže vytvoriť všetky možné kombinácie zo zadaných slov.

Vyhľadanie vety v korpuse

Ide o metódu implementovanú v jazyku Java, ktorá komunikuje s generátorom viet, a MySQL databázou slovenských textov. Cieľom je nájdenie niektorej z potenciálnych viet v korpuse. Ak sa vetu podarí nájsť, je isté, že ide o takú, ktorá je spisovná, a tak môže byť zobrazená používateľovi. Pre komunikáciu s databázou používa rozhranie JDBC.

Ukážka zdrojového kódu pre hľadanie viet v korpuse:

```
publicStringfindInDatabase(finalArrayList<String>   vety,   finalStringuser,   finalStringpass)
throwsInstantiationException, IllegalAccessException, ClassNotFoundException, SQLException{
        finalIterator<String>itr = vety.iterator();
        PreparedStatementprepStatement = null;
        ResultSetresultSet = null;
        try {
        while (itr.hasNext()){
                finalStringtmp = itr.next();
                finalStringquery = "SELECT * FROM sentences WHERE sentence = '"+tmp+"'";
                prepStatement = connection.prepareStatement(query);
                resultSet = prepStatement.executeQuery();
                if (resultSet.next()){
                        returntmp;
                }
        }
        }finally{
                prepStatement.close();
                resultSet.close();
        }
        returnnull;
}
```

Implementácia metódy pre vyhľadávanie SME článkov

Problém č.1: Text článkov obsahuje nepotrebné html značky.

Pri odstraňovaní HTML značiek sú použité funkcie externej knižnice ***JSoup*** (v.1.6.1.) - ***parse(String text)*** a ***text()***.

```
for(result = cmd.executeQuery("SELECT COUNT(id) FROM articles"); result.next();)
{
    pocet = result.getInt(1);                                                    // ziskaj pocet viet
}

for(i = 0; i <= pocet; i++)                                                      // prejdi vsetky vety
{
    ps_select.setInt(1, i);
    for(result = ps_select.executeQuery(); result.next(); ps_update.executeUpdate())    // cyklus pre vyberanie konkretneho clanku
    {
        text = result.getString(2);                                              // ziskaj text - parameter body
        text = text.replaceAll("<p>", "_start_parag_");                          // nahrad <p> a </p> specialnym retazcom
        text = text.replaceAll("</p>", "_end_parag_");                           // stringov
        text = Jsoup.parse(text).text();                                         // odstran vsetky HTML tagy
        text = text.replaceAll("_start_parag_", "<p>");                          // nahrad vsetky specialne retazce
        text = text.replaceAll("_end_parag_", "</p>");
        ps_update.setBytes(1, text.getBytes("utf-8"));                           // uloz clanok spat do tabulky articles
        ps_update.setString(2, result.getString(1));
    }
}
```

Obr. 16 - Odstránenie nepotrebných HTML značiek s použitím JSoup funkcií.

Problém č.2: Hľadanie slov alebo viet v celých článkov je pomalé.

```
// ak na zaciatku medzera
if (vety[j].startsWith(" ")==true)
    vety[j]=vety[j].substring(1,vety[j].length());

// ak na konci medzera
if (vety[j].endsWith(" ")==true)
        vety[j]=vety[j].substring(0,vety[j].length()-1);

// ak zaciatku vety nie je pozadovny znak - nastal pripadny posun (korekcia)
if (vety[j].length()>2 && vety[j].charAt(1)==' ' && vety[j].charAt(2)>='A' && vety[j].charAt(2)<='Z')
        vety[j]=vety[j].substring(2,vety[j].length());

// odstrani skratku mena
if (vety[j].length()>2)
{
    // nastavi typ vety
    if (vety[j].endsWith(".")==true)
        typ_vety=1;
    if (vety[j].endsWith("?")==true)
        typ_vety=2;
    if (vety[j].endsWith("!")==true)
        typ_vety=3;

    //  zisti pocet slov
    for (int q=0;q<=vety[j].length()-1;q++)
        if (vety[j].charAt(q)==' ')
            pocet_slov++;

    ps_insert.setBytes(1, vety[j].getBytes("utf-8"));       // nastav text vety
    ps_insert.setInt(2, id);                                // nastav id clanku
    ps_insert.setInt(3, typ_vety);                          // nastav typ vety
    ps_insert.setInt(4, vety[j].length());                  // nastav dlzku vety
    ps_insert.setInt(5, pocet_slov);                        // nastav pocet slov
    ps_insert.executeUpdate();                              // uloz do tabulky sentences
    pocet_viet++;
}
```

Obr. 17 - Implementácia algoritmu na vytváranie viet – krok 9 až 12.

Problém č.3: Vhodný návrh a prepojenie tabuliek.

Databázový model, ktorý sa nachádza na Obr. 16 bol zrealizovaný. V súčasnosti sú tabuľky naplnené dátami.

4.1.4 Testovanie

Pre testovanie funkčnosti prekladača sme navrhli nasledujúce testovacie scenáre, podľa ktorých bola overená funkčnosť prekladača.

Tabuľka 8 - Akceptačný test prekladu slova zangličtiny do slovenčiny

Meno	Preloženie slova z angličtiny do slovenčiny		**ID-Testu**	02-01-01
Rozhranie	Stránka prekladača		**ID-UC**	01
Účel	Preloženie anglického textu			
Vstupné podmienky		Žiadne		
Výstupné podmienky		Žiadne		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**	
1	Zadanie textu pre preklad slova „pen“	Zobrazenie zadaného textu	Zobrazenie zadaného textu	
2	Zvolenie jazyka „ENG-SVK“ z vyberača jazykov	Zobrazenie správnej hodnoty vyberača „ENG-SVK“	Zobrazenie správnej hodnoty vyberača „ENG-SVK“	
3	Stlačenie tlačidla „prelož“	Zobrazenie preloženého textu „pero“ a ďalších významov	Preloženie textu a jeho zobrazenie v zodpovedajúcich poliach	

Tabuľka 9 - Akceptačný test preloženia vety z angličtiny do slovenčiny

Meno	Preloženie vety z angličtiny do slovenčiny		**ID-Testu**	02-01-02
Rozhranie	Stránka prekladača		**ID-UC**	01
Účel	Preloženie anglického textu			
Vstupné podmienky		Žiadne		
Výstupné podmienky		Žiadne		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**	
1	Zadanie textu pre preklad vety „itwascoincidence“	Zobrazenie zadaného textu	Zobrazenie zadaného textu	
2	Zvolenie jazyka „ENG-SVK“ z vyberača jazykov	Zobrazenie správnej hodnoty vyberača „ENG-SVK“	Zobrazenie správnej hodnoty vyberača „ENG-SVK“	
3	Stlačenie tlačidla „prelož“	Zobrazenie preloženého textu „to bola náhoda“ a ďalších významov	Preloženie textu a jeho zobrazenie v zodpovedajúcich	

			poliach

5 Tretí šprint - radler

Tretí šprint s podtitulom víno bol zameraný na nasledujúce úlohy:

- Vytvorenie webovej služby typu REST
- Preklad hovorového jazyka s využitím filmových titulkov
- Vyhodnotenie úspešnosti prekladu
- Zrýchlenie vyhľadávania

5.1 Služba typu REST

Ako používateľ chcem mať preklad dostupný pomocou služby *typu REST*.

5.1.1 Analýza využitia služby typu REST

Postupom času sa ukázalo prvotné rozhodnutie použitia služby typu WSDL nesprávne. Služba WSDL je značne zložitá a jej volanie je náročné. Zákazník si teda vyžiadal modernejší a jednoduchší spôsob využívania služieb. Rozhodli sme sa v tomto prípade o využitie služby typu REST. Pri tomto type sa parametre požiadavky formulujú v adrese požiadavky a nie pomocou parametrov v samotnom tele požiadavky ako je to pri WSDL. Služba takisto odpovedá jednoduchým spôsobom kedy vracia výsledok do odpovede.

Pre služby typu REST má podporu aj náš webový server Tomcat a tak ich implementácia nebude značne komplikovaná. Implementácia tejto služby prebehne v programovacom jazyku Java.

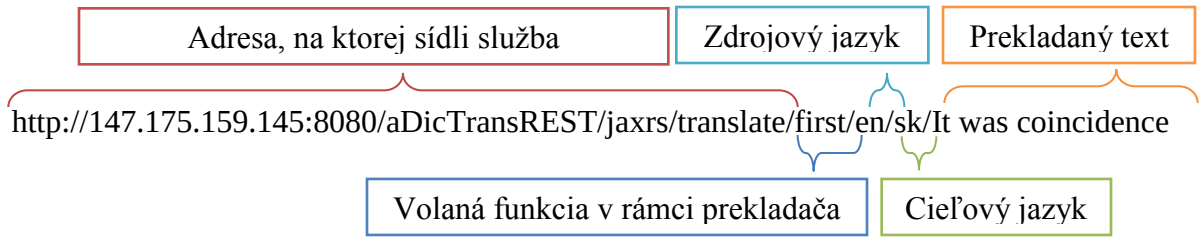
5.1.2 Návrh použitia služby typu REST

Službu typu REST sme navrhli, ako samostatnú triedu, ktorá bude odpovedať na adrese http://147.175.159.145:8080/aDicTransREST/jaxrs/translate. Služba v prvej fázy bude obsahovať dve základné funkcie a to získanie najpravdepodobnejšieho prekladu a získanie všetkých prekladov pre danú sekvenciu. Získanie najpravdepodobnejšieho prekladu je možné zavolať pomocou dodania */first* do adresy služby REST-u. Získanie všetkých je možné zavolať pomocou dodania */all* do adresy služby REST.

Pre určenie zdrojového a cieľového jazyka sa využívajú ďalšie dva parametre oddelené /. Prvý parameter definuje *zdrojový jazyk* a ten druhý *cieľový jazyk*. Momentálne sú dostupné jazyky angličtina pod premennou *en* a slovenčina pod premennou *sk*.

V prípade funkcie vracajúcej najpravdepodobnejší preklad je výstupom čistý text. V prípade funkcie, ktorá vracia všetky možné preklady je výstupom JSON ktorý obsahuje pole *translates* s prekladmi.

Nasledujúca časť oddelená / je rezervovaná pre text určený na preklad. Štruktúru dopytu pre preklad vety je možné vidieť na príklade:



5.1.3 Implementácia služby typu REST

Ako bolo povedané už v analýze implementácia prebehla v prostredí Java a ako webový server bol využitý Tomcat. Boli vytvorené dve funkcie, ktoré získavajú prostredníctvom požiadavky typu GET parametre a podľa toho vykonávajú akciu.

Predpis funkcie s parametrami vyzerá takto:

```
    @GET
@Produces("text/plain; charset=utf-8")
@Path("first/{lang_from}/{lang_to}/{text}")
/**
     * ziskanieprvehoprekladupredany text zadavasazdrojovy a prekladovyjazyk
     * @param text -prekladany text
     * @paramlang_from- jazyk, z ktorehosapreklada
     * @paramlang_to- jazyk, do ktoréhosapreklada
     * @returnvracianajpravdepodbonejsiprekladaksadadohladat
     * @throwsException-výnimka v prípadeproblémov
     */
publicStringgetFirstTranslate(
             @PathParam("text") String text,
             @PathParam("lang_from") Stringlang_from,
             @PathParam("lang_to") Stringlang_to
      )
```

Prvý riadok definuje, na aký typ požiadavky funkcia odpovedá, nasleduje riadok, aký typ výstupu produkuje a nakoniec, ako sa majú získať parametre. Názvy parametrov sú pritom uzatvorené v zložených zátvorkách. Ďalšie riadky vystihujú bežný Javadoc. V hlavičke funkcie je okrem toho ešte zadefinované, aké parametre z cesty napĺňajú vstupné parametre funkcie.

5.2 Úspešnosť prekladov

Ako používateľ chcem porovnať preklady s inými službami.

5.2.1 Analýza

Výsledok prekladu, ktorý vráti nás webový prekladač, je nutné porovnať s výsledkami existujúcich prekladačov. Manuálne porovnávanie človekom (pozri príloha A, príloha B) je neefektívne. Absencia tohto automatického porovnávania vyústila do rozhodnutia vytvoriť rozhranie, ktoré používateľovi umožní pohodlne skontrolovať úspešnosti prekladov. Toto porovnávanie bude uskutočňované medzi vzorovým prekladom, našim riešením a službami, ktoré sú najčastejšie

používané. Nimi sú Google Translate a Bing. Tieto prekladače ponúkajú restovú službu, ktorá po autentifikácii používateľa, vráti výsledok prekladu. Náš prekladač používa dopytovú funkciu vytvorenú cez post požiadavky.

Vektorový model

Model slúžiaci na reprezentáciu textu prostredníctvom vektoru termov (slov), ktorý obsahuje informáciu o výskyte jednotlivých termov nazývanú váha. Poradie slov v texte je pri použití tejto reprezentácie nepodstatné. Pre výpočet váhy termu sa používajú rôzne prístupy, častým a jednoduchým je napríklad frekvencia výskytov v analyzovanom texte, ktorú zachytáva bag-of-words model. Predpokladom tohto modelu je, že častejšie sa vyskytujúce slová viac prispievajú k obsahu ako tie menej časté [1].

Podobnosť textov

Reprezentácia textu vektorovým modelom nám umožňuje použiť geometrické funkcie pre hľadanie ich podobností. Najpoužívanejšou metódou je kosínusová podobnosť. Meria sa pomocou výpočtu kosínusu z uhlu, ktorý tieto vektory zvierajú.

Vzorec pre výpočet kosínusovej podobnosti:

$$CosSim(D_A, D_B) = \cos(\theta) = \frac{A.B}{\| A \| . \| B \|} = \frac{\sum_{i=1}^{n} A_i . B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} . \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (1)$$

Výsledkom je hodnota v intervale -1 až po 1. Pričom -1 predstavuje úplne opačné vektory, 0 nezávislé a 1 úplne závislé vektory. Keďže však počet výskytov slova nemôže byť záporný, výsledky pre podobnosť textov ležia len v rozsahu 0 až 1 [2].

5.2.2 Návrh riešenia

Pre porovnávanie prekladov bolo navrhnuté grafické rozhranie (Obr. 18) obsahujúce časti „Prekladaný text“ slúžiacej na zadanie slova alebo skupiny slov, ktoré chceme preložiť, a „Vzorový preklad“, ktorý slúži ako základ pre porovnávanie. Do ďalších troch textových polí je natiahnutý preklad z daných služieb. Výsledky budú zobrazené v grafe. Tiež je potrebné zapracovať možnosť voľby viacerých jazykov prekladu.

Obr. 18 - Návrh rozhrania pre porovnávanie prekladov

Pri Google Translate bol identifikovaný nasledujúci problém: *pokus o získanie požadovaných výsledkov pomocou zaslania požiadavky bol neúspešný.* Preto bolo potrebné nájsť webovú stránku, ktorá tento Google Translate preklad poskytuje. Hľadaním bola objavená stránka http://translate.gharjistan.org/index.php. Preklad bol z nej jednoducho vyparsovaný.

Pre získanie podobnosti preložených textov a vzorového textu reprezentujeme texty vektorovým modelom a následne vypočítavame kosínusovú podobnosť dvojíc (preklad a vzorový preklad). Avšak jedna veta môže mať viacero správnych prekladov, a preto je nevyhnutné umožniť používateľovi vložiť aj alternatívne preklady. Do poľa so vzorovým prekladom je preto možné vkladať viacero správnych prekladov, s ktorými bude testovací algoritmus počítať. Jednotlivé alternatívne preklady je nutné oddeliť bodkočiarkou. V prípade, že používateľ zadá viacero správnych prekladov, výsledkov správnosti prekladu je najvyššia hodnota získaná pri porovnávaní prekladu s jednotlivými zadanými prekladmi.

5.2.3 Implementácia

Obr. 19 zobrazuje jednotlivé dopyty na služby, ktoré vracajú požadované preklady. Požiadavky sú realizované parametrami:

- **$text** – text, ktorý je nutné preložiť
- **$from** – jazyk, z ktorého sa prekladá
- **$to** – jazyk, do ktorého je uskutočňovaný preklad

Náš prekladač **aDicitIT** v aktuálnej verzii využíva tieto parametre iným spôsobom a to tak, že ich mapuje ako *$_POST* požiadavky priamo z textových polí. Po mapovaní sa zavolá funkcia *input(),* ktorá vráti preklad textu.

Pre získanie **Google Translate** prekladu je nutné použiť volanie funkcie *get_google()*. Následne je podhodená požiadavka inej webovej stránke, ktorá vráti v textových poliach potrebný preklad. Ten sa je extrahovaný a priradený do premennej v našom algoritme.

V prípade **Bing** ide o autentifikácia kľúčom, ktorý je zadaná ako parameter objektu BingTranslateWrapper. Nad ním sa následne zavolá funkcia *translate()*.

```
/* aDictIT - start */
require_once ( './TranslateClasses/class.translate.php' );
$_SESSION['adictit']=input();
/* aDictIT - end */

/* Google translation - start */
require_once('./TranslateClasses/google.php');
$_SESSION['google']= get_google($from,$to,$text);
/* Google translation - end */

/* Bing translation - start */
require_once('./TranslateClasses/BingTranslate.class.php');
$gt = new BingTranslateWrapper('251FD8EA1120A12CF9560FF86D1A2231149350B7');
$_SESSION['bing'] = $gt->translate($text, $from, $to);
/* Bing translation - end */
```

Obr. 19 - Získanie prekladov aDictIT, Google a Bing

Potom, čo máme zadaný vzorový preklad a získané všetky preklady prostredníctvom nášho a konkurenčných riešení, vypočítame úspešnosť jednotlivých prekladov. Prvým krokom je odstránenie interpunkčných znamienok z koncom viet a nadbytočných medzier. Následne sa pre každú dvojicu prekladu a vzorového prekladu vypočíta kosínusová podobnosť. Posledným krokom je vyobrazenie vypočítaných hodnôt prostredníctvom stĺpcového diagramu používateľovi.

Jadrom výpočtu podobnosti prekladu a vzorového prekladu je funkcia getCosineSimilarity(), ktorú môžete vidieť na Obrázku 3.

```php
function getCosineSimilarity($tokensA, $tokensB){
    $a = $b = $c = 0;
    $uniqueTokensA = $uniqueTokensB = array();
    $uniqueMergedTokens = array_unique(array_merge($tokensA, $tokensB));
    foreach ($tokensA as $token) $uniqueTokensA[$token] = 0;
    foreach ($tokensB as $token) $uniqueTokensB[$token] = 0;
    foreach ($uniqueMergedTokens as $token) {
        $x = isset($uniqueTokensA[$token]) ? 1 : 0;
        $y = isset($uniqueTokensB[$token]) ? 1 : 0;
        $a += $x * $y;
        $b += $x;
        $c += $y;
    }
    return $b * $c != 0 ? $a / sqrt($b * $c) : 0;
}
```

Obr. 20 - Výpočet kosínusovej podobnosti.

5.2.4 Testovanie

Získanie prekladov

Pre overenie úspešnosti mapovania prekladov sme použili nasledovný testovací scenár, podľa ktorého sme otestovali funkčnosť implementácie.

Tabuľka 10 - Test mapovania prekladov

Názov	Mapovanie prekladov	ID Testu	03-01
Rozhranie	translate_checker/translators.php	ID UC	03
Účel	Overenie úspešnosti mapovania prekladov		
Vstupné podmienky	Zadaný text na preklad, vzorový text a vybraté jazyky prekladu		
Výstupné podmienky	Používateľ vidí na obrazovke preložené výrazy a graf úspešnosti		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**
1.	Získanie prekladov	Zobrazenie výsledkov prekladu v troch textových poliach a grafu úspešnosti prekladu	Výsledky prekladu sa zobrazili v požadovaných HTML elementoch. Graf bol vykreslený na určenom mieste.

Testovanie úspešnosti

Úspešnosť prekladu bola overená na piatich textoch, ktoré sú uvedené v nasledujúcej tabuľke.

Tabuľka 11: Overenie úspešnosti prekladačov.

Text	Úspešnosť prekladu		
	aDictIT	Google Translate	Bing
a big bug	50 %	100 %	100 %
a black pig	0 %	50 %	0 %

it's raining	20 %	100 %	100 %
go out of home	0 %	100 %	72 %
it was coincidence	100 %	100 %	67 %

5.3 Titulky

Ako používateľ chcem vedieť prekladať vety ktoré sa nachádzajú v hovorovom jazyku(dialógoch).

5.3.1 Analýza

Používateľ prekladača chce mať možnosť prekladať aj dialógy, ktoré sa bežne používajú v hovorovej reči. Tieto hovorové výrazy sa nenachádzajú v korpuse SME článkov. Preto sme sa korpus rozhodli doplniť vetami z filmových titulkov. Filmové titulky, filmov rôznych žánrov, obsahujú hovorové vety, ktoré vhodným spôsobom doplnia náš zdroj viet v databáze.

Analýza možností konvertovania

Pre potreby spoľahlivého vyfiltrovania iba slovenských titulkov, bude nevyhnutné stiahnuté súbory previesť do jednotného kódovania UTF-8.

Analyzujeme viacero možností konvertovania súborov titulkov s cieľom vybrať ten najjednoduchší, najrýchlejší a najspoľahlivejší spôsob.

Pomocou vytvoreného PHP skriptu

Konverziu titulkových súborov možno vykonať pomocou skriptu zapísaného v jazyku PHP pomocou nasledovných krokov:

- V hlavičke súboru si zadefinujeme typ kódovania, ktorý chceme v súbore používať
- Nasledujúce operácie sú vykonávané v cykle, ktorý sa ukončí až keď budú spracované všetky súbory v adresári
- Obsah súboru sa načíta do premennej v kódovaní, ktoré je definované v hlavičke súboru
- Pomocou zabudovanej PHP funkcie mb_convert_encoding(premenná, ‘UTF-8‘) je obsah premennej skonvertovaný do kódovania UTF-8
- Následne je tento obsah zapísaný do iného adresára, ale s rovnakým názvom súboru.
- Po skonvertovaní všetkých súborov v zdrojovom adresári, program ukončí svoju činnosť

Problémom tohto riešenia je, že stiahnuté súbory sú v rôznych kódovaniach. Konvertovanie týmto spôsobom nespôsobí žiadny problém pri znakoch anglickej abecedy. Pri znakoch iných jazykov obsahujúcich diakritiku, však celkom znemožní ďalšie spracovanie textu, pretože nesprávne konvertuje špecifické znaky daného jazyka.

Pre potreby použitia správnej detekcie kódovania je možné použiť knižnicu v jazyku Java s názvom „juniversalchardet“. Knižnica dokáže na základe určitej vzorky bitov presne určiť kódovanie daného reťazca. Rovnakým spôsobom sa to snažia určiť aj moderné webové prehliadače, pokiaľ nie je v HTML hlavičke explicitne zadaný, ktorý encoding má prehliadač použiť na zobrazenie obsahu. Problém však je, že knižnica dokáže určiť iba najbežnejšie používané kódo-

vania. Množstvo titulkových súborov v adresári je napr. v kódovaní WINDOWS-1250, ktoré knižnica nedokáže rozlíšiť.

Ak by sme chceli pokračovať využívaním vytvoreného PHP skriptu, bolo by potrebné postupovať nasledovne:

- Po skonvertovaní všetkých súborov do UTF-8 odfiltrovať všetky súbory, ktoré sú v jazyku, ktorý nevyužíva diakritiku (napr. angličtina). Tieto súbory by boli týmto spôsobom konvertované bez akýchkoľvek problémov.
- Tieto súbory následne odstrániť zo zdrojového adresára
- Pokračovať v konvertovaní pre jednotlivé kódovania aj pre ostatné jazyky a po spoľahlivom určení slovenského jazyka, odstrániť tieto titulky zo zdrojového adresára
- Spoľahlivo určené titulky v slovenskom jazyku uchovávať vo vytvorenom cieľovom adresári

Analýza nástroja Kaboom

Nástroj Kaboom od spoločnosti Sisulizer je voľne dostupný nástroj, ktorý dokáže skonvertovať vybraný súbor do požadovaného kódovania. Pre jeho správne fungovanie je však potrebné manuálne zadať kódovanie súboru, ktorý chceme konvertovať. Celá konverzia sa následne vykoná automaticky bez potreby akéhokoľvek programovania. Vzhľadom na fakt, že zdrojové súbory majú rozličné kódovania, konverzia týmto spôsobom nebude vykonaná vždy správne, so zachovaním potrebnej diakritiky. Nástroj nedokáže kódovanie daného súboru dedukovať samostatne.

Analýza nástroja UTF-8 konvertor

Nástroj UTF-8 konvertor je voľne dostupný na adrese: http://amos.911mb.com/download/UTF8_convert.php. Nástroj je schopný konvertovať súbory do formátu UTF-8 bez potreby explicitného definovania kódovania zdrojového súboru. Pred použitím nástoja sa odporúča si zdrojové súbory zálohovať. Nástroj následne iba skonvertuje pôvodné zdrojové súbory v adresári. Program nevytvára žiadne zápisy do registrov operačného systému a preto sa dá spúšťať aj z USB diskov.

Analýza zdrojov tituliek

Stiahnutie titulkov z webu

Mnoho webových stránok[2] ponúka možnosť sťahovania zo širokého výberu slovenských titulkov k rôznym filmom a seriálom.

Výhody:

- Veľká databáza slovenských titulkov.

[2] Napríklad www.titulky.sk, www.opensubtitles.org

Nevýhody:

- Rôzne obmedzenia pri sťahovaní titulkov, ktorých úlohou je zamedziť automatické sťahovanie.

Ďalším zdrojom slovenských titulkov sú amatérske webové stránky vytvárané prevažne obdivovateľmi konkrétnych filmov alebo seriálov. Takéto stránky väčšinou ponúkajú možnosť voľného sťahovania titulkov bez obmedzení.

Výhody tohto riešenia:

- Jednoduchý prístup k titulkom a možnosť automatického sťahovania.

Nevýhody riešenia:

- Malá databáza titulkov. Pre získanie väčšieho množstva titulkov je nutné použitie viacerých takýchto stránok.

Zdieľané titulky na intranetovej sieti združenia YNET

YNET je občianske združenie, ktoré ponúka služby v oblasti počítačových sietí a pripojenia k internetu pre viacero internátov v Bratislave. Členovia tohto združenia sú pripojený do intranetovej sieti, v ktorej medzi sebou zdieľajú veľké množstvo dát.

Výhody:

- Neobmedzený prístup k súborom.
- Dobré možnosti automatického sťahovania.
- Pomerne veľké množstvo titulkov.

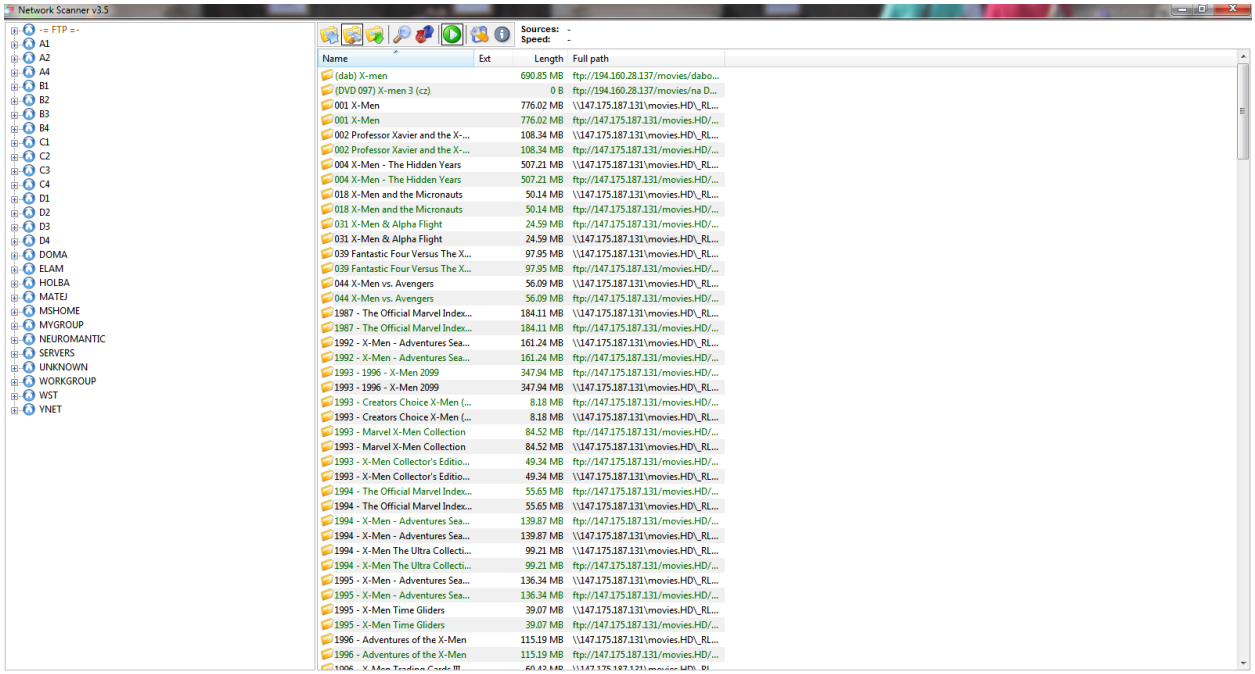
Nevýhody:

- Neexistencia kategorizácie súborov podľa ich obsahu.

Analýza nástrojov na stiahnutie titulkov

Network scanner

Tento nástroj umožňuje prehliadanie celej intranetovej siete YNET. Vyhľadáva súbory zdieľané pomocou protokolov SMB a FTP. Veľkou nevýhodou tohto nástroja je nedostatočná podpora pokročilého vyhľadávania. Network scanner síce umožňuje vyhľadanie súborov podľa „File extension“ avšak obsahuje podmienku, že musia byť zadané aspoň tri písmená pre vyhľadávanie. Táto skutočnosť veľmi výrazne zhoršuje možnosť automatického vyhľadávania a sťahovania slovenských titulkov.



Obr. 21 - Pracovná plocha vyhľadávača network scanner.

LASE 2

Program vyvíjaný v združení YNET. Jeho úlohou je nahradiť svojho predchodcu LASE, ktorý už nie je naďalej podporovaný a v dnešnej dobe už neposkytuje požadovanú funkcionalitu. Program je implementovaný v jazyku JAVA. Pôvodne vznikol ako študentská práca v predmete VPPJ. Autorom programu je študent Matúš Michalko. Program dokáže prehľadávať IP adresy v zadanom rozsahu a zisťuje či sa nezdieľajú súbory pomocou protokolov SMB a FTP. Veľkou výhodou je, že existuje možnosť prístupu k zdrojovým súborom tohto projektu.

5.3.2 Návrh

Získanie viet z obsahu slovenských titulkov pre obohatenie nášho korpusu si vyžaduje vysporiadať sa z viacerými problémami. Celý návrh procesu môžeme rozdeliť na štyri časti, ktoré bližšie opíšeme:

- Získanie titulkov z intranetovej siete.
- Úprava súborov do jednotnej znakovej sady.
- Vyfiltrovanie súborov na základe jazyka.
- Extrahovanie čistých viet zo súborov.

Získanie titulkov z intranetovej siete

Na základe analýzy sme sa rozhodli použiť ako zdroj titulkov intranetovú sieť združenia YNET, nakoľko viacero členov tímu má prístup k tejto sieti. Tiež bolo pre naše rozhodnutie dôležité, že je na tejto sieti zdieľané veľké množstvo filmov a seriálov. To dáva dobrý predpoklad na získanie veľkého počtu titulkov, ktoré na viac máme možnosť jednoducho získať. Nevýhodu, ktorou je neexistencia kategorizácie týchto súborov sa dá odstrániť pomocou použitého nástroja LASE

2, ktorý je schopný vyhľadať všetky súbory zdieľané na tejto sieti. Keďže máme prístup k zdrojovým súborom tohto programu nebude problém pomocou malých úprav rozšíriť ho tak aby dokázal sťahovať súbory na základe ich koncovky (pre nás sú dôležité koncovky .srt a .sub). Stiahnuté súbory sa umiestnia do jedného adresára kde budú ďalej spracované.

Úprava súborov do jednotnej znakovej sady

Na základe vykonanej analýzy možností konverzie titulkov navrhujeme titulkové súbory konvertovať nástrojom UTF-8 konvertor. Nástroj sa jednoducho používa a mal by previesť všetky titulky do kódovania UTF-8 bez ohľadu na ich pôvodné kódovanie. Ostatné dve analyzované riešenia by si vyžadovali ďalšie úsilie potrebné na eliminovanie faktu, že zdrojové súbory sa nachádzajú v rôznych kódovaniach.

Vyfiltrovanie súborov na základe jazyka

Keďže sme nemali možnosť vedieť v akom jazyku sú sťahované titulky, je nutné nejakým spôsobom automaticky rozoznať jazyk. Toto sa dá dosiahnuť jednoduchými pravidlami orientovanými na výskyt špecifických symbolov, ktoré sú bežné v rôznych jazykoch. Taktiež sa dá aplikovať zoznam stop slov podľa ktorých sa určí použitý jazyk. Aby sa predišlo falošne pozitívnym alebo falošne negatívnym prípadom je vhodné určiť nejaké tolerancie výskytu nežiadúcich znakov. Napríklad ak sa v anglických titulkoch nachádza slovenské alebo české meno môže sa stať, že program fungujúci na princípe sledovania výskytu písmen typických pre slovenský jazyk označí tieto titulky ako slovenské. Práve zavedenie tolerancie na počet výskytov týchto znakov sa dá tento problém odstrániť.

Extrahovanie čistých viet zo súborov

Pri takto spracovaných titulkoch je možné pristúpiť k poslednému kroku a to extrahovaniu zmysluplných viet, ktoré budú obohacovať náš korpus. Na základe analýzy štruktúry súborov typu .srt a .sub sa dá vytvoriť jednoduchý algoritmus, ktorý bude parsovať vložený text. Je jasné, že v takto získaných vetách môže byť ešte mnoho rôzneho balastu v podobe značiek alebo prebytočných znakov. Odstránenie tohto balastu je záležitosťou ďalšej úpravy korpusu.

5.3.3 Implementácia

Úprava programu LASE 2

Pre možnosť stiahnutia titulkov bolo potrebné do už existujúceho programu LASE 2 doplniť dve metódy, ktoré zabezpečili uloženie prehľadávaných súborov. Názov doplnených metód sú *saveSMBFile*, ktorá sťahuje súbory do vopred určeného umiestnenia a obdobne fungujúca druhá metóda *saveFTPFile*. Celkovo bolo z intranetovej siete stiahnutých 10134 súborov s koncovkou .srt alebo .sub.

Ukážka zdrojového kódu týchto metód:

```
   private void saveSMBFile(FileEntry collectedData, String path)
{
         BufferedInputStream download;
            BufferedOutputStream writer;
        try {
            int i;
            download = new BufferedInputStream(new SmbFileInputStream(collectedData.fileLocation));
           writer = new BufferedOutputStream (new FileOutputStream(new File(path+collectedData.fileName)));
            while((i=download.read())!=-1)
            {
                writer.write(i);
            }
           writer.close();
           download.close();

        } catch (IOException ex) {
            Logger.getLogger(DataStorage.class.getName()).log(Level.SEVERE, "mal by to byt access denied", ex);

        }
    }
```

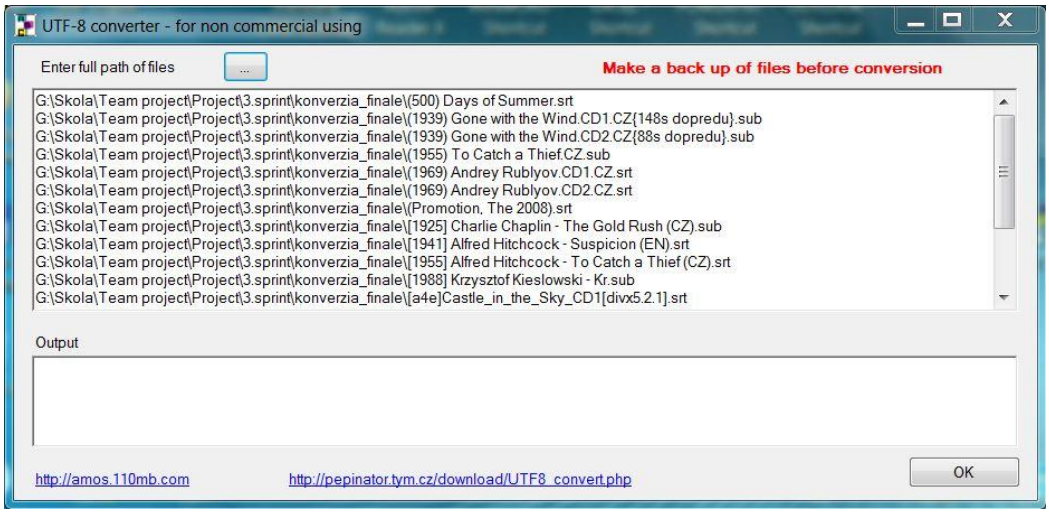
Obr. 22 - Zdrojový kód metódy saveSMBFile.

```
    private void saveFTPFile(FileEntry collectedData, String path, FTPClient client)
{
        int i;
    try {
        client = new FTPClient();
        client.connect(collectedData.getHostIP());
            client.login("anonymous", "lase@ynet.sk");


    String daco =collectedData.fileLocation.substring(6);
    StringTokenizer token = new StringTokenizer(daco,"/");
    daco = daco.substring(token.nextToken().length());

        BufferedInputStream download = new BufferedInputStream (client.retrieveFileStream(daco));
       BufferedOutputStream writer =new BufferedOutputStream(new FileOutputStream(new File(path+collectedData.fileName)));
        while((i=download.read())!=-1)
       {
           writer.write(i);
       }
        writer.flush();
        writer.close();
        download.close();
        client.disconnect();
    } catch (SocketException ex) {
        Logger.getLogger(DataStorage.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(DataStorage.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Obr. 23-Zdrojový kód metódy saveFTPFile

Konverzia titulkov

Na konverziu využívame voľne dostupný nástroj UTF-8 konvertor. Otvorením aplikácie sa nám zobrazí jednoduché okno Obr. 24..



Obr. 24 - Ukážka okna aplikácie UTF-8 konvertor.

Pred začatím konverzie je odporúčané vykonať zálohu konvertovaných súborov. Označíme súbory, ktoré si želáme konvertovať a stlačíme tlačidlo OK. Program nám zmení kódovanie vo všetkých označených súboroch na UTF-8 aj so zachovaním pôvodnej diakritiky.

Filtrovanie súborov na základe jazyka

Zmyslom tejto etapy bolo zistiť, ktoré súbory obsahujú slovenské titulky. Použitím Jednoduchého algoritmu na princípe výskytu špecifických znakov alebo stop slov bol schoný odstrániť väčšinu inojazyčných titulkov.

Celý proces filtrovania prebiehal vo viacerých etapách. V prvej etape sa za dôkaz, že sa jedná o slovenské titulky považovalo ak obsahovali znaky „á, ô, ľ“. Minimálna hranica počtu výskytu takýchto znakov bola určená na 50. Pre odfiltrovanie jazykov, ktoré používajú podobné znaky medzi ktoré patrí napríklad čeština, sa zaviedlo počítadlo nevhodných znakov. To malo tiež hranicu 50 a ak bola táto hranica dosiahnutá titulky boli považované za cudzojazyčné. Počítadlo reagovalo na znaky ako napríklad „ě, ř“ pre češtinu a „ö, ő“ pre maďarčinu. Ako sa zistilo, takéto podmienky dokázali veľmi dobre odfiltrovať maďarčinu, nemčinu a angličtinu. Čeština nebola úspešne odfiltrovaná nakoľko sa v tomto jazyku často vyskytuje písmeno „á“. Po odstránení tohto nedostatku už bola čeština spoľahlivo odstránená. Takto nastavený filter mal ale veľké problémy z jazykmi ako francúzština, španielčina a vietnamčina. Pre odstránenie týchto jazykov bolo použité filtrovanie na základe stop slov, pričom ako stop slová sa použili „alebo, jej“. Tento filter už bezpečne odstránil aj tieto jazyky. Na druhú stranu sa ale spätnou kontrolou zistilo, že spolu s cudzojazyčnými titulkami boli odstránené aj niektoré slovenské. Ich počet nebol vysoký (okolo 10 súborov) a boli späť zaradené medzi slovenské titulky. Nakoniec bola vykonaná manuálna kontrola získaných súborov. Z celkovo spomenutého počtu 10134 súborov bolo identifikovaných 733 slovenských titulkov. To je len niečo vyše 7 percent.

Ukážka použitého filtra pre slovenský jazyk:

```
private void applyPositiveRulesFilter(String text,String language)
{
    if(language.compareTo("svk")==0)
    {

        text =text.toLowerCase();
        if(text.matches(".*[Iô].*"))

        {
            positiveCounter++;
        }
        if(text.matches(".*[ŕěöő].*"))
        {
            negativeCounter++;
        }
    }
}
```

Obr. 25 - Zdrojový kód aplikovaného filtru

Extrahovanie viet zo súborov

Poslednou etapou procesu bolo získať vety, ktoré môžu byť použité na obohatenie korpusu. Na tento účel bol vytvorený jednoduchý parser, ktorý načítaval súbory po riadkoch. Keďže ako z analýzy vyplynulo, súbory s titulkami majú svoju presnú štruktúru dalo sa jednoduchým spôsobom implementovať riešenie. V prípade súborov s koncovkou .srt stačilo hľadať riadky v ktorých sa začínajú na ľubovoľné písmeno. V prípade .sub súborov bolo nutné odstraňovať z riadkov údaje v množinových zátvorkách predstavujúcich čas. Takto získané riadky zo súborov boli uložené do nových súborov, ktoré mali rovnaké pomenovanie ako tie originálne. Takto sme získali len vety, ktoré môžu byť doplnené do korpusu. Celkovo sa nám podarilo extrahovať okolo 650000 viet.

5.4 Zrýchlenie vyhľadávania

Ako používateľ chcem *znížiť čas odozvy* prekladača.

5.4.1 Analýza možností zrýchlenia

Keďže naše riešenie bude vyžadovať veľké množstvo vyhľadávaní v obrovskom množstve dokumentov potrebuje nájsť riešenie, ktoré nám prinesie takéto možnosti. Samotné MySQL, ktoré používame na uloženie dát obsahuje síce indexové vyhľadávanie v podobe tabuliek typu MyISPAN, toto vyhľadávanie však nie je dostatočne výkonné pre veľké množstvá dát a zväčšovaním tabuľky sa zväčšuje aj časová náročnosť vyhľadávania.

V oblasti indexového vyhľadávania sa databázy delia na dve množiny. Na množinu využívajúcu SQL syntax pre prácu s dátami a tie, ktoré sa snažia tejto syntaxi vyhnúť a pokúšajú sa využiť iné technológie ako je JSON alebo XML. Druhá skupina databáz sa označuje ako NoSQL databázy.

NoSQL databázy

Absolútnym základom pre dnešné NoSQL databázy so širokou podporou indexovania sa stal Lucene. Ide o projekt z dielne spoločnosti Apache, ktorý sa snaží o zrýchlené vyhľadávanie. Jedným z prvých nástrojov, ktorý využili jeho možnosti sa stal Solr. Jeho hlavnou nevýhodou je však to, že indexovanie neprebieha v reálnom čase a tak je pre náš projekt len veľmi slabo použiteľný.

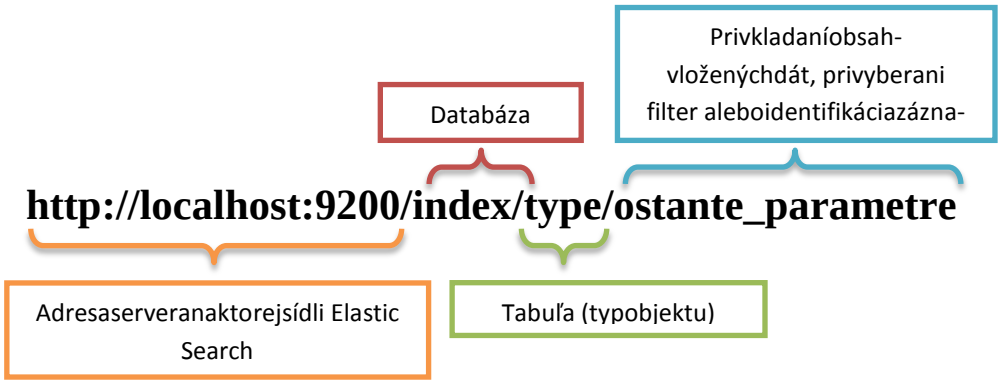
ElasticSearch

Tento vyhľadávací a indexovací nástroj je takisto, ako Solar založený na projekte ApacheLucene. Na rozdiel od Solar-u však prináša jednoduchšiu prácu s vkladaním a získavaním dát. Okrem toho ide o vyhľadávač, ktorý pracuje v reálnom čase a tak nenastávajú problémy pri pridávaní ďalších dokumentov.

ElasticSearch je naprogramovaný v programovacom jazyku Java, čo so sebou prináša nutnosť inštalácie Java virtuálneho stroja (JavaVirtualMaschine alebo JVM). Samotná inštalácia spočíva iba v rozbalení inštalačného balíčku na zvolené miesto. ElasticSearch funguje ako na operačných systémoch typu Unix tak aj na operačnom systéme Windows. Po samotnom spustení bude ElasticSearch počúvať na porte 9200 (poprípade na jemu nasledujúcich portoch, keďže umožňuje spustenie viacerých inštancií).

ElasticSearch pracuje na báze webových požiadaviek. Vie pracovať s webovými požiadavkami typu GET, POST, PUT a DELETE. Požiadavky typu PUT vkladajú dáta do databázy, požiadavky typu GET ich získavajú a pomocou požiadavky typu DELETE je možné mazať požiadavky.

Kostra požiadavky:



Volanie webových požiadaviek je možné spraviť napríklad pomocou programu curl. Tento program je v podobe objektu obsiahnutý aj v jazyku PHP ktorý čiastočne využívame pri našom rieše-

ní. ElasticSearch je schopní pracovať s požiadavkami vo formáte JSON. Týmto spôsobom vkladá dáta nastavuje vyhľadávacie podmienky, ale aj poskytuje výstupy.

Okrem tejto možnosti ponúka ElasticSearch aj knižnicu pre prácu s dátami pre programovací jazyk Java, v ktorom sme sa aj my rozhodli implementovať väčšiu časť nášho zadania.

Rýchlosťou a možnosťami filtrovania patrí ElasticSearch medzi tie najrýchlejšie riešenia. Zvláda spracovanie dát v rozsahu Petabytov a dokáže zrealizovať vyhľadanie približne za 20 až 30 milisekúnd. Počas práce nezaťažuje výrazne hardvér na ktorom beží.

ElasticSearch umožňuje písanie aj vlastných porovnávacích algoritmov, čo v neskorších fázach určite využijeme. Vďaka tomu budeme vedieť vyhľadávať nezávisle na slovné tvary slov tým že znížime hodnoty porovnávania písmen na konci slov.

Zaujímavou možnosťou systému ElasticSearch je možnosť nasadiť na systém Hadoop vďaka čomu je možné urobiť niekoľko nezávislých uzlov, z ktorých bude systém vedieť získavať dáta ešte rýchlejšie.

SQL databázy

PostgreSQL

PostgreSQL je objektovo-relačný databázový systém s otvoreným zdrojovým kódom. Počas pätnástich rokov vývoja získal silnú reputáciu za svoju spoľahlivosť, korektnosť a dátovú integritu. Striktne dodržiava štandardy (ANSI-SQL:2008 štandard) a poskytuje rozsiahlu funkcionality.Obsahuje tiež natívne rozhrania pre C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC. Je výborne zdokumentovaný.

Je možné ho nasadiť na všetky hlavné operačné systémy:

- *Linux*
- *Unix (Solaris, Mac OS)*
- *Windows*

Plne podporuje:

- *cudzie kľúče*
- *spojenia* (angl. Joins)
- *spúšťače* (angl. triggers)
- *pohľady* (angl. views)
- *vložené procedúry* – písané vo viacerých jazykoch (Java, Perl, Python, Ruby, Tcl, C/C++ a vo vlastnom PL/pgSQL)
- *vnorené dotazy* (angl. Subquery)

Podporované dátové typy:

- *SQL:2008* (INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL a TIMESTAMP)
- *Binárne velké objekty* (obrázky, hudba, video)
- *Vlastné definované typy*

Obsahuje množstvo zabudovaných funkcií napríklad pre základnú matematiku, operácie so slovami alebo kryptovanie. Pričom sú kompatibilné s Oracle.

Podporované funkcie:

- *viac-verziová kontrola zhody* (angl. Multi-VersionConcurrencyControl)
- *bod v čase obnovi* (angl. point in timerecovery)
- *tablespaces*
- *asynchrónna replikácia*
- *vnorené transakcie*
- *online zálohovanie*
- *prepracovaný plánovač/optimizér dotazov*
- *logovanie pred zápisov kvôli predchádzaniu chýb* (angl. writeaheadlogging)

Taktiež podporuje:

- *medzinárodné znakové sady*
- *viacbajtové kódovanie znakov*
- *unicode*
- *zotrieďovanie a formátovanie písmen na základe lokálnych pravidiel*

Vysoko škálovateľný čo sa týka množstva dát ktoré dokáže spracovať a počtu používateľov, ktorý s ním môžu súčasne pracovať. V nasledujúcej tabuľke (Tabuľka 12**)** sú zobrazené niektoré limutyPostgreSQL.

Tabuľka 12 - Limity PostgreSQL

Limit	Hodnota
Maximálna veľkosť databázy	Neohraničná
Maximálna veľkosť tabulky	32 TB
Maximálna veľkosť riadku v tabuľke	1.6 TB
Maximálna veľkosť stĺpca	1 GB
Maximálny počet riadkov v tabuľke	Neohraničný
Maximálny počet stĺpcov v tabuľke	250 - 1600 v závislosti od dátových typov
Maximálny počet indexov na tabuľku	Neohraničený

Indexy

Podporované typy indexov:

- *zmiešaný* (angl. compound)
- *jedinečný* (angl. unique)
- *čiastočný* (ang. partial)
- *funkčný* (angl. functional)

Typy úložných metód indexov:

- *B-strom*
- *R-strom*
- *Hash*
- *GiST* (skratka z *GeneralizedSearchTree*)

GiST

Indexovanie je zložitý systém, ktorý spolu spája množstvo odlišných zotrieďovacích a vyhľadávacích algoritmov vrátane B-stromov, B+-stromov, R-stromov, stromy čiastkových súčtov (angl. partialsumtrees), ohodnotených B+-stromov a mnoho ďalších. Gist ponúka flexibilitu pre definíciu vlastného spôsobu indexovania a teda toho čo chceme ukladať, ako to ukladať a ponúka možnosť pre vytvorenie nových spôsobov pre prehľadávanie uložených záznamov.

Gist slúžil ako základ pre mnoho verejnýcj projektov ktoré spolupracujú s PostgreSQL ako napríklad OpenFTS a PostGIS.

OpenFTS (skratka z OpenSourceFull Text Searchengine)

Je vyhladávací prístup založený na PostgreSQL a Gist, ktorý umožňuje on-line indexovanie dát a hodnotenie relevantnosti pre vyhľadávanie v databázach. Úzka integrácia s databázou umožňuje využitie metadát pre obmedzenie výsledkov vyhľadávania.

Základné informácie:

- Aktuálna verzia – 0.4 (Dec 4, 2009)
- Distribuovaný ako súbor Perl a TCL skriptov
- Poskytuje interface pre programovací jazyk Python
- Stále vo fáze vývoja, pričom vývoj verzií prebieha relatívne pomaly (verzia 0.32 – rok 2001)
- Aktuálne testovaný pre dátovú množinu 500 000 dokumentov, pričom každý z nich obsahuje 1000-2000 slov

PostGIS

Je projekt ktorý pridáva do PostgreSQL podporu geografických objektov. Čím je umožnené to aby bolo PostgreSQL použiteľné ako priestorová databáza pre geografické informačné systémy

Sphinx

Je veľmi podobný projektu ElasticSearch. Jeho hlavnou výhodu je však to, že je ho možné využiť z vnútra MySQL databázy. Použitie je teda jednoduché stačí vytvoriť tabuľku typu Sphinx a on sa o indexovanie postará sám. Tým odpadá dodatočná réžia s iným spôsobom získavania dát. Jeho vyhľadávacie schopnosti sú porovnateľné so systémom ElasticSearch. Z nasledujúceho porovnania však vychádza ElasticSearch, ako rýchlejší vyhľadávač.

Porovnanie rýchlosti Sphinx, Solar a ElasticSearch

Pri porovnaní bolo v každom systéme naindexovaných 197 000 dokumentov. Rýchlosť vyhľadávania je relatívna pretože veľkú časť testu zabral výpis dát ale je predpoklad že tento výpis zabral pri každom systéme rovnaký čas a preto je výsledky možné porovnávať. Výsledky porovnania sa nachádzajú v tabuľke Tabuľka 13**.**

Tabuľka 13 - Rýchlosť vyhľadania a veľkosť indexov pre rôzne vyhľadávacie systémy

Vyhľadávací stroj	Veľkosť indexu (MB)	Čas vyhľadania(s)
ElasticSearch	512,9	0,146
Sphinx	131,5	0,182
Solar	330,4	0,193

Z porovnania je vidieť že vyhľadávací systém ElasticSearch síce vytvára najväčší index na druhej strane, ale jeho vyhľadanie je rýchlejšie, ako pri ostatných systémoch.

Vyhodnotenie

V ďalšej práci sa budeme využívať indexovací nástroj ElasticSearch hlavne pre jeho rýchlosť a rozšíriteľnosť. Pritom nám nebudú vadiť väčšie nároky na miesto na disku, ktoré potrebuje ElasticSearch pre svoje indexy. V jeho prospech okrem iného hovorí aj možnosť nasadenia na distribuovaný clasterHadoop, ku ktorému máme prístup a v ďalších fázach projektu ho určite využijeme. Takisto využijeme v ďalších fázach vývoja aj možnosti písania vlastných porovnávacích algoritmov.

Hibernate

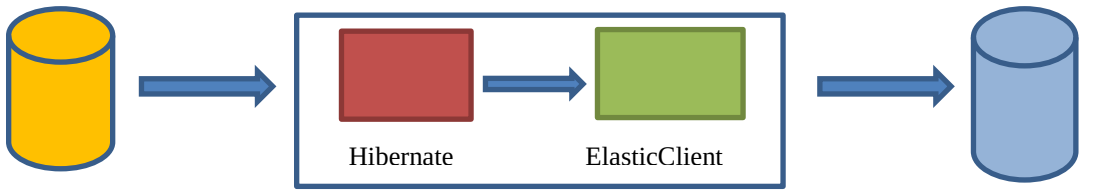
Pred možnosťou využitia ElasticSearch v našom prekladači je potrebné naplniť ho vetami z korpusu ktorý sa nachádza v MySQL databáze. Pre jednoduchšiu prácu s databázou sme si zvolili nástroj Hibernate.

Hibernate je framework pre jazyk Java, ktorý poskytuje možnosť mapovania objektového modelu domény na tradičnú relačnú databázu. Rieši problémy s použitím priameho prístupu k databáze pomocou funkcií vyššej úrovne manipulácie s objektom. Je to slobodný softvér distribuovaný pod GNU licenciou. Jeho primárnou funkciou je zobrazenie tried z Javy do databázových tabuliek (Java dátové typy na SQL dátové typy).Taktiež poskytuje zadávanie dotazov v jazyku HSQL (skratka z Hibernate SQL) slúžiace na získanie dát z databázy, ktoré sa mapujú do objektu triedy v Jave.

Pre definovanie mapovacích pravidiel sa používajú konfiguračné súbory v ktorých je zadefinované ktorý atribút v objekte triedy v Jave sa má mapovať na ktorý z tabuľky v databáze. Podporuje všetky rozšírené SQL databázové systémy.

5.4.2 Návrh nástroja na prenos údajov z MySQL do Elastic

Keďže pre zrýchlenie prekladu sme si z ponúkaných možnosti zvolili Elasticsearch, potrebujeme preniesť vety korpusu z MySQL databázy ktorú aktuálne používame práve do Elasticsearchu. Pre tento účel sme navrhli nastroj ktorý to dokáže.



MySQL

MySQL to ElasticConvertor

ElasticSearch

Nástroj pomocou Hibernate získa údaje z tabuľky Sentences z databázy sme_clanky a načíta ich do objektu rovnako pomenovanej triedy v Jave. Ten je ďalej posúvaný do elastic klienta ktorý z neho vytvorí JSON súbor, ktorý je vstupným dátovým typom Elasticsearchu a vloži ho doň.

5.4.3 Implementácia nástroja na prenos údajov z MySQL do Elasic

Nástroj je implementovaný v jazyku Java a spolupracuje s Hibernate frameworkom a klientom nástroja Elasticsearch. Princíp fungovania je jednoduchý a popísaný v časti návrh. Záznamy sa presúvajú po 1000-cových dávkach, aby sa predišlo zahlteniu Elasticu iba samotnou komunikáciou.

Konfigurácia Hibernate:

Pred použitím Hibernate je potrebné vytvoriť konfiguračné súbory, ktoré obsahujú základné nastavenia potrebné pre správne fungovanie frameworku.

Hibernate.cfg.xml - súbor obsahujúci informácie o zdrojovej databáze a odkazy na mapovacie súbory

```
<hibernate-configuration>
<session-factoryname="Factory">
<propertyname="hibernate.connection.driver_class">
com.mysql.jdbc.Driver
</property>
<propertyname="hibernate.connection.url">
jdbc:mysql://host/názov_databázy
</property>
<propertyname="hibernate.connection.username">meno</property>
<propertyname="hibernate.dialect">
org.hibernate.dialect.MySQLInnoDBDialect
</property>
<propertyname="hibernate.connection.password">heslo</property>
<mappingresource="sentences.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

Sentences.hbm.xml – konfiguračný súbor obsahujúci mapovanie atribútov z tabuľky MySQL na atribúty Java triedy

```
<hibernate-mapping>
<classname="sme_clanky.sentences" table="sentences">
<idname="id" type="int">
<columnname="ID"/>
<generatorclass="assigned"/>
</id>
<propertygenerated="never" lazy="false" name="sentence" type=
"java.lang.String">
<columnname="SENTENCE" sql-type="LONGVARCHAR"/>
</property>
  ...
</class>
</hibernate-mapping>
```

Zdrojový kód nástroja:

Inicializácia:

```
privatevoidsetUp() throwsException {
       sessionFactory = newConfiguration().configure().buildSessionFactory();
       node = nodeBuilder().node();
       client = node.client();
       }
```

Ukončenie:

```
privatevoidshutDown() {
              node.close();
       }
```

Metóda pre prenos údajov:

1. Zistenie počtu záznamov z MySQL databázy

```
QuerynumberQuery = session.createQuery( "selectcount(id) fromsentences" );
longcount = (long)numberQuery.list().get(0)/1000;
```

2. V cykle sa vykonáva získanie 1000 záznamov pomocou Hibernate

```
bulkRequest = client.prepareBulk();
intstartingColumn = i*1000;

Query q = session.createQuery( "fromsentences" );
q.setFirstResult(startingColumn);
q.setMaxResults(1000);

Listresult = q.list();
```

3. Pripravenie dávky

```
for ( sentencessentence : (List<sentences>) result )
{
bulkRequest.add(client.prepareIndex("sentences", "slovak", sentence.getId()
.setSource(jsonBuilder()
.startObject()
.field("sentence", sentence.getSentence())
.field("id_article", sentence.getId_article())
.field("id_type_of_sentence", sentence.getId_type_of_sentence())
.field("sentence_in_basic_form", sentence.getSentence_in_basic_form())
.field("length", sentence.getLength())
.field("words", sentence.getWords())
.endObject()
 ));
}
```

4. Vloženie dávky do Elasticsearch

```
bulkResponse = bulkRequest.execute().actionGet();
```

6 Štvrtý šprint – pivo 12°

Šiesty šprint sme pomenovali tuzemák. Tento šprint bol zameraný na:

- Automatické testovanie prekladu
- Preklad čísel a vlastných mien

6.1 Automatický test

Ako používateľ chcem automaticky otestovať prekladač.

6.1.1 Analýza

Vývoj nášho prekladača neustále napreduje. Sledovať úspešnosti jeho prekladu je nutná podmienka ďalšieho pokroku. Ten je možné sledovať na vopred určenom texte. Po implementácii novej funkcionality sa spustí automatický test, ktorého výsledkom sú grafy úspešnosti jednotlivých prekladačov na odlišnej množine zadaného textu. Porovnaním výsledkov prechádzajúcej a novovytvorenej verzie dostaneme informáciu, či je vhodné novú verziu ďalej rozvíjať, opraviť alebo zahodiť.

6.1.2 Návrh riešenia

Navrhované riešenie automatického testera musí spĺňať požiadavku jednoduchého pridávania testov, ktoré bude nutné vkladať s pribúdajúcim pokrokom prekladača. Vytvorená štruktúra testov by mala obsahovať:

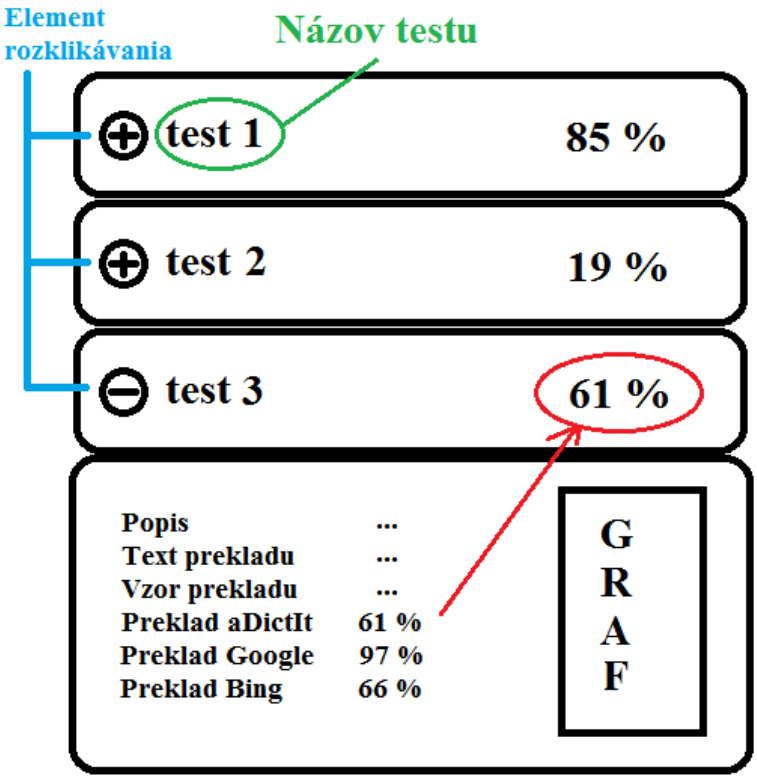
- **názov testu** – každý test by mal byť charakterizovaný slovom, prípadne skupinou slov
- **popis testu** – názov je vhodné rozviesť kratším popisom
- **text prekladu** – text, ktorý vstupuje ako parameter do prekladačov
- **vzorový preklad** – text, pomocou ktorého sa výsledky prekladačov budú porovnávať

Pridávané testy objektívne reprezentujú stupne náročnosti, ktoré prekladač musí zvládnuť. Medzi základné testy patria *slovo, skupina slov, veta a paragraf*[3].

Grafické prostredie

Výsledky testov nie je vhodné ihneď zobrazovať na obrazovke a preto bola zvolená metóda postupného zobrazovania rozbaľovaním. Navrhnuté grafické rozhranie popisuje Obr. 26.

[3] Náš prekladač zatiaľ neprekladá paragraf, keďže nami vytvorené rest služba takýto preklad ešte nepodporuje.



Obr. 26 - Návrh grafického rozhrania

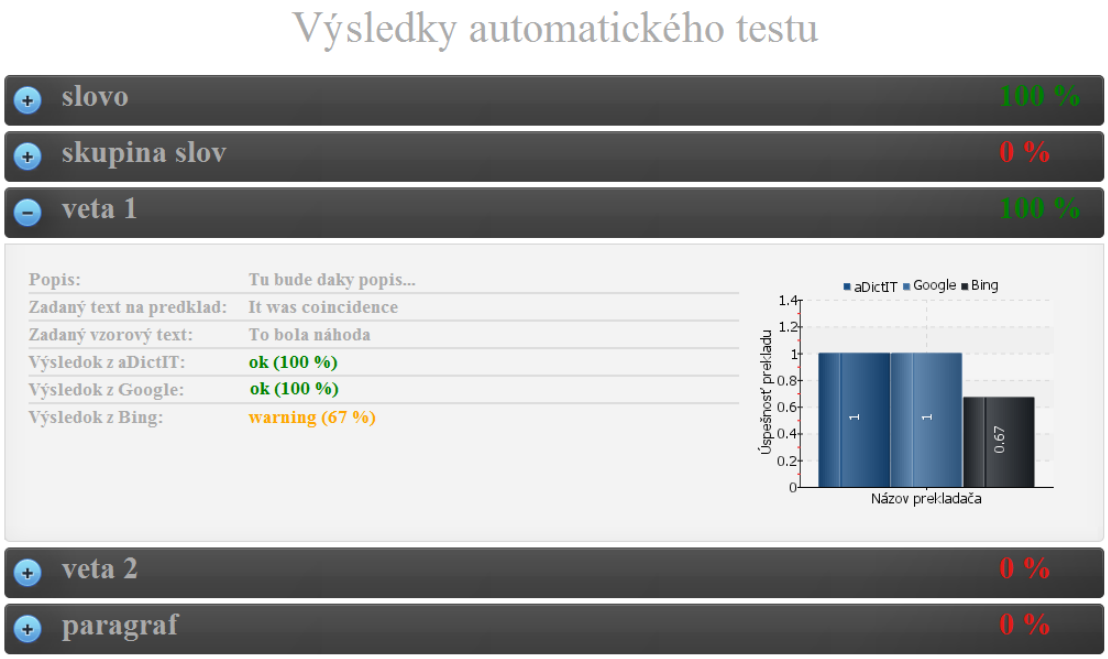
6.1.3 Implementácia

Štruktúra testov bola implementovaná pomocou dvojrozmerného poľa, ktoré opisuje Obr. 27. Pridávanie prvkov do pola je jednoduché a preto je prekladač kedykoľvek ľahko rozšíriteľný o nový druh testu.

```
$all_tests=array(
            array (
                    meno=>"slovo",
                    popis=>"Tu bude daky popis...",
                    text=>"spring",
                    vzor=>"jar"),
            array ( meno=>"skupina slov",
                    popis=>"Tu bude daky popis...",
                    text=>"severe winter",
                    vzor=>"krutá zima"),
            array ( meno=>"veta",
                    text=>"It was coincidence",
                    popis=>"Tu bude daky popis...",
                    vzor=>"To bola náhoda"),
            array ( meno=>"paragraf",
                    popis=>"Tu bude daky popis...",
                    text=>"At the age of 17 he had got his p
                    vzor=>"V 17 rokoch dostal súhlas rodičov
);
```

Obr. 27 - Štruktúra pre pridávanie testov

Grafické rozhranie obsahuje graf, ktorý je vykresľovaný externou knižnicou *pGraph*. Výsledky prekladačov do grafu boli zaslané pomocou get parametrov.



Obr. 28 - Grafické rozhranie automatického testera..

6.1.4 Testovanie

Prvky rozhrania

Pre overenie prvkov rozhrania sme použili nasledovný testovací scenár, podľa ktorého sme otestovali funkčnosť implementácie.

Tabuľka 14 - Test prvkov rozhrania

Názov	Prvky rozhrania	ID Testu	0x-0x
Rozhranie	translate_checker/automatic_test.php	ID UC	0x
Účel	Overenie rozhrania, rozbaľovanie jednotlivých prvkov		
Vstupné podmienky	Kliknutie na tlačidlo „Automatický test“		
Výstupné podmienky	Používateľ vidí na obrazovke preložené výrazy a graf úspešnosti, ktoré sú zabalené v štyroch testoch – slovo, skupina slov, veta, paragraf		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**
1.	Zobrazenie rozhrania	Zobrazenie štyroch zabalených výsledkov.	Výsledky boli zobrazené v požadovanom formáte – štyri rozklikávacie elementy.
2.	Kliknutie na element	Rozbalí sa test, v ktorom sú úspešnosti jednotlivých prekladov v percentách, a graf úspešnosti.	Rozbalenie testu bolo úspešné. Výsledky prekladov v percentách boli zobrazené na požadovanom mieste, graf úspešnosti odpovedal daným

			výsledkom.

6.2 Preklad čísel a vlastných mien

Ako používateľ chcem, aby sa čísla a vlastné mená prekladali správne (resp. neprekladali).

6.2.1 Analýza projektu Fact-Extractor

Tento projekt je prácou študenta FIIT. Funkcionalitou projektu by mala byť schopnosť vyextrahovať dôležité informácie (fakty) z textu. Takto nájdené fakty potom vhodným spôsobom prezentovať používateľovi.

Štruktúra projektu

Projekt je napísaný v *JAVE*. Projekt využíva jednu externú knižnicu s názvom *JSON.jar*. Zdrojový kód je rozdelený do troch balíkov:

- *Gazetteer* – nosná časť zdrojového kódu
- *Analyzer* – upravuje a dopĺňa implementovaný stemmer
- *Stemmer* – obsahuje triedu *stemmer*

Program pre jeho plnú funkčnosť vyžaduje ešte konfiguračný súbor, ktorý by mal obsahovať zoznam ďalších súborov, z ktorých by sa mal vytvoriť „slovník“. Prednastavene sa tieto súbory berú z adresára „*lists*“, ktorý musí byť umiestnený v adresáre projektu. Avšak program funguje aj bez týchto súborov. Rozdiel sa prejavuje v trochu obmedzenej množine vyextrahovaných faktov.

Projekt neobsahuje žiadnu dokumentáciu a komentáre k zdrojovému kódu.

Postup pri analyzovaní fungovania zdrojového kódu

Nakoľko nebolo možné sa oprieť o dokumentáciu alebo o komentáre, bolo nutné surové čítanie zdrojového kódu a sledovanie jeho vykonávania pomocou debugera. Takýmto spôsobom sme zistili nasledujúce zistenia.

Princíp fungovania

Srdcom celého projektu je metóda *search* nachádzajúca sa v triede *Gazetteer Obr. 27*.

```
public JSONArray search(String s) {
        annotations = new ArrayList<Annotation>();
        fromLists = new ArrayList<Annotation>();
        StringTokenizer st = getToken(s);
        while (st.hasMoreTokens()) {
                List<String> words = getListOfWords(new StringTokenizer(
                                st.nextToken()));
                lineAnalyze(words);
                lineSearch(words);

        }
        checkFirsts();
        findStarts(s);

        JSONArray jsonArray = new JSONArray(annotations);
        return jsonArray;
}
```

Obr. 29 - Zdrojový kód metódy search nachádzajúcej sa v triede Gazetteer.

Metóda na vstupe dostáva reťazec, ktorý predstavuje načítaný textový súbor, predstavujúci analyzovaný dokument. Tento súbor sa prednastavene vyžaduje aby bol v tvare „*file1.in*“ a nachádzal sa v adresáre projektu. Metóda má ako svoj výstup objekt *JSONArray*, v ktorom sú obsiahnuté všetky extrahované slová. Táto metóda tiež vytvára inštancie atribútov triedy *Gazetteer* ako sú *annotations* a *fromLists*.

Fungovanie metódy *search*

Načítaný reťazec sa pomocou *Tokenizeru* rozdelí na vety. Ako delimitery sa používajú prednastavene *„ ();:'\"?!-.*“. Takto vytvorené vety sa ešte raz rozdelia na slová. Takto získané slová sa vložia do zoznamu. Potom sa pre tento zoznam zavolajú metódy:

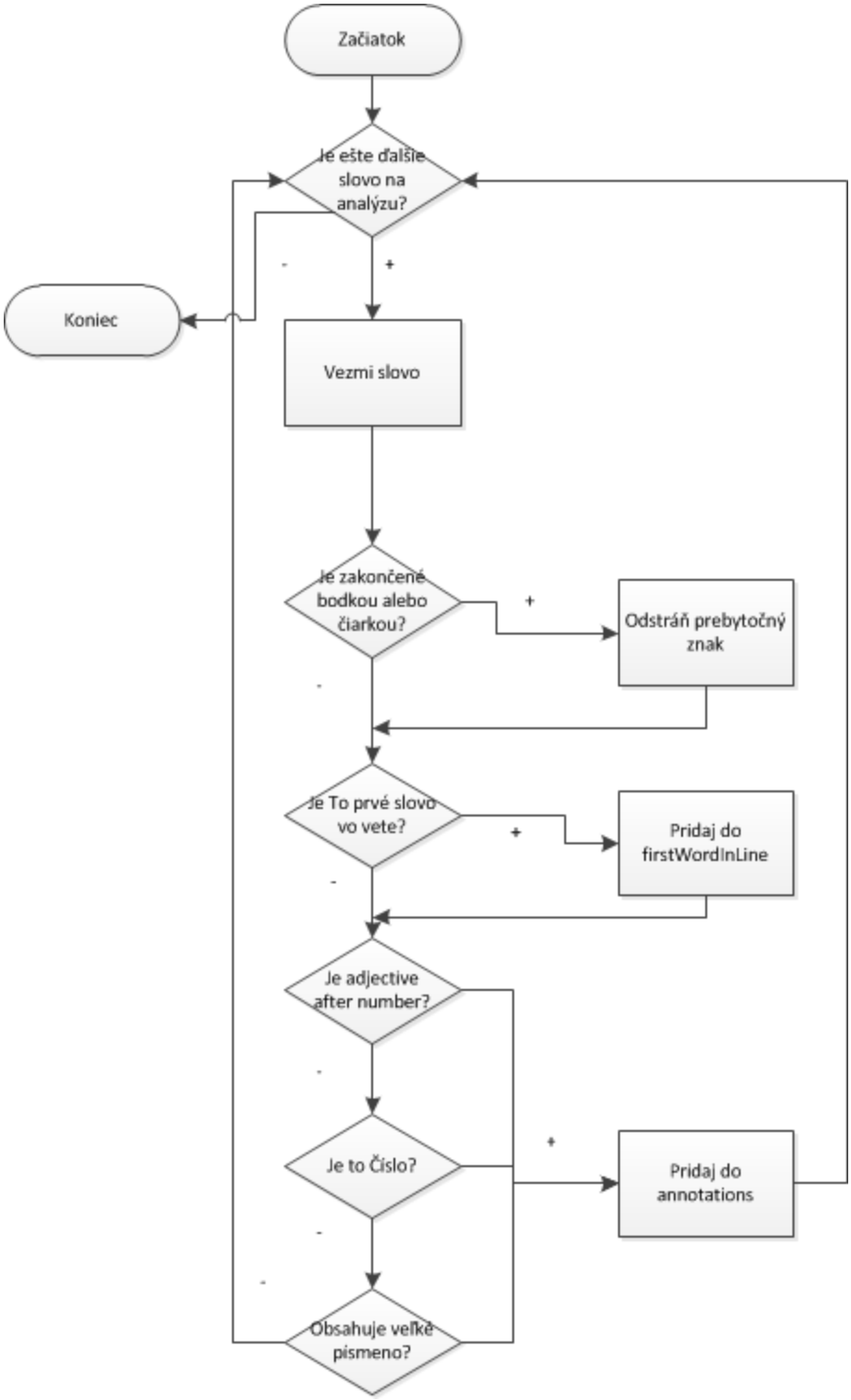
- *lineAnalyze()*
- *lineSearch()*

Obe metódy majú za úlohu vyextrahovať dôležité slová a tie vkladajú do už spomínaných atribútov triedy *Gazetteer*. Okrem týchto atribútov sa používa ešte aj *ArrayList firstWordInLine*.

Po prejdení všetkých slov v danom texte (reťazci) sa ďalej vykonjú metódy, ktoré majú za úlohu skontrolovať kompletnosť nájdených slov a prípadne ich doplniť. Tiež sa pre každé nájdené slovo zistí pozícia, na ktorej sa v danom reťazci nachádza. Nakoniec sa vytvorí objekt *JSONArray*, do ktorého sa tieto údaje vložia. Tento objekt je potom touto metódou vracaný.

Metóda lineAnalyze

Na vývojovom diagrame je vidieť priebeh vykonávania metódy lineAnalyze.



Obr. 30 - Priebeh vykonávania metódy lineAnalyze.

Z uvedeného vyplýva, že do zoznamu *annotations* sa vkladajú také slová, ktoré obsahujú veľké písmeno, nasledujú po nejakom čísle alebo samy sú číslom. Ak je slovo prvé vo vete tak sa pridá do iného zoznamu a to *firstWordInLine*. Vzhľadom na to, že takéto slovo je skonvertované na

formát s malými písmenami, už nie je možné, aby sa dostalo do *annotations* ak samo o sebe nie je číslom.

Metóda lineSearch

Táto metóda má za úlohu porovnať zoznam slov analyzovanej vety s obsahom *TreeMap dictionary*. *Dictionary* sa pri prvom použití naplní slovami alebo frázami zo súborov, ktoré boli uvedené v konfiguračnom súbore. Analyzované slovo sa porovná so záznamami v tejto štruktúre, pričom sa dá nájsť nielen jednotlivé slová ale aj frázy alebo vety. Takto nájdené zhody sú vložené do štruktúry *ArrayList fromLists*.

Metóda checkFirsts

Porovnáva záznamy v *ArrayListe annotations* so záznamami v *ArrayListe firstWordsInLine*. Ak sa nájde zhoda tak sa takýto záznam vloží do *ArrayListu Firsts*. Zmyslom tohto je aby sa našli anotácie, ktoré boli prv odfiltrované metódou *lineAnalyze* lebo boli na začiatku vety. Vtedy sa ešte nedalo vedieť či sú tieto záznamy anotácie alebo nie. Teraz keď sa už vytvoril zoznam anotácií to už možné je.

Metóda findStarts

Má za úlohu nájsť pre každý záznam *annotations*, *firsts* alebo *fromList* kde v pôvodnom reťazci, ktorý predstavuje analyzovaný súbor, začína. To jest na ktorej pozícii. Táto metóda tiež zhlukuje všetky relevantné záznamy uložené v *annotations*, *firsts*, *fromList* a takto vytvorený *Arraylist* sa zoradí. Položky sú zoradené podľa toho na ktorej pozícii v tom stringu začínajú vzostupne.

Výsledok

Nakoniec sa vytvorený *JSONArray*, vypíše na štandardný výstup. Nižšie je uvedený príklad výstupu. Analyzovaný text bol článok uverejnený na SME.sk, jeho témou je kríza v zdravotníctve.

[{"start":0,"what":"upperword","word":"**SLK**","end":0},{"start":0,"what":"upper
word","word":"**Milan**","end":0},{"start":161,"what":"upperword","word":"**Slovenskej**","end":1
71},{"start":201,"what":"upperword","word":"**Dragula**","end":208},{"start":234,"what":"upper
word","word":"**O**","end":235},{"start":466,"what":"upperword","word":"**Dragula**","end":473},{
"start":643,"what":"upperword","word":"**Ivan**","end":647},{"start":648,"what":"upperword","w
ord":"**Uhliarik**","end":656},{"start":762,"what":"upperword","word":"**ničPripomenul**","end":7
75},{"start":804,"what":"upperword","word":"**Slovensku**","end":813},{"start":884,"what":"uppe
rword","word":"**SLK**","end":887},{"start":913,"what":"upperword","word":"**Slovensku**","end":
922},{"start":1302,"what":"upperword","word":"**Dragulu**","end":1309},{"start":1315,"what":"u
pper-
word","word":"**Slovensko**","end":1324},{"start":1385,"what":"number","word":"**66**","end":1387
},{"start":1487,"what":"upperword","word":"**rukojemníciStav**","end":1502},{"start":1514,"wha
t":"number","word":"**1200**","end":1518},{"start":1542,"what":"upperword","word":"**Milana**","e
nd":1548},{"start":1549,"what":"upperword","word":"**Lopašovského**","end":1561},{"start":156
4,"what":"upperword","word":"**Asociácie**","end":1573},{"start":2382,"what":"upperword","wor

d":"**Pažitný**","end":2389},{"start":2413,"what":"upperword","word":"**pacientovMemorandum**","end":2432},{"start":2498,"what":"upperword","word":"**Peter**","end":2503},{"start":2504,"what":"upperword","word":"**Pažitný**","end":2511},{"start":2515,"what":"upperword","word":"**Stredoeurópske-ho**","end":2531},{"start":2592,"what":"number","word":"**30**","end":2594},{"start":2671,"what":"number","word":"**9**","end":2672},{"start":2879,"what":"number","word":"**100**","end":2882},{"start":2886,"what":"number","word":"**150**","end":2889},{"start":3199,"what":"upperword","word":"**Pažitného**","end":3208},{"start":3209,"what":"upperword","word":"**Slovensko**","end":3218}]

Trieda stemmer

Trieda stemmer slúži na zistenie základu slovenského slova. Je volaná v dvoch iných triedach programu a to konkrétne v triede SlovakWordAnalyzer a v triede SlovakStemmer.

Metóda stem

Metóda stem je hlavná metóda nachádzajúca s v triede stemmer. Metóda očistí skúmané slovo o príponu. Celá akcia sa vykoná volaním metódy zbavSaPripon, ktorá si pomáha inými metódami obsiahnutými v triede. Zoznam hľadaných prípon je definovaný v zozname.

Rozobrané sú určité príklady skupín slov, ktoré je potrebné ošetriť osobitne. Tieto slová pridávaním prípony menia svoj základ. Ako príklad si uveďme slovo Peter, ktoré sa pridaním prípony –ovi nezmení na slovo Peter-ovi, ale na slovo Petrovi. Program túto situáciu ošetruje spôsobom, že od slov končiacich na –er, najskôr odoberie koncovku –er a následne pridá iba písmeno –r. Takto dokáže spoľahlivo určiť, či dané slovo obsahuje príponu. Podobným spôsobom sú ošetrené aj ďalšie príklady, ktoré pridaním prípony čiastočne menia aj svoj základ slova.

Spôsob ako určiť, že napr. slová pes a psa alebo deň a dňa majú rovnaký základ slova ostáva stále otvorený.

Zhodnotenie funkcionality

Na základe preskúmania tohto projektu môžeme opísať základné funkcionálne vlastnosti. Program je schopný spoľahlivo rozoznať z analyzovaného textu čísla. Tiež spoľahlivo nájde slová, ktoré sú umiestnené vo vopred priloženom slovníku. Program mal niektoré problémy s formátovaním textu, pretože je veľmi citlivý na spôsob zápisu a oddeľovania viet. Preto je nutné niektoré nastavenia zmeniť pre možnosť použitia v našom projekte.

Máme za to, že tento zdrojový kód obsahuje hodnotné časti, ktoré by po malej úprave mohli byť v našom projekte užitočné.

7 Piaty šprint – slivka

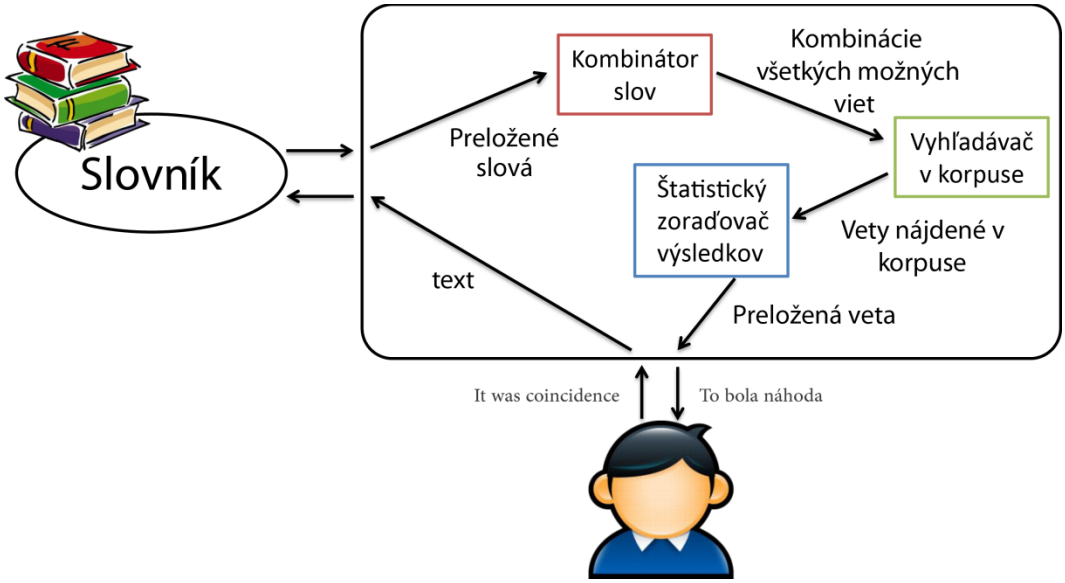
Ako používateľ chcem vedieť, ako dobre prekladá môj prekladač v porovnaní s konkurenciou.

7.1 Nová štruktúra prekladača

Chceme mať možnosť odčleniť jednotlivé časti prekladu

7.1.1 Analýza

Pôvodná štruktúra prekladača je zviazaná a nedovoľuje samostatné vyvíjanie jeho častí. Na nasledujúcom obrázku je zobrazený momentálny spôsob fungovania. Prekladaný text postupne prechádza cez jednotlivé komponenty, pričom ich fungovanie je pevne zviazané.



Obr. 31 - Schéma prekladu v aktuálnej verzii prekladača

Pre ďalší postup a prácu v tíme je preto nutné projekt rozčleniť.

7.1.2 Návrh

Počas vývoja prekladača sme identifikovali dve samostatné časti:

1. Prekladač viet
2. Vyhľadávač viet

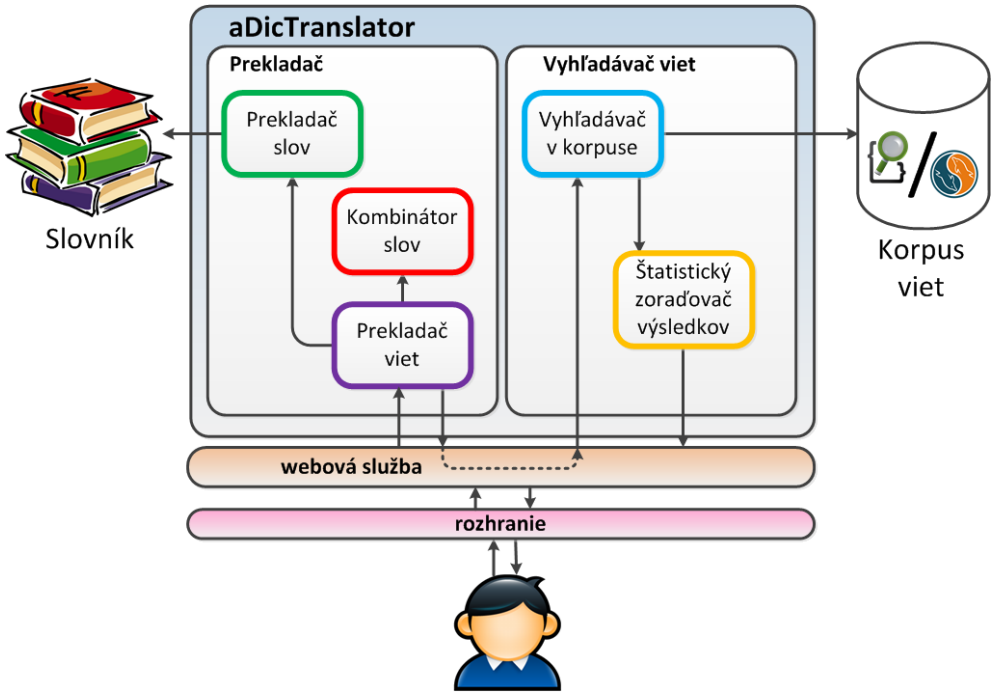
Tieto dve časti sú samostatné, a preto je ich možné dobre odčleniť. Odčlenenie je predprípravou pre ďalšiu úlohu, na ktorej budeme pracovať, a ktorou je vytváranie oddelených verzií týchto častí.

7.1.3 Implementácia

Pri implementácii sme preto vytvorili dve samostatné triedy - Translator a SentenceFinder. Tieto dve triedy na seba nijakým spôsobom nerefrencujú. SentenceFinder využíva preklady získané z Translator-a. Tie je však možné nahradiť iným generátorom viet.

Translator dostáva na svojom vstupe zadaný text, ktorý rozdelí na vety, neskôr na slová a vyhľadá v korpuse. V ďalšej fáze ich skombinuje a vytvorí všetky možné skupiny slov (teda vety).

Tieto putujú do SentenceFindera, ktorý sa ich snaží nájsť v korpuse a používateľovi vrátiť najsprávnejší preklad.



Obr. 32 - Schéma prekladu s odčinenými časťami prekladača a vyhľadávača viet

7.2 Extraktor faktov z korpusu viet

7.2.1 Návrh

V našom projekte sme sa na základe analýzy Fact-Extractora rozhodli vytvoriť vlastné pozmenené riešenie, ktoré by pokrývalo naše potreby. Hlavnou vlastnosťou nášho riešenia by mala byť schopnosť rozoznať čísla a fakty nachádzajúce sa v korpuse. Rozpoznané čísla a fakty potom vymeníme za príslušný tag. Takto vymeníme všetky relevantné termy za tagy v korpuse. Takéto riešenie by nám v konečnom dôsledku malo zlepšiť vyhľadávanie viet v korpuse.

Napríklad ak máme vetu:

„Pondelok 13.5.2010 bolo po výbuchu v Bagdade zranených 10 ľudí.“

Táto veta bude zmenená nasledovne:

„#=?+ #=? bolo po výbuchu v #=?+ zranených #=? ľudí.“

Takto bude ElasticSearch schopný hľadať relevantné preklady bez prihliadania na čísla a faktografické údaje. Vďaka tomu bude schopný objaviť s väčšou pravdepodobnosťou vhodný preklad v korpuse.

Aby sme mohli relevantné termy pretagovať musíme vhodným spôsobom identifikovať tieto termy. Ako navrhované riešenie pre nájdenie čísiel použijeme regulárne výrazy. Zároveň okrem základných čísloviek takto nájdeme aj radové číslovky, dátumy alebo výsledky športových zápasov. Tieto termy označíme dohodnutým tagom. Vzhľadom na to, že v korpuse sa nenachádza „#=?“, môže sa tento tag použiť ako identifikátor pre označenie všetkých čísiel. Tag „#=?+“ môže byť označenie pre fakty.

Pre objavovanie faktov je potrebný omnoho sofistikovanejší prístup. Analyzovaný Fact-Extractor použiť nemôžeme, nakoľko nemá pre nás dostatočné rozpoznanie relevantných faktov. Vedúci navrhol využiť práve vyvíjané riešenie jedného študenta, ktorý vyvíja podobné riešenie ako svoj bakalársky projekt.

7.2.2 Implementácia

Pre rôzne typy čísiel používame rôzne regulárne výrazy.

Pre nájdenie základných a radových čísloviek:

„([^a-z,A-Z][0-9]+((\\.|,)[0-9]+)?)|(^[0-9]+((\\.|,)[0-9]+)?)“

Dátum:

„((0[1-9]|[1-9]|[12][0-9]|3[01])[\\/\\-\\.])?((0[1-9]|1[012]|[1-9])[\\/\\-\\.])?([0-9]{1,4})?“

„((0[1-9]|1[012]|[1-9])[\\/\\-\\.])?((0[1-9]|[1-9]|[12][0-9]|3[01])[\\/\\-\\.])?([0-9]{1,4})?“

Športový zápas:

„[0-9]{1,3}[\\s]?:[\\s]?[0-9]{1,3}“

Na základe iných urgentnejších úloh sme sa rozhodli túto úlohu odložiť na neskôr a v implementácii pokračovať po vyriešení dôležitejších úloh. V rámci implementácie je ešte nutné použiť vytvorené regulárne výrazy a implementovať spomínané vyvíjané riešenie.

7.3 Skupiny slov v korpuse

Chceme vedieť preložiť aj slovné spojenia, ktorých preklad je jedno slovo a neprekladať ich po slovách

7.3.1 Analýza

Momentálne sa nám všetky slová prekladajú po slovách, čo nemusí byť vždy to najsprávnejšie. Napríklad slovné spojenie „*prime minister*“ sa do slovenčiny prekladá ako „*premiér*“. Náš prekladač však toto slovné spojenie prekladá po slovách a tak sa k prekladu *„premiér“* nikdy nedostaneme.

Tento stav by sme chceli zmeniť tak, aby náš prekladač hľadal aj takéto slová. Pričom tieto slová by mali mať vyššiu relevanciu ako samostatný preklad slov.

7.3.2 Návrh

Pri prekladaní viet je nutné do prekladača vložiť celý zvyšok vety, teda od aktuálne prekladaného slova až po koniec. Týmto spôsobom vieme získať nielen preklad daného slova, ale aj väčšieho slovného spojenia.

Údaj o tom, koľko slov nahradil daný preklad, je nutné uchovať tak, aby sa v nasledujúcich fázach spájania a generovania viet dalo povedať, ktorá veta je výhodnejšia a preto má byť vo výsledkoch vyššie umiestnená.

7.3.3 Implementácia

Túto metódu sme vo výsledku implementovali jednoduchým rozšírením SELECT príkazu, ktorý získava dáta zo slovníka.

```
SELECT DISTINCT
      W_TO.word AS translate,
      W_FROM.word AS source
FROM
      word W_FROM
      JOIN translation T ON W_FROM.id_word = T.id_word
      JOIN word W_TO ON W_TO.id_word = T.id_translation
WHERE
      T.id_language_to = <LangTo> AND
      W_FROM.word LIKE <WordToTranslate>% AND
      W_FROM.id_language = <LangFrom> AND
      <SentencePart> LIKE CONCAT(W_FROM.word,'%')
```

V tomto príkaze sa namiesto parametrov dosádzajú v prvej fáze jazyky, z ktorého(LangForm), a do ktorého(LangTo) sa prekladá text.

Ďalej sa do príkazu dosádza slovo, ktoré chceme preložiť (WordToTranslate) s tým, že na jeho koniec sa umiestňuje znak %, ktorý nahrádza akúkoľvek ďalšiu sentenciu slov. Na koniec sa do príkazu dosádza celá postupnosť slov v prekladanej vete (SentencePart) nasledujúca za aktuálne

prekladaným slovom vrátane tohto slova. To znamená, že sa vo výsledku vyhľadajú nielen slová, ktoré sú prekladom len jedného slova, ale aj celej sentencie, ktorá sa zhoduje so zadanou vetou.

7.4 Automatický tester

7.4.1 Analýza

V šprinte 3 sme opísali implementáciu automatického testera. Google však v decembri prestal poskytovať službu, ktorá nám zabezpečovala ich preklad. Preto je potrebné nájsť iné riešenie, ktorým by sme spomenutý preklad získali.
Existuje niekoľko riešení ako sa dopracovať k požadovaným informáciám. Jedným z nich je objavenie takej stránky, z ktorej by bolo možné tento prekladu extrahovať. Ďalšou možnosťou je odchytávať HTTP komunikáciu, v ktorej jej výsledok prekladu takisto uložený. Posledným riešením je nájsť spôsob, ako Google Translate prinútiť, aby nám tento výsledok vrátil sám.

7.4.2 Návrh

Z vyššie opísaných možností sme nakoniec využili posledné riešenie. Cez skrátené vyhľadávanie v prehľadávači sa nám podarilo nájsť takú adresu URL, ktorá vo svojom kontexte obsahovala požadovaný preklad. Následne bol vytvorený nový Google parser, ktorý odstránil zo získaného zdrojového kódu všetok nepotrebný text.

7.4.3 Implementácia

Implementovaný parser otvorí stránku Google Translate prostredníctvom URL adresy

*$url="http://translate.google.sk/?js=n&prev=_t&hl=".**$to**."&ie=UTF-8&layout=2&eotf=1&sl=".**$from**."&tl=".**$to**."&text=".**$text**."&file=#".**$from**."|".**$to**."|".**$text**." %0D%0A";*

Parameter **$to** definuje jazyk, do ktorého sa požadovaný text (**$text**) prekladá. Jazyk, z ktorého sa prekladá je opísaný parametrom **$from.**

V získanom zdrojovom kóde stránky je následne hľadaný podreťazec

*span title="'.**$what**.'" onmouseover="this.style.backgroundColor=\'#ebeff9\'" onmouseout="this.style.backgroundColor=\'#fff\'">*

kde parameter **$what** predstavuje upravený tvar vstupného textu **$text.**

7.4.4 Testovanie

Získanie prekladu z Google Translate

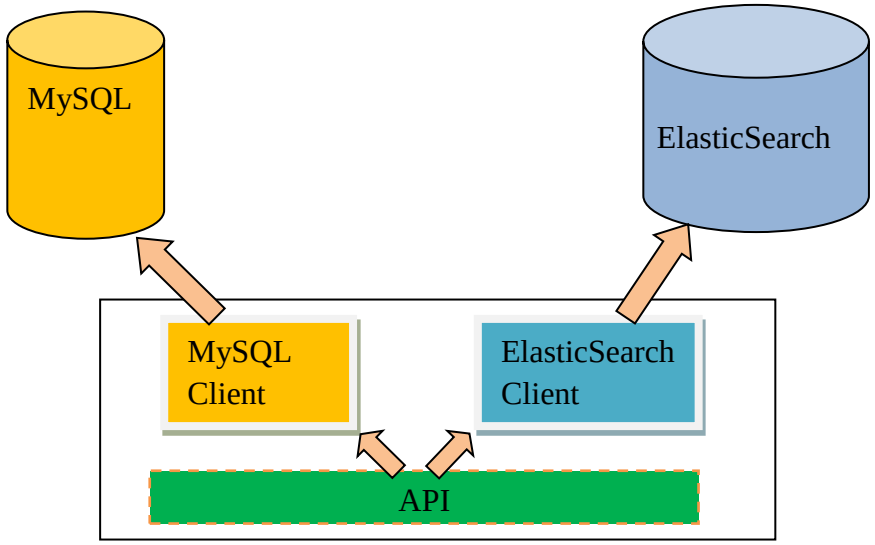
Tento testovací scenár overuje schopnosť funkcie *get_google($from,$to,$what)* získať požadovaný preklad.

Názov	Získanie prekladu z Google Translate	ID Testu	**07-01**

Rozhranie	translate_checker/TranslateClasses/google.php	ID UC	**0x**
Účel	Overenie správnej extrakcie prekladu zo zdrojového kódu		
Vstupné podmienky	Zadaný text na preklad, vstupný a výstupný jazyk		
Výstupné podmienky	Funkcia *get_google($from,$to,$what)*vráti rovnaký preklad ako je zobrazený na stránke translate.google.com		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**
1.	Zavolanie funkcie	Funkcia stiahne požadovanú webovú stránku Google Translate a extrahuje z nej požadovaný preklad.	Preklad z Google Translate bol korektne získaný a zobrazený na stránke automatického testera.

7.5 Návrh nástroja pre automatické pridávanie zmien do ElasticSearch a MySQL

Po tom ako sme preniesli korpus viet z MySQL do ElasticSearch vznikla požiadavka na jednotný nástroj pomocou ktorého by sme vedeli vykonávať zmeny v obidvoch spomínaných systémoch Obr. 4. Vďaka tomuto nástroju by sme nemuseli vykonávať zmeny v každom z nich samostatne a eliminovali by sme zbytočnú duplicitu práce.



Obr. 33 – Nástoj na pridávanie zmien do ElasticSearch

Nástroj poskytuje rozhranie, ktoré umožnuje vykonávať nasledujúce úlohy s oboma databázami:

- Pridávanie nových viet
- Úpravu existujúcich viet
- Zmazanie zvolených viet

7.5.1 Implementácia nástroja na pridávanie zmien

Nástroj je implementovaný v jazyku Java. Poskytuje rozhranie pomocou, ktorého je možné volať metódy na výkon úloh identifikovaných v časti návrh.

7.5.2 Zdrojový kód poskytovaného rozhrania nástroja

Pridávanie nových viet:

Vstup:

- Id prvého nového záznamu
- pole viet, ktoré chcete vložiť do databáz

Návratová hodnota:

- 0: podarilo sa vložiť všetky požadované vety
- -1: vety sa nepodarilo vložiť

```
public int insertNewSentences(int startingId,String[] newSentences) {
	}
```

Úprava existujúcich viet:

Vstup:

- Id záznamu, ktorý chcete zmeniť
- Požadovaná nová veta

Návratová hodnota:

- 0: podarilo sa zmeniť záznam
- -1: záznam sa nepodarilo zmeniť

```
public int updateSentence(int id,String sentence) {
	}
```

Zmazanie zvolených viet:

Vstup:

- Id záznamu, ktorý chcete zmazať

Návratová hodnota:

- 0: záznam bol zmazaný
- -1: záznam sa nepodarilo zmazať

```
public int deleteSentence(int id) {
	}
```

8 Šiesty šprint – mariška

Ako používateľ chcem mať možnosť si prekladač sám škálovať.

8.1 Konfigurovateľnosť prekladača

Chceme, aby bol prekladač jednoducho konfigurovateľný bez zmeny kódu služby

8.1.1 Analýza

Pri ladení nášho prekladača je nutné pracovať s konfiguráciou služby. Doteraz bolo nutné meniť tieto parametre priamo v zdrojovom kóde. Chceme tieto parametre teda presunúť do jedného konfiguračného súboru.

8.1.2 Návrh

Keďže náš prekladač je implementovaný v prostredí Java, využijeme na konfigurovanie Property súbory. Tieto súbory majú jednoduchú štruktúru:

```
Meno_premennej  hodnota
```

Property súbor bude v prekladači prístupný pomocou typu ConfigInfo. Tento obsahuje všetky aktuálne podporované premenné. Konfiguračný súbor sa bude získavať pomocou vzoru Singleton.

8.1.3 Implementácia

Načítavanie súboru je náročnejšie, pretože súbor chceme mať sprístupnený aj vo webovej službe, kedy java pracuje v inej časti súborového systému, ako by sme čakali.

```
private ConfigInfo() throws IOException
{
	final Properties prop = new Properties();

	try
	{
		prop.load(ConfigInfo.class.getResourceAsStream("/config.properties"));
	}
	catch (NullPointerException ex)
	{
			prop.load(new FileInputStream("config.properties"));
	}
	mySqlAddress  = prop.getProperty("MYSQL_ADDRESS");
	...
}
```

Po samotnom načítaní súboru sa inicializujú všetky potrebné parametre. Vzor SIngleton je implementovaný pomocou statickej metódy *instance*:

```
public static ConfigInfo instance() throws IOException
{
	if(instance == null)
```

```
        {
                instance = new ConfigInfo();
        }

        return instance;
}
```

Vďaka tomu sa súbor načítava vždy iba raz.

8.2 Pomocné funkcie webovej služby

Chceme mať prístupný zoznam všetkých verzií a aktuálne využívané premenné

8.2.1 Analýza

Keďže nás prekladač ma niekoľko rôznych verzií, je dobré ich mať popísané a zdokumentované. Tento popis môže byť prístupný aj verejne, keďže pri zadávaní prekladu je potrebné zadať verziu podľa jej čísla. Pri ladení je okrem toho dobré vedieť, s akými parametrami pracuje služba.

8.2.2 Návrh

Na tento účel chceme využiť ďalšie webové služby, ktoré budú prístupné našim zákazníkom.

Pre možné verzie bude dopyt vyzerať takto:

http://localhost:8080/aDicTranslator/support/versions

Výstup bude vo formáte JSON takto:

```
{
        "count":7,
        "versions":[{
                "id":1,
                "name":"vyhladajVMysql",
                "comment":"Verzia používa základný prekladač a vyhľadáva dáta ..."
                },
                ...
        ]
}
```

Pre získanie parametrov bude dopyt vyzerať takto:

http://localhost:8080/aDicTranslator/support/config

Výstup bude vo formáte JSON takto:

```
{
        "MYSQL_ADDRESS":"localhost",
        "MYSQL_USER":"root",
        ...
}
```

8.2.3 Implementácia

Pomocné servisy budú teda takisto realizované pomocou Jersy knižnice nasledovne:

```
@Path("support")
public class SupportService
{
	@GET
	@Produces("application/json; charset=UTF-8")
	@Path("versions")
	public String getVersions() throws JSONException
	{
		...
	}
}
```

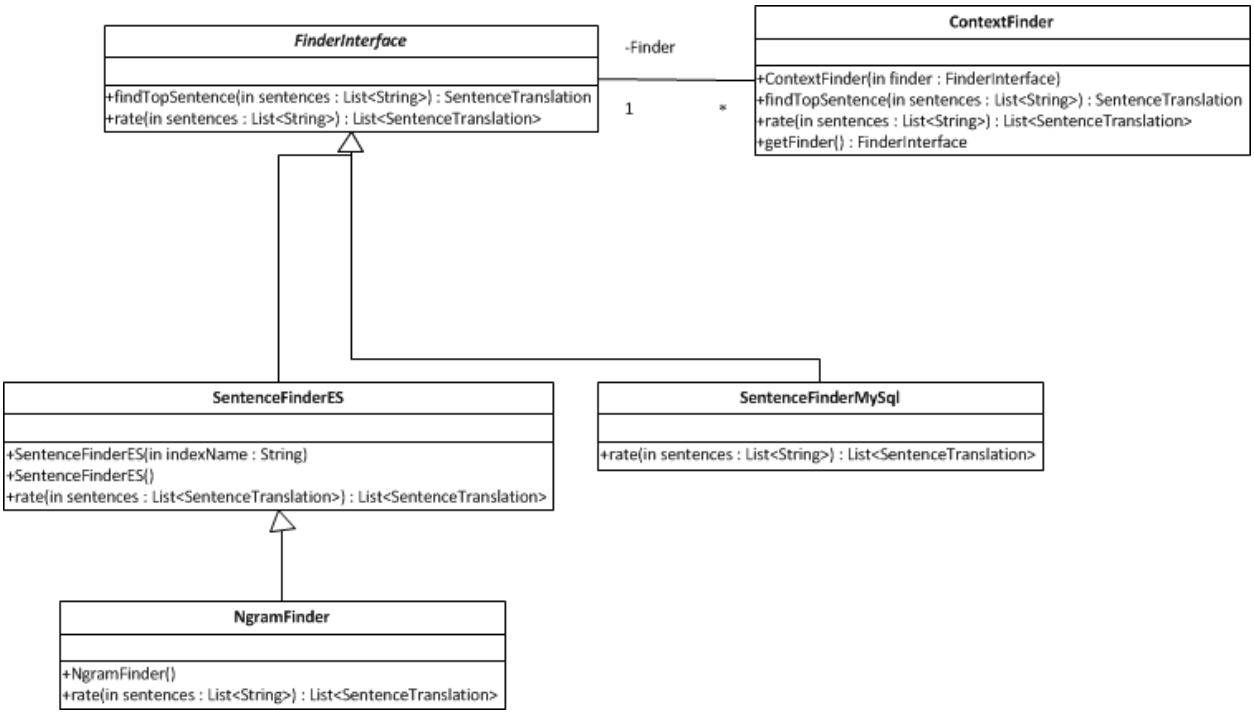
8.3 Vytvorenie Stratégií

8.3.1 Analýza

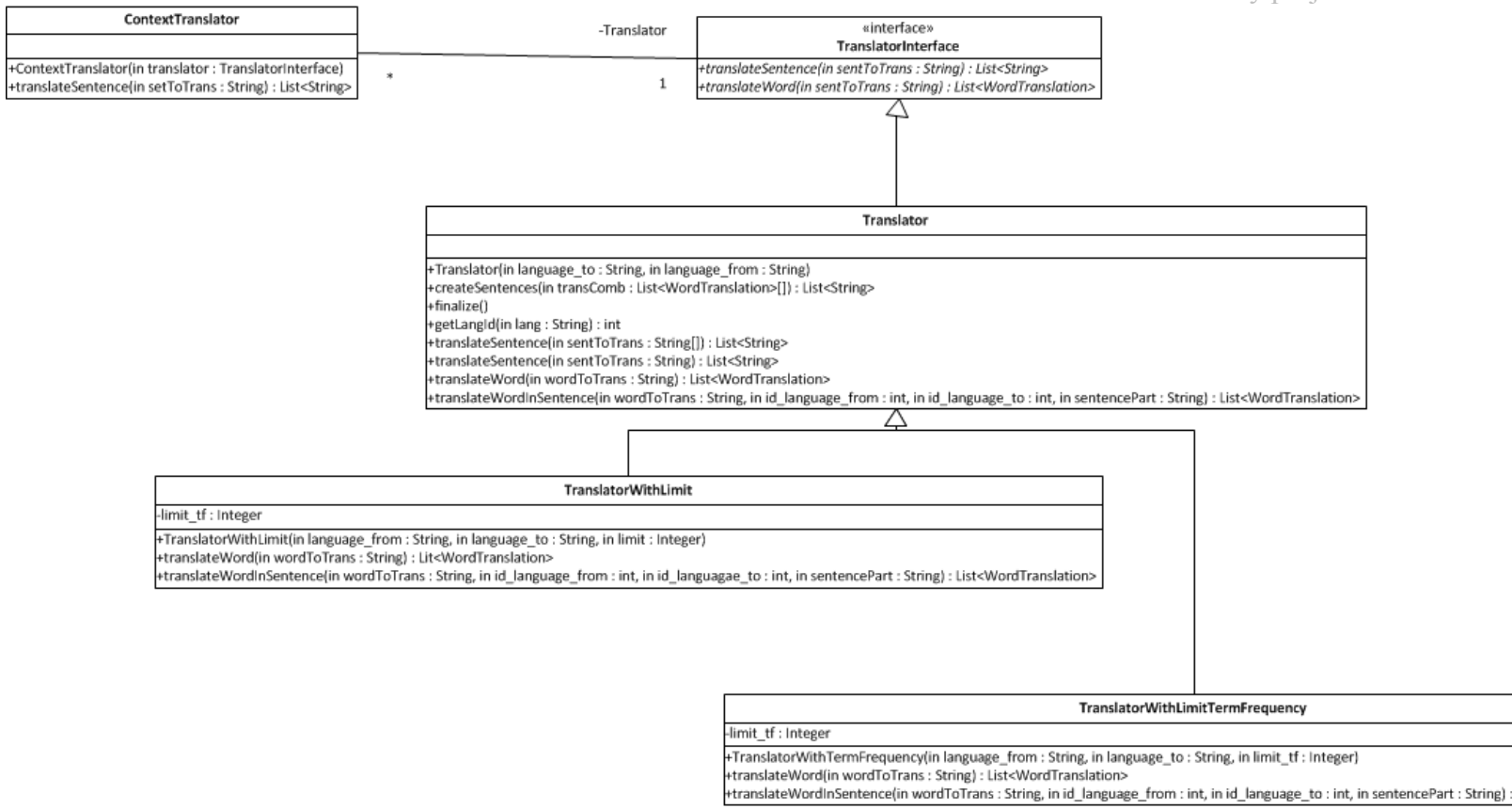
Vzhľadom na to, že vyvíjame viacero alternatív, ktoré chceme medzi sebou porovnávať a testovať je potrebné aby zdrojový kód RESTovej služby bol čo možno najviac nezávislý od kódu samotného prekladača. Tiež potrebujeme, aby jednotlivé časti prekladača boli čo najmenej ovplyvnené pri modifikácií alebo pridaní ďalšej časti a v konečnom dôsledku potrebujeme sprehľadniť náš zdrojový kód.

8.3.2 Návrh

Preto sme sa rozhodli využiť návrhový vzor Strategy, ktorý toto umožňuje. Implementovaním tohto vzoru sme schopný pridávať ďalšie verzie prekladača pričom to len minimálne ovplyvní ostatný kód.

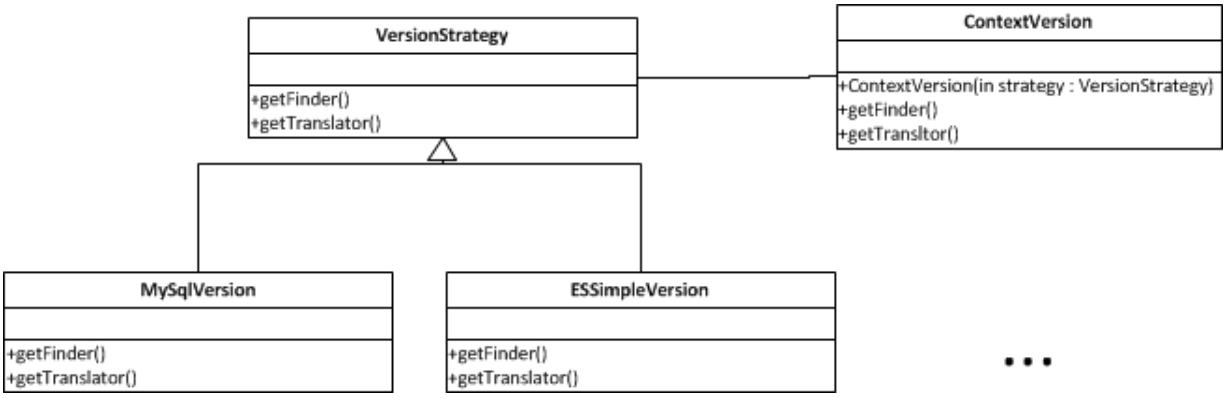


Obr. 34 - Diagram stratégie pre objekty súvisiace s vyhľadávaním viet



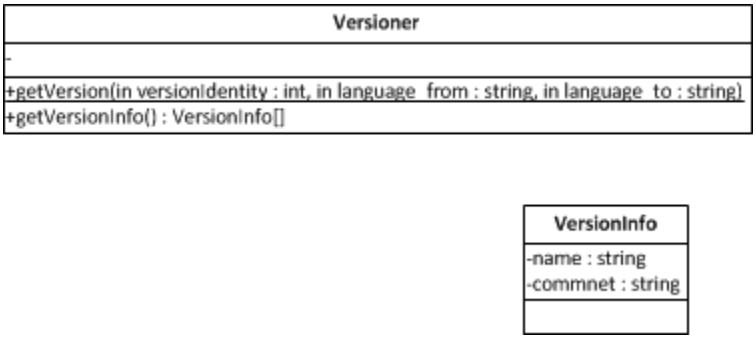
Obr. 35 - Diagram stratégie pre objekty súvisiace s prekladom textu

Následne sme vytvorili stratégiu verzionovania. Každá verzia má v tomto prípade priradený jeden Translator a jeden SentenceFinder podľa toho, aké činnosti chce realizovať.



Obr. 36 - Stratégia pre verzie prekladača

Okrem toho sme vytvorili objekt Verzioner, ktorý vytvára verziu podľa zadaného ID a takisto vie získať informácie o aktuálnych verziách ktoré sú uchovávané vďaka objektu VersionInfo.



Obr. 37 - Verziner pre získavanie verzie podľa zadaného ID a VersionInfo pre uchovávanie informácií o verziách

8.3.3 Implementácia

Pri implementácií bola umiestnená stratégia pre Prekladač do balíčka sk.stuba.fiit.adictit.translator. Stratégia pre SentenceFinder bola umiestnená v sk.stuba.fiit.adictit.translator.sentences.

Pre verzie bol vytvorený nový balíček: sk.stuba.fiit.adictit.utils.versioning. Výber verzií podľa ID je realizovaný nasledovne:

```
public static ContextVersion getVersion(final int versionIdent,final String langua-
ge_from,final String language_to) throws CannotCreateTranslatorException, CannotCrea-
teSenteceFinderException
	{
		ContextVersion version;

		switch (versionIdent)
```

```
            {
                  case 1:
                        version = new ContextVersion(new MySqlVer-
sion(language_from, language_to));
                        break;
                  case 3:
                        version = new ContextVersion(new ESVersionWithLi-
mit(language_from, language_to));
                        break;
                  case 4:

....
```

8.4 Optimalizácia služby typu REST

Chceme mať pohodlne prístupnú webovú službu so zadaním verzie prekladu

8.4.1 Analýza

Naša rest služba prijíma momentálne 4 parametre prekladu a to:

- Metóda výstupu (najlepší preklad, množina najlepších prekladov, všetky relevantné preklady)
- zdrojový jazyk
- cieľový jazyk
- text

štruktúra URL adresy:

```
http://adresaServera/aDicTranslator/metoda/jazykZ/jazykDo/prekladanyText
```

Všetky tieto parametre sú časťou URL adresy a tak nie je možné žiaden vynechať. Takisto dĺžka adresy je limitovaná, a tak by v prípade prekladu dlhšieho textu nastali problémy. Do adresy je preto nutné pridať nový parameter a to verziu prekladača a prerobiť štruktúru tak, aby bola viac prispôsobivá.

8.4.2 Návrh

V samotnej URL adrese dopytu na získanie prekladu zostanie už len metóda prekladu, ku ktorej sa pridá číslo verzie, ktorou budeme prekladať. Zdrojový, cieľový jazyk a samotný text sa presunú do formy parametrov. Volanie služby bude prístupné nielen HTTP metódou GET, ale aj POST pre prekladanie dlhších textov.

Nová štruktúra adresy bude vyzerať takto:

```
http://adresaServera/aDicTranslator/VERZIA/METODA?
      text=PREKLADANY_TEXT&
      from=JAZYKY_Z&
      to=JAZYK_DO
```

8.4.3 Implementácia

Služba typu REST bola na našom serveri momentálne realizovaná za pomoci knižnice WINKS. Tá nám však nedovoľovala pracovať s parametrami formulára, a tak sme boli nútení prejsť na inú knižnicu. Použili sme preto Jersy knižnicu.

Kód pre vytvorenie webovej služby s danou štruktúrou vyzerá takto:

```
@GET
@Produces("text/plain; charset=utf-8")
@Path("{version:\\d}/first")
public String getFirstTrasnalte(
            @PathParam("version")                  final  int    versionIdent,
            @QueryParam("text")                    final  String text,
            @QueryParam("from") @DefaultValue("en") final  String lang_from,
            @QueryParam("to")   @DefaultValue("sk") final  String lang_to)
      throws  CannotCreateTranslatorException,
            CannotCreateSenteceFinderException, CannotTranslateTextException
{
      ...
}
```

Tento kód zobrazuje metódu „first“, pričom definícia ostatných metód vyzerá analogicky. Ako bolo spomenuté vyššie, v URI (@Path) ostali zachované iba verzia a metóda, pričom verzia je definovaná číslom. Verzia je potom parameter funkcie, čo definuje štruktúra @PathParam(„version“). Ďalšie parametre sú získané z parametrov typu Query, teda z formulárových premenných.

9 Siedmy šprint – extáza

Ako používateľ chcem prekladať rýchlejšie a presnejšie.

9.1 Vyhľadávanie cez n-gramy

9.1.1 Analýza

Pri prekladaní vety spôsobom preloženia každého slova a vyhľadávaní všetkých viet, ktoré vznikli kombináciou týchto prekladov, vzniklo veľké množstvo dopytov na korpus v ElasticSearch. Taktiež táto metóda pre úspech predpokladala, že výsledná veta sa v korpuse bude nachádzať. Tieto dva problémy je možné vyriešiť tým, že sa prekladaná veta bude deliť na n-gramy (n-tice slov). Týmto spôsobom je možné vetu preložiť aj keď sa celá v korpuse nenachádza, ale nachádzajú sa v ňom jej jednotlivé časti.

Ako príklad majme vetu o dĺžke 6 slov, kde priemerný počet možných prekladov slova je 5 prekladov. Pri vytváraní všetkých potencionálnych viet vznikalo pri klasickom preklade teda 5^6 (15625) možných kombinácii. Veľký počet kombínácií predstavuje veľa požiadaviek na vyhľadávanie a tak spomaľuje rýchlosť celého prekladu. Pri použití n-gramov o dĺžke 3 dostaneme pre uvedený príklad len $4*5^3$ (500) kombinácií, čo predstavuje 31 násobne menšiu náročnosť.

Náročnosť doterajšieho prekladu sa dá vyjadriť vzorcom: x^y

Náročnosť prekladu využitým n-gramov vzorcom: $(x - n + 1) \times x^n$

Kde x je počet slov vo vete, y je priemerný počet prekladov jednotlivých slov a n je veľkosť n-gramu. Z uvedeného vyplýva náročnosť $O(x^Y)$ v porovnaní s $O(x^3)$ pri n-gramoch a teda efektivita n-gramov sa prejaví najmä pri dlhých vetách.

Nájdené n-gramy je následne nutné spojiť do jednej vety. Vhodným spôsobom je n-tice spájať na základe ich spoločného prieniku. Teda budú sa spájať n-gramy, ktoré majú po svojich krajoch spoločný prienik. Tým pádom je vyššia pravdepodobnosť, že výsledná veta bude zmysluplná a nebude to len zlepenec slov bez významu. Dosiahnuteľné to je úpravou hodnotenie, kde sa zvýhodnia vety, ktoré vznikli spojením n-gramov s väčším prienikom.

9.1.2 Návrh

Na základe skúseností s rýchlosťou existujúceho riešenia a zvážení minimálnej veľkosti n-gramu sme sa dohodli, že riešenie bude pracovať s trigramami. Táto hodnota však bude konfigurovateľná a bude ju možné v budúcnosti meniť.

Aby bolo možné prekladať vety prostredníctvom trigramov je nevyhnutné upraviť nie len dopyty na korpus, ale aj samotný korpus viet. Upravíme preto existujúci korpus, rozsekaním viet na vety o maximálnej dĺžke troch slov.

Nový proces prekladu bude pozostávať z piatich hlavných krokov:

1. Preloženie vety po slovách (získanie všetkých možných prekladov pre každé slovo).
2. Vyskladanie trigramov.
3. Vyhľadanie trigramov v korpuse.
4. Spájanie výsledkov z korpusu a vypočítanie skóre na základe skóre z SlasticSearch a prekryvu spájaných trigramov.
5. Zobrazenie výsledkov podla dosiahnutého skóre.

9.1.3 Implementácia

Základom pre implementáciu bolo rozdeliť vety v korpuse na n-gramy. Z hľadiska výkonu bolo pre nás najideálnejšie rozdeliť vety na trojice slov. K tomu sme použili nasledujúcu metódu.

```
private static ArrayList<String> generateNgrams(int n, String senten-
ce){
	String words[] = sentence.split(" ");
	String generatedSentence = null;
	ArrayList<String> sentGroup = new ArrayList<String>();

	for (int i=0; i<words.length-n+1; i++){
		generatedSentence = "";
		for (int j=0; j<n; j++){
			generatedSentence = generatedSentence.concat (words[i
			+ j]).concat(" ");
		}
		sentGroup.add(generatedSentence.trim());
	}
	return sentGroup;
}
```

Preklad viet cez n-gramy sme implementovali ako novú verziu prehliadača, pričom sme vychádzali z existujúcej verzie hľadajúcej v ElasticSearch.

```
public class NgramFinder extends SentenceFinderES{

	public NgramFinder() throws CannotCreateSenteceFinderException {
		super("shingleindex");
	}
```

Pre optimalizovanie výkonu sme najprv identifikovali všetky unikátne trigramy a dopyt na ElasticSearch sme vykonali pre všetky hromadne. Neskôr sme k prekladom pristupovali cez objekt triedy HashMap.

```
HashMap<String, List<SentenceTranslation>> hmap = new HashMap<String,
...
for(SentenceTranslation st: translations){
      if(hmap.containsKey(st.getOriginalTranslate())){
            hmap.get(st.getOriginalTranslate()).add(st);
      }
      else{
            List<SentenceTranslation> newlist = new Array-
            List<SentenceTranslation>();
      newlist.add(st);
      hmap.put(st.getOriginalTranslate(),newlist);
      }
}
```

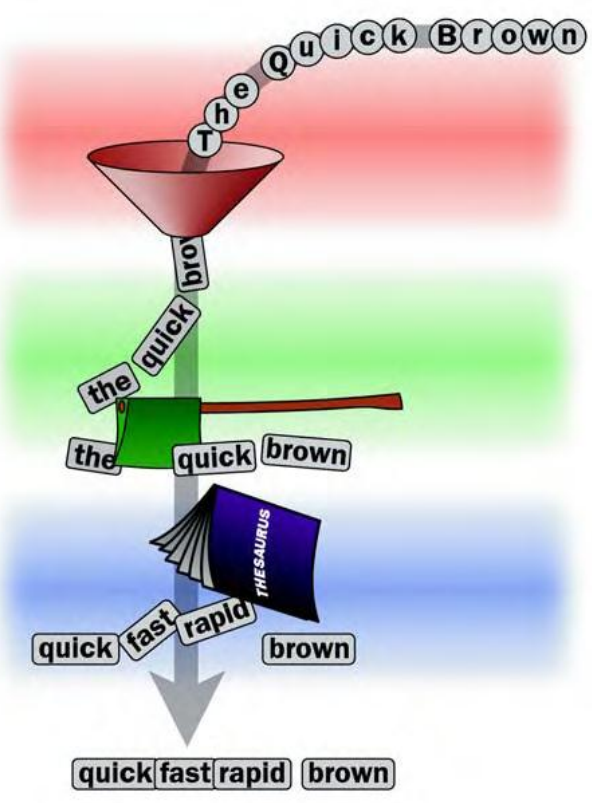
Pre spájanie jednotlivých trigramov a ich ohodnocovanie je nevyhnutné odhaliť veľkosť prekryvu dvoch ngramov. Slúži na to nasledujúca funkcia *getSentenceSimilarity*.

```
private static int getSentenceSimilarity(String s1, String s2){
            String a1[] = s1.toLowerCase().split(" ");
            String a2[] = s2.toLowerCase().split(" ");

            int val = 0;
            for (int i=0; i<a1.length; i++){
                  if (a1[i].equals(a2[0])){
                        for (int j=0; j<a1.length-i; j++){
                              if (a1[i+j].equals(a2[j])){
                                    val++;
                              }
                              else{
                                    val = 0;
                                    break;
                              }
                        }
                        break;
                  }
            }
            return val;
      }
```

9.2 Vyhľadávanie pomocou synoným

9.2.1 Analýza

Elasticsearch umožňuje využívanie zoznamu synoným ako podpory pre vyhľadávanie v indexovaných dokumentoch. To znamená, že pri vyhľadávaní Elasticsearch neprehľadáva len dané slovo ale aj všetky synonymá k tomuto slovu. Táto funkcionalita nám umožní zrýchliť a zlepšiť vyhľadávanie prekladu.



Obr. 38 - princíp fungovania token filtra „synonym“ v elasticsearch.4

Elasticsearch podporuje dva typy formátov synonymických slovníkov:[5]

- Solr format
- WordNet

[4] Prevzaté z HATCHER, E. et al.: Lucene in Action. 2. Vyd. 2010. ISBN 1933988177.
[5] http://www.elasticsearch.org/guide/reference/index-modules/analysis/synonym-tokenfilter.html

9.2.2 Návrh

Pre využite token filtra "synonym" potrebujeme vytvoriť textový súbor obsahujúci synonymá v tvare , ktorý podporuje Elasticsearch. Vhodným formátom je Solr formát.

Kedže máme k dispozícii slovník z PC Translatora v textovom formáte nebude problém ho upraviť do správneho tvaru. Rovnako bude vhodné získať iný slovník, ktorý by sme tiež použili a mohli tak porovnať rôzne verzie a vybrať tú lepšiu. Ako ďalší slovník na úpravu sa dá použiť otvorený slovník Thesaurus, ktorý obsahuje okolo 14000 synoným.

9.2.3 Implementácia

Vytvorili sme dva súbory s názvami "synonymspc.txt" a "synonymsthesaurus.txt", ktoré slúžia ako synonymiské slovníky pre elasticsearch.

Format súborov:

```
administratíva, správa
administratívny, byrokratický, správny, úradný
administrácia, riadenie, správa
administrátor, správca, superpoužívateľ
adopcia, osvojenie, prijatie
adoptovanie, osvojenie si
adoptovať, osvojiť si, prijať za vlastné
adresovať, obrátiť sa na, osloviť
adresár, priečinok
adresát, príjemca
aerodróm, letisko
aeronaut, aviatik, letec, pilot
afekt, afektovanosť, strojenosť, umelosť
afektovanosť, afekt, strojenosť, umelosť
afektovaný, neprirodzený, prehnaný, strojený
agent, dohadzovač, sprostredkovateľ, zástupca, vyzvedač, špión
agregovaný, agregovaný , akcesorický , nepodstatný, nepričlenený, vedľajší
```

Obr. 39 - Časť zo súboru synonymsthesaurus.txt

Taktiež sme vytvorili dva nové indexi, ktoré sme nazvali "synpc" a "synthesaurus". Názvy korelujú s použitými slovníkmi. V oboch indexoch sme vytvorili token filter s názvom "synonym".

Nastavenia indexov:

{"synpc":{"settings":{"index.analysis.analyzer.synonym.filter.0":"synonym","index.analysis.filter.synonym.type":"synonym","index.analysis.analyzer.synonym.tokenizer":"whitespace","index.analysis.filter.synonym.synonyms_path":"analysis/synonymspc.txt","index.number_of_shards":"5","index.number_of_replicas":"1","index.version.created":"190053"}}}

{"synthesaurus":{"settings":{"index.analysis.analyzer.synonym.filter.0":"synonym","index.analysis.filter.synonym.type":"synonym","index.analysis.analyzer.synonym.tokenizer":"whitespace","index.analysis.filter.synonym.synonyms_path":"analysis/synonymsthesaurus.txt","index.number_of_shards":"5","index.number_of_replicas":"1","index.version.created":"190053"}}}

9.3 Reindexácia údajov z indexu sentences do indexov synoným

Po vytvorení požadovaných indexov, bolo potrebné ich naplniť dátami. Namiesto opätovného manuálneho vkladania viet z MySQL databázy do oboch nových indexov, sme sa rozhodli reindexovať dáta z už existujúceho ElasticSearch indexu menom sentences. Toto riešenie bolo zvolené z dôvodu vyššej efektívnosti a nižšej dobe trvania.

9.3.1 Implementácia reindexácie údajov

Reindexáciu sme implementovali v jazyku Perl s použitím knižnice *ElasticSearch - An API for communicating with ElasticSearch*[6] vo verzii 0.52.

Zdrojový kód skriptu:

```
#!/usr/bin/perl

use ElasticSearch();
my $es = ElasticSearch->new(
   transport => 'http',
   servers => '127.0.0.1:9200' -- host:port na ktorom beží ElasticSearch
);

my $source = $es->scrolled_search(
   index => 'sentences',       -- index z ktorého chceme získať dáta
   search_type => 'scan',
   scroll => '15m',
   version => 1
);

$es->reindex(
   source => $source,          -- dáta ktoré chceme vložiť do indexu
   dest_index => 'synpc',      -- názov indexu kam chceme vložiť dáta
   bulk_size => 5000           -- veľkosť dávky po akej sa budú dáta
);                             -- vkladať
```

Uvedený skript bol spustený pre obidva nové indexy, synpc a synthesaurus.

9.4 Logovanie v prekladači

9.4.1 Analýza

Počas procesu prekladanie slov a viet prostredníctvom našej služby sa niekoľkokrát stalo, že náš prekladač nevrátil požadovaný výsledok. Táto neočakávaná situácia bola spôsobená internou chybou, ktorá vznikla v niektorej fáze prekladu. Elimináciu týchto chýb je možné dosiahnuť logovaním činnosti prekladača.

[6] https://metacpan.org/module/ElasticSearch

java.util.logging

Java nám poskytuje balík *java.util.logging*, ktorý však predstavuje iba okliešteną verziu log4j. Základným objektom je trieda typu Logger. Správy, ktoré prostredníctvom nej zaznamenávame, môžu mať rôzne úrovne dôležitosti (FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVERE). Na začiatku logovania je nutné nastaviť jej povolenú úroveň – štandardne je INFO. Všetky správy sú posielané do handlera, ktorý overuje jej dôležitosť. V prípade, že je nižšia ako vopred definovaná úroveň, handler správu neakceptuje a zahodí ju. Správy je možné formátovať triedou formater, ktorá poskytuje ich výstupy do štandardných súborov typu .txt či .xml.

log4j

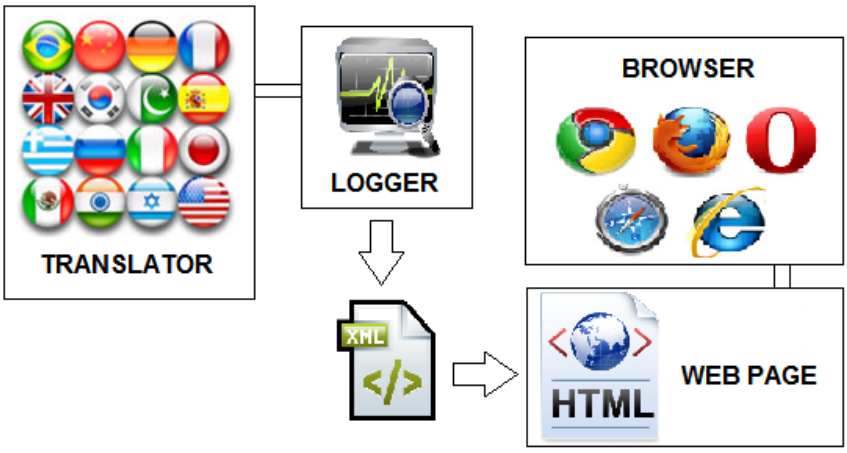
Log4j je balík, ktorý slúžil ako základ pri tvorbe JUL. Publikácia správ sa vykonáva implementáciou rozhrania Appender, ktorý je ekvivalentom vyššie spomínaného handleru. Na rozdiel od neho je však možné použiť mapovanie do jednotlivých stratégií výstupu. Zdedením abstraktnej podtriedy Layout vytvoríme vlastnú štruktúru výstupu.

Výsledky získané logovaním je nutné mapovať na webovú stránku tak, aby bolo umožnené ich zobraziť v prehľadnej forme. Z tohto dôvodu je potrebné vytvoriť vlastný formát výstupu do XML. V tomto prípade je preto vhodnejšie použiť log4j.

9.4.2 Návrh

Vytvárané riešenie logovanie musí spĺňať niekoľko základných požiadaviek. Medzi ne patrí hlavne plná automatizovateľnosť procesu. Ten zhromaždí množinu výsledkov, ktoré následne poskytnú programátorovi relevantné informácie o príčinách pádu služby.

Nami navrhnuté riešenie je opísané na Obr. 11. K nášmu prekladaču je pripojený logovací balík log4j, ktorý monitoruje jeho činnosť. Správy, ktoré mu prekladač posiela, sú mapované do pripraveného XML formátu. Vytvorená webová stránka načíta informácie zo vstupného súboru. Tie následne zobrazí vo svojom obsahu prostredníctvom zoznamu umiestneného v tabuľke. Keďže predpokladáme, že XML súbor bude obsahovať veľké množstvo záznamov, používateľ musí mať možnosť tento zoznam filtrovať.



Obr. 40 - Návrh logovania našej služby

Do nášho prekladača bola pridaná knižnica *log4j-1.2.16.jar*, ktorá zabezpečuje logovanie do súboru. Súčasne bol vytvorený balík *sk.stuba.fiit.adictit.xml_logger*, ktorý obsahuje triedu *API_Logger*. Tá implementuje viditeľnú funkciu *makeLog* prostredníctvom ktorej je realizované logovanie do súboru. Parametre funkcie opisuje Obr. 12.

```
catch(ClassNotFoundException ex)
{
    API_Logger.makeLog(Translator.class, "ERROR", ex, "CannotTranslateTextException - Can't inicialize class!");
    throw new CannotCreateTranslatorException(ex);
}
```

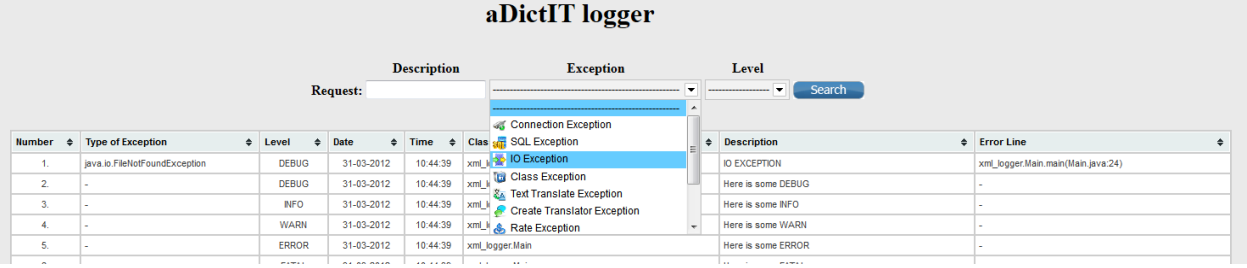
Názov triedy
Dôležitosť správy
Výnimka, ktorá nastala
Správa pre programátora

Obr. 41 - Volanie vytvoreného loggera

Vyššie opísaná funkcia využíva nami vytvorenú XML šablónu dokumentu (Obr. 13). Takto uložený súbor je následne poskytnutý webovej stránke (Obr. 14).

```
<EXCEPTION>
    <TYPE>java.io.FileNotFoundException</TYPE>
    <LEVEL>DEBUG</LEVEL>
    <DATE>31-03-2012</DATE>
    <TIME>10:44:39</TIME>
    <CLASS>xml_logger.Main</CLASS>
    <DESCRIPTION>IO EXCEPTION</DESCRIPTION>
    <ERROR_LINE>xml_logger.Main.main(Main.java:24)</ERROR_LINE>
</EXCEPTION>
```

Obr. 42 - Správa z prekladača zapísaná v súbore XML.



Obr. 43 - Webová stránka s načítaným XML súborom.

9.4.3 Testovanie

Zobrazenie vstupného XML súboru na webovej stránke

Pre overenie úspešnosti mapovania XML súboru sme použili nasledovný testovací scenár, podľa ktorého sme otestovali funkčnosť implementácie.

Názov	Mapovanie XML súboru	ID Testu	**0x-0x**
Rozhranie	logger/index.php	ID UC	**0x**
Účel	Overenie úspešnosti mapovania XML súboru		
Vstupné podmienky	XML súbor umiestnený v rovnakom priečinku ako webová stránka.		
Výstupné podmienky	Používateľ vidí na obrazovke vyplnenú tabuľku.		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**
1.	Načítanie XML súboru.	XML bude načítaný v tabuľke webovej stránky.	Dáta z XML súboru boli v tabuľke zobrazené v požadovaných stĺpcoch.

Filtrovanie správ načítaných z XML súbora

Nasledovný testovací scenár overuje funkčnosť implementácie filtrov.

Názov	Filtrovanie	ID Testu	**0x-0x**
Rozhranie	logger/index.php	ID UC	**0x**
Účel	Overenie úspešnosti filtrovanie XML súboru		
Vstupné podmienky	XML súbor umiestnený v rovnakom priečinku ako webová stránka.		
Výstupné podmienky	Používateľ vidí na obrazovke vyplnenú tabuľku podľa zvoleného kritéria.		
Krok	**Akcia**	**Očakávaná akcia**	**Skutočná reakcia**
1.	Načítanie XML súboru.	XML bude načítaný v tabuľke webovej stránky.	Dáta z XML súboru boli v tabuľke zobrazené v požadovaných stĺpcoch.
2.	Používateľ nastaví filter.	V tabuľke sa zobrazia iba tie kritéria, ktoré vyhovujú filtru.	Na webovej stránke sa zobrazili iba tie dáta z XML súboru, ktoré boli filtrom vybraté.

9.5 Nové webové rozhranie prekladača

Chceme implementovať viacero moderných prvkov do rozhrania prehliadača. Cieľom je využívať možnosti škálovania prekladača ako aj zrýchliť odozvu servera na používateľove dopyty.

9.5.1 Analýza

Súčasná funkcionalita webového rozhrania prekladača je zabezpečovaná iba na strane servera prostredníctvom PHP. To spôsobuje, že pri každom preklade sa musí stránka znovu načítať. Keďže stránka rozhrania je v súčasnosti veľmi jednoduchá, toto načítavanie nespôsobuje významné zdržanie. V budúcnosti ak sa však rozhodneme pridať nejaké grafické komponenty na stránku, nebude ich potrebné načítavať opätovne pri každom preklade. To určite skráti čakanie používateľa na zobrazenie odpovede servera. S využitím technológie AJAX by sa pri preklade načítali znova dáta iba do tých častí, kde sa zobrazujú prekladané výrazy. Tento prístup úspešne využíva aj konkurenčný prekladač Google Translate. Jeho rozhranie je aj vďaka tomu veľmi rýchle na odozvu.

Samotný preklad je v súčasnosti vykonávaný tak, že text zadávaný do textového poľa sa odošle funkcii, ktorá vráti jeden najlepší preklad danej vety. Pre zobrazenie alternatívnych prekladov je volaná iná funkcia, ktorá vráti všetky ostatné získané preklady. Spustením prekladu sa znovu načíta aj celá stránka prekladača. Aby sme na stránke popri prekladu zobrazili aj pôvodný prekladaný zdrojový text, zobrazí sa v ňom obsah globálnej premennej $_POST[‘type_here’], do ktorej bol text po spustení prekladu uložený.

9.5.2 Návrh

Používateľ má mať možnosť vybrať, s ktorou verziou prekladača chce vykonať preklad. Pre túto činnosť navrhujeme pridať do rozhrania rozbaľovacie menu, ktoré dynamicky načíta zoznam dostupných verzii z danej URL. Použijeme pritom nejaké dostupné efekty s využitím JavaScript knižnice jQuery a CSS.

Na zrýchlenie samotného prekladu zavoláme iba raz funkciu, ktorá zabezpečí preklad a jej prvú hodnotu zobrazíme v rámčeku pre najpravdepodobnejší preklad. Prípadné ďalšie hodnoty zobrazíme v časti pre alternatívne preklady. Výsledok zobrazíme na stránke, bez potreby jej znovu načítania. Pomocou technológie AJAX načítame iba získané dáta do častí, ktoré zobrazujú výsledky prekladu.

9.5.3 Implementácia

Dostupné verzie prekladača sú do rozbaľovacie menu načítavané priamo z webovej služby. Vytvorením novej verzie prekladača je táto verzia automaticky viditeľná aj v rozhraní a to bez potreby akéhokoľvek zásahu do zdrojového kódu rozhrania. Z URL adresy zobrazujúcej JSON je pomocou natívnej PHP funkcie file_get_contents(“URL“) získaný obsah, ktorý je následne skonvertovaný do asociatívneho reťazca. Jednotlivé hodnoty tohto reťazca sú zobrazené v príslušných HTML elementoch a tvoria rozbaľovacie menu a aj obsah pomocníka pre verzie prekladača.

Samotné odoslanie požiadavky s prekladaným textom na server vykonávame s použitím technológie AJAX. Pre vykonanie AJAX volaní a následné spracovanie odpovedí zo servera využívame JavaScript knižnicu jQuery. Vzhľadom na fakt, že volaná webová služba beží na inom porte

ako rozhranie prekladača, nie je možné vykonať toto volanie na službu priamo. Namiesto toho pomocou AJAX voláme, pre tento účel vytvorený, súbor s názvom service_connect.php. Jedinou úlohou tohto súboru je zabezpečenie komunikácie s príslušnou webovou službou. Keďže chceme prekladať aj veľké úryvky textu využívame pri tom HTTP metódu POST. Pre jej využitie v PHP bolo potrebné použiť niektoré curl funkcie dostupné v PHP knižnici libcurl.

Využitím tohto prístupu na obídenie tzv. cross-site scripting prevencie sme docielili, že rozhranie prekladača je jednoducho prenositeľné. To všetko bez potreby menenia nejakých serverových nadstavení. Stačí totiž zmeniť iba dva linky v PHP súboroch a rozhranie môže byť implementované na inom serveri.

Samotná odpoveď webovej služby na zadaný prekladaný text je vlastne JSON obsahujúci príslušné preklady. JSON je u klienta skonvertovaný na asociatívny reťazec a jeho prvé hodnoty sú zobrazené v rámčeku pre najviac pravdepodobný preklad. Ostatné hodnoty reťazca sú zobrazené v časti pre alternatívne preklady. Celé toto spracovanie prebieha v klientskom okne prehliadača pomocou JavaScript-u.

10 Ôsmy šprint – koks

Ako používateľ chcem mať prekladač už hotový.

10.1 Pridanie vplyvu Term frequency na preklad cez N-gramy

10.1.1 Analýza

Využívanie N-gramov má niekoľko výhod. Hlavnou je schopnosť nafúknuť existujúci korpus čím sa získa možnosť lepšieho prekladu. Napríklad ak je vyhľadávaná taká veta, ktorá sa v korpuse viet nenachádza, výsledkom vyhľadávania bude len časť hľadanej vety. Ak sa použijú n-gramy je väčšia pravdepodobnosť, že časti hľadanej vety sa v korpuse nájdu. To znamená, že sme schopný nájsť aj takú vetu, ktorá v korpuse neexistuje ale je prítomná len po častiach. Táto vlastnosť na druhej strane spôsobuje problémy v podobe nutnosti spájať vety a generovať vysoký počet kombinácií viet pre hľadanie prekladu. Napríklad z päť slovnej anglickej vety sa pri preklade vygeneruje zhruba tisíc viet. Spracovanie toľkých viet trvá značnú dobu čím sa zhoršuje celková použiteľnosť systému.

Ďalším problémom je strata kontextu. Nakoľko sa pracuje s trojicami slov dochádza k strate informácie o význame. Tento jav sa prejavuje negatívne vo vyhľadávaní. Napríklad pri vete „My dog is angry“ sa medzi generovanými kombináciami objavia preklady ako „môj pes je nahnevaný“ , „svoj pes je rozzurený“ , „môj pes informačný systém nasadený“. Tieto vety sa rozdelia na N-gramy a vyhľadávajú sa v elasticsearch. Problém je dvojica slov „informačný systém“, ktorý sa v korpuse nachádza. Aj keď je preklad slova „is“ ako informačný systém úplne zlý, v korpuse sa nájde a vďaka absencii kontextu dostane takáto veta vysoké skóre. Výsledkom je nesprávny preklad, ktorý dostane používateľ.

Oba problémy, strata kontextu a veľké množstvo kombinácií, sa dá odstrániť zavedením Term frequency (ďalej len TF) do procesu prekladu ešte pred vyhľadávaním vygenerovaných kombinácií viet. Zároveň TF použitá po vyhľadávaní dokáže dodatočne pridať skóre vhodným vetám. „Informačný systém“, ktorý sa v korpuse nachádza len zriedka tak nebude mať dosť sily na to aby sa veta, ktorá ho obsahuje, dostala medzi výsledky.

10.1.2 Návrh

Nakoľko je v našom slovníku uložená aj informácia o TF slova v korpuse, nie je nutné robiť v tomto smere veľké zmeny. Stačí aby táto informácia bola spolu so slovami vybraná z databázy. Informácia o TF slova sa musí použiť ešte pred vyhľadávaním kombinácií viet v korpuse. Na základe TF dokážeme odstrániť tie preklady, ktoré majú veľmi nízku relevanciu vzhľadom na korpus. Takto signifikantne znížime počet kombinácií viet (resp. n-gramov), ktoré sa budú vyhľadávať v korpuse a zároveň nedovolíme nezmyselným prekladom aby sa vôbec zúčastnili vyhľadávania.

Ďalším krokom je pridávanie TF ku výslednému skóre. Vhodným vzorcom, ktorým sa pridá TF ku skóre spôsobí, že n-gramy so slovami s vysokým TF (predpokladá sa vyššia relevantnosť týchto slov) sa dostanú na predné pozície prekladov. Týmto spôsobom sa vracia kontext vety tak ako to bolo pri vyhľadávaní celých viet.

Vzorec, ktorý navrhujeme použiť je nasledovný:

N-gram s najvyšším skóre dostane 1.2 násobok skóre bez ohľadu na jeho TF. Ostatné n-gramy získajú skóre podľa formuly:

$$x_v = \frac{\mathrm{x} * \ 1{,}2 * \sum F_s}{\mathrm{F_n}}$$

X_v – výsledné skóre.

X – skóre n-gramu vrátené elasticsearchom.

$\sum F_s$ – súčet term frekvencií slov v n-grame.

F_n – hodnota súčtu term frekvencií najlepšieho n-gramu.

Pomocou tohto vzorca dosiahneme, že n-gramy s vysokou TF sú schopné získať vyššie skóre. Správnosť navrhovaného vzorca sa ukáže pri testovaní riešenia a v prípade potreby bude zmenený.

10.1.3 Implementácia

Bola vytvorená verzia, ktorá pracuje s TF s názvom NgramVersion. Táto verzia používa tiež novovytvorenú triedu pre prácu s prekladačom TranslatorWithLimitTermFrequencyForNgram a pre hľadanie v korpuse NgramFinder.

```
public NgramVersion(final String language_from, final String language_to)
                throws CannotCreateTranslatorException,
                CannotCreateSenteceFinderException
{
try {
    final ConfigInfo config = ConfigInfo.instance();
    Evalve evaluation = new Evalve();
    int limit = config.transWLimTFLimit;
        finder = new ContextFinder(new NgramFinder(evaluation));
        translator = new ContextTranslator(new TranslatorWithLimitTermFrequencyForNgram(language_from,
                        language_to,limit,evaluation));
} catch (IOException ex) {
    Logger.getLogger(NgramVersion.class.getName()).log(Level.SEVERE, null, ex);
}
}
```

Obr. 44 - verzia schopná pracovať s TF

Hlavnou zmenou je pridanie triedy Evalve, ktorá obsluhuje filtrovanie nevhodných prekladov a zároveň aj modifikáciu skóre.

```
public List<SentenceTranslation> evaluateNgram(List<SentenceTranslation> sentences)
 {
     float score;
     float TF =0f;

     for(int e=0;e<sentences.size();e++)
     {

         String[] s =sentences.get(e).getOriginalTranslate().split(" ");
         score=0.01f;
         for(int i=0;i<s.length;i++)
         {
             if(mapOfWord.containsKey(s[i]))
             {
                 score +=mapOfWord.get(s[i]).TF;
             }
         }

         if(e==0)
         {
             sentences.get(e).setScore(sentences.get(e).getScore()*(1.2f));
             TF=score;
         }
         else
         {
             sentences.get(e).setScore(sentences.get(e).getScore()*(1.2f * (score/TF)));
         }
     }
     return sentences;
 }
```

Obr. 45 - modifikujúca skóre v triede Evalve

```
List<SentenceTranslation> translations = evaluation.evaluateNgram(super.rate(Arrays.asList(uniquengrams.toArray(new String[0]))));
```

Obr. 46 - volanie metódy pre modifikovanie skóre v triede NgramFinder

```
for(int i = 0; i < transComb[0].size(); i++)
{
        sentences = cobineSentences(transComb, 0, i, sentence, sentences);
}
sentences = evaluation.evaluateTranslations(sentences);
```

Obr. 47 - volanie metódy pre filtrovanie prekladov

10.1.4 Testovanie

Pri zadaní vety „My dog is angry“ už nevrati „môj informačný system a“ ako to bolo predtým ale vráti:

{"sentences":[{"original":"my dog is angry","translations":["svoj pes je nahnevaný"]}]}

Bude potrebné dôkladné testovanie aby sme zistili najsprávnejšie parametre pri, ktorých bude TF plniť svoju úlohu.

10.2 Pridanie filtrovania podľa dĺžky preloženej vety

10.2.1 Analýza

Pri prekladaní viet sa často stane, že v korpuse sa síce nachádza veta, ktorá obsahuje všetky slová slovníkom preloženej vety, avšak táto veta je veľmi dlhá. Vďaka tomu sa často stáva, že preklad, ktorý sa vráti späť používateľovi je sémanticky úplne nezmyselný. Napríklad, majme anglickú vetu „It was compliance“. Jeden so slovníkových prekladov, ktoré sa vygenerujú je aj „To bol súlad“. Elasticsearch hľadá túto vetu v korpuse a ako najlepší výsledok vráti „komunikácia s Milanom Haborákom bola maximálne korektná, bol k nám ústretový a našli sme súlad“. Dôvodom tohto výsledku je malý rozsah korpusu, nakoľko sa v ňom nenachádza žiadna lepšia veta obsahujúca hľadané slová. Aj keď je táto veta v skutočnosti najlepším prekladom, je po sémantickej stránke nesprávna. Preto by sa takéto vety radšej nemali zobrazovať používateľovi.

10.2.2 Návrh

Na odstránenie spomínanej chyby použijeme filtrovanie prekladov na základe dĺžky vyhľadávanej vety. Aby bolo možné nájsť aj vety, ktoré nemajú presne rovnaký počet slov, bude stanovený rozsah o počte dvoch slov, ktoré môže veta nájdená v korpuse mať viac alebo menej.

V elasticsearch je možné vykonať takéto filtrovanie tromi spôsobmi:

- Range Query.
- Range Filter.
- Numeric Range Filter.

Vzhľadom na pamäťové a časové nároky je v našom prípade vhodné použiť Numeric Range Filter.

10.2.3 Implementácia

Požiadavka na hľadanie bola pozmenená tak aby obsahovala Numeric Range Filter, ktorý filtruje vety na základe počtu slov, ktoré obsahujú. Filter je aplikovaný na pole „words“ predstavujúci počet slov v danej vete.

Implementácia v Java:

```
    s =sentence.split(" ");
final SearchRequestBuilder request = client
                .prepareSearch(indexName).setTypes(typeName)
                .setSearchType(SearchType.QUERY_AND_FETCH)
                .setQuery(QueryBuilders.fieldQuery("sentence", sentence))
        .setFilter(FilterBuilders.numericRangeFilter("words").from(s.length-2).to(s.length+2))   // ADD FILTER
                .setFrom(0).setSize(1);
```

Obr. 48 - ukážka implementovaného kódu

Použitie filtra cez curl:

```
[xrumanv@tim04 ~]$ curl -XGET http://localhost:9200/sentences/_search -d '{"quer
y" : {"field" : {"sentence" : "No tak toto som căte nevidel"}} , "filter" : { "n
umeric_range" : {"words" : {"from" :4, "to" :8 }}}}'
```

Obr. 49 - použitie filtru cez curl

10.3 Čistenie viet

10.3.1 Nahradenie menných entít

Korpus viet, uložený v MySQL databáze, je potrebné očistiť o identifikovateľné vlastné mená a názvy. Tie je potrebné nahradiť vopred dohodnutou sadov znakov. Rovnako tak je z korpusu potrebné odstrániť číslovky a nahradiť ich odlišnou sadov znakov. Cieľom tejto úlohy je vybrať z korpusu viet slová, nachádzajúce sa vo vetách, ktoré používateľ môže prekladať, ale pravdepodobne s inými názvami alebo inými číslovkami. Tým dosiahneme, že takáto veta, obsahujúca číslovky alebo názvy, nedosiahne horšie skóre pri výsledkoch prekladu len preto, že používateľ zadal číslovku 100 a v korpuse sa nachádza číslo 22 alebo, že používateľ zadal názov Vrakuňa a v korpuse sa nachádza identická veta s názvom Žilina. Tým dosiahneme, že tieto identifikované entity sa nebudú prekladať. Prekladač bude teda prekladať len tzv. významovú časť vety.

Realizácia

Implementáciu tejto úlohy sme realizovali prostredníctvom jazyka Java. Prostredníctvom vytvorených regulárnych výrazov sme najskôr identifikovali všetky entity typu: RSS, FeedReader. Tie sme nahradili dohodnutou sadov znakov. Následne sme každé začiatočné písmeno vety, ktoré neobsahovalo začiatočné písmeno z dohodnutej sady znakov, previedli z veľkého písmena na malé. Pomocou ďalších dvoch regulárny výrazov sme identifikovali všetky číslovky a názvy začínajúce veľkým písmenom a nahradili dohodnutou sadov odlišných znakov. Úloha bola implementovaná pomocou vytvoreného súboru placeholder.java.

Overenie

Vizuálnou kontrolou sa nám nepodarilo odhaliť žiadne entity, ktoré by sme nami vytvorenou funkciou nedokázali identifikovať a nahradiť. Jediné slabé miesto vytvorenej funkcie je, že ak veta začína nejakým názvom napr. Bratislava, tak by sa nám tento názov nepodarilo identifikovať. Dôvodom je, že názov je na začiatku vety, kde všetky slová začínajú veľkým písmenom. Prevedená vizuálna kontrola však neodhalila takéto prípady.

10.3.2 Čistenie viet v korpuse

Pri prezeraní aktuálnych viet v korpuse sme zistili, že nie všetky s nich sú v korektnom stave. Teda obsahovali rôzne problémy, ktoré by sa vo vetách nemali vyskytovať. Preto sme sa rozhodli vykonať ďalšie čistenie viet v korpuse.

Tabuľka 15 - Problémy vo vetách v korpuse

Názov problému	Príklad
Nevhodné znaky	-, #,), (,], [, :, ”, $, %, |
Html značky	,,,
Viac viet v jednom riadku	Ahoj ako sa máš? Ja dobre. A ty?

10.3.3 Realizácia

Pri realizácií tejto úlohy sme postupovali nasledovne:

1. Vytvorenie CSV exportu viet nachádzajúcich sa v MySQL databáze s názvom SME_CLANKY.
2. Vytvorenie textového súboru z exportovaného CSV v ktorom sa v jednom riadku nachádza práve jeden záznam. Realizované pomocou programu implementovaného v Jave.
3. Odstránenie problémov vo vetách. Vyčistenie textového súboru do požadovanej podoby.
4. Vytvorenie základného indexu v ElasticSearch s názvom sentences_clean. Pomocou príkazu

```
$ curl -XPUT 'http://localhost:9200/sentences_clean/'
```

5. Naplnenie indexu dátami so súboru. Realizované pomocou programu[7] implementovaného v Jave, ktorého vstupom bol vytvorený textový súbor.

10.4 Nahrádzanie číslovieк

10.4.1 Analýza

Vytvorený korpus obsahuje niekoľko miliónov záznamov, ktoré slúžia ako základ nášho prekladača. Pri preklade sa snažíme nájsť podobnú vetu v korpuse, prípadne ju vyskladať. Používateľ sa rozhodne, že chce preložiť vetu, ktorá obsahuje číslovku. Pri vytváraní prekladu však nastáva problém, ktorý vzniká zámenou číslovky. Preto by bolo vhodné, aby sa v ElasticSearchi hľadala veta bez zadanej číslovky. Vrátené skóre jednotlivých potenciálnych prekladov by sa tak zbytočne nezničovalo.

10.4.2 Návrh

Do zdrojových kódov nášho prekladača je potrebné implementovať triedu, ktorá odstráni zo vstupnej vety číslovky. Zvyšná časť vety sa následne hľadá v korpuse. Ak sa vo vrátenom výsledku nachádza číslovka, skóre vety vynásobíme o vopred definovanú konštantu. V prípade viacerých čísloviek sa proces násobenia opakuje. Vo výsledku prekladu nahradíme existujúce číslovky tými, ktoré používateľ zadal na vstupe. Vrátenú štruktúru JSON nakoniec zoradíme podľa dosiahnutého skóre.

10.4.3 Implementácia

Do riešenia bola pridaná trieda *NumberExtractor*, ktorá je umiestnená v balíku *sk.stuba.fiit.adictit.translator,sentences*. Jej obsahom sú metódy:

- ***private int sentenceNumberChecker(String sentence)*** – vráti koľko čísloviek sa nachádza v zadanej vete *sentence*.

[7] git@gitbus.fiit.stuba.sk:statisticky-preklad-volneho-textu/elastic-index-data-filler.git

- ***public String prepareSentence(String sentence)*** – zo vstupnej vete *sentence* odstráni všetky číslovky.
- ***public String replaceNumbers(String sentence)*** – nahradí číslovky v preloženej vete *sentence* za číslovky, ktoré používateľ zadal v pôvodnej vete.
- ***public float boostNumberScore(String translate, float sentenceScore)*** **–** overí, či sa v pôvodnej a v preloženej vete nachádzajú číslovky. V prípade, že áno, vynásobí skóre vety vopred definovanou konštantou. Počet násobení závisí od počtu čísloviek vo vete (Obr. 50).
- ***public JSONArray sortJSONByScore(JSONObject sentenceInfoJson)*** – výsledky, ktoré sa nachádzajú v objecte JSON, zoradí podľa veľkosti dosiahnutého skóre

```
public float boostNumberScore(String translate, float sentenceScore) throws CannotCreateSenteceFinderException
{
    int numbersAtOriginalSentence = originalSentenceNumberCount;
    int numbersAtTranslatedSentence = sentenceNumberChecker(translate);

    try
    {
        // nacitaj hodnotu konstanty, o ktoru budeme zvysovat skore vety
        final ConfigInfo info = ConfigInfo.instance();
        float boostSentenceNumber = info.esSfBoostSentenceNumber;

        // ak sa nachadzaju vo vete cislovky a nachadzaju sa cislovky aj v prelozenej vete
        if (numbersAtTranslatedSentence>0 && numbersAtOriginalSentence>0)
        {
                for (int q=0;q<numbersAtTranslatedSentence;q++)
                {
                    // navys skore vety
                    sentenceScore*=boostSentenceNumber;
                }
        }
    }
    catch (IOException ex)
    {
        throw new CannotCreateSenteceFinderException(ex);
    }

    return sentenceScore;
}
```

Obr. 50 - Metóda zvyšujúca skóre vety podľa počtu čísloviek, ktoré sa v nej nachádzajú

10.4.4 Testovanie

Testovanie implementovanej triedy bolo vykonané na viacerých vetách. Ako príklad možno uviesť anglickú vetu *„was born in 1966“*. Pred implementovaním triedy nám prekladač vracal ako najvhodnejší preklad vetu *„Narodil som sa v Marseille.“*. Nami hľadaný preklad *„Narodil som sa v 2005.“* sa nachádzal až na piatom mieste. Bolo to z toho dôvodu, že táto veta obsahovala číslovku *2005* no v skutočnosti bola hľadaná veta s číslovkou *1966*. Priradené skóre teda nezodpovedalo reálnemu prekladu. Bolo potrebné nájsť vhodnú hodnotu konštanty, ktorou by sme násobili dosiahnuté skóre vety v prípade, že obsahuje číslovku. Vykonané testovanie prekladania viacerých viet ukázalo, že najlepšie výsledky boli dosiahnuté pri hodnote 1,45. Implementácia tejto zmeny spôsobila, že nás prekladač nám vrátil ako najlepší preklad vetu *„Narodil som sa v roku 2005.“*, v ktorej sa následne nahradila číslovka *2005* za *1966*. Používateľovi sa tak na výstupe zobrazila veta *„Narodil som sa v roku 1966.“*, ktorá zodpovedala prekladu jeho vstupu *„was born in 1966“*.

10.5 Verzia poradie slov v slovníku

10.5.1 Analýza

Aktuálne verzie prekladača nedokážu prekladať niektoré vety správne, napriek tom, že medzi prekladmi sa správny preklad nachádza. Stáva sa, že nesprávne preklady, získajú pri ohodnocovaní väčšie skóre a tak správny preklad používateľovi nie je zobrazený alebo sa nachádza len medzi alternatívnymi prekladmi. Problémom je, že skóre určuje len Elastic Search, ktorý nerobí rozdiely medzi jednotlivými prekladmi, a tak napríklad pre angl. slovo water sú preklady voda a jas považované za ekvivalentné.

Slovník, ktorý aktuálne používame však pri preklade slov, vracia usporiadané pole, kde najsprávnejšie preklady sú na prvých pozíciách. Existujúci spôsob vytvárania a obohacovanie viet však toto poradie slov v slovníku nezohľadňuje. Vytvorenie metódy, ktorá zvýhodní vety obsahujúce najpravdepodobnejšie preklady jednotlivých slov môže prispieť k správnejšiemu usporiadaniu výsledných viet.

10.5.2 Návrh

Pre vytvorenie metódy, ktorá bude zohľadňovať poradie slov v slovníku je potrebné upraviť metódy pre preklad slov, generovanie viet a ich ohodnocovanie. Navrhujeme nasledovné 3 zmeny:

- Priradenie skóre slovám zo slovníka, pričom prvý preklad slova bude mať skóre 1 a každé ďalšie 0.9 násobok predchádzajúceho.
- Priradenie skóre vetám, ktoré sa vyhľadávajú v korpuse. Toto skóre sa vypočíta ako súčin skóre jednotlivých slov, ktoré tvoria vetu.
- Vynásobenie skóre vety so skóre, ktoré vypočíta ElasticSearch.

10.5.3 Implementácia

Pre priradenie skóre zohľadňujúce poradie v slovníku sme vytvorili rozšírenie triedy Translator pod názvom ScoringTranslator. Prvé slovo vždy dostane hodnotu 1, ďalším slovám je priradený 0.9 násobok skóre predchádzajúceho slova.

```
double score = 1.0;

while(result.next())

{

	translations.add(

		new WordTranslation(

			result.getString("translate"),

			result.getString("source"),

			score

		)
```

```
        );
        score *= 0.9;
}
```

Od tohto momentu už nestačilo vetu reprezentovať len dátovým typom String, ale museli sme vytvoriť triedu ScoredSentence, ktorá okrem samotného teľa vety uchováva aj jej skóre.

```
public class ScoredSentence {
        public String body;
        public double score;
        public ScoredSentence(String body, double score){
                this.body = body;
                this.score = score;
        }
        public ScoredSentence getCopy(){
                return new ScoredSentence(this.body,this.score);
        }
}
```

Následne sme museli upraviť metódy prekladača aby podporovali prácu s novým typom. V nasledujúcej ukážke kódu je možné vidieť počítanie skóre pre celú vetu.

```
public List<ScoredSentence> combineSentences(List<WordTranslation>[] transComb, int
word, int meaning, ScoredSentence sentence, List<ScoredSentence> sentences)
{
        sentence.body += transComb[word].get(meaning).translate + " ";
        sentence.score *= transComb[word].get(meaning).score;

        if(word + transComb[word].get(meaning).worlds == transComb.length)
        {
                sentences.add(new ScoredSentence(sentence.body.trim(), sentence.score));
        }
        else
```

```
        {
                for(int i = 0; i < transComb[word +
                            transComb[word].get(meaning).worlds].size(); i++)
                {
                        sentences = combineSentences(transComb, word +
                        transComb[word].get(meaning).worlds, i, sentence.getCopy(),
                        sentences);
                }
        }
        return sentences;
}
```

11 Použitá literatúra

[1] SLOVAKE: *O jazyku*. 2011-03-13 [cit. 2011-05-01]. Dostupné na internete: http://slovake.eu/sk/intro/language/general

[2] HORVÁTH, R.: Spracovanie textu v slovenčine, In: Podpora výkladu neznámeho pojmu pri prehľadávaní v slovenčine, s. 3-6.

[3] LACLAVÍK, M. et al.: Dostupné zdroje a výzvy pre počítačové spracovanie informačných zdrojov v slovenskom jazyku. In: 1st Workshop on Intelligent and Knowledge oriented Technologies, Institute of Informatics SAS, 2007, s. 92-98.

Príloha A – Preklad viet

A.1. Vysvetlenie označenia v prílohách

Pri tabuľkách, ktoré sa vyskytnú v nasledujúcich kapitolách, budú použité symboly. Ich význam je potrebný pre správnu orientáciu vo výsledkov.

Zoznam symbolov:

* - zadaný český výraz rovnakého významu

A.2. Z angličtiny do slovenčiny a češtiny

Prvým objektom záujmu je správnosť prekladu anglickej vety do slovenčiny či češtiny. Na vybraných dvanástich vzorkách bola vypracovaná ukážka ohodnocovania, ktorá zachováva farebnú konvenciu z tabuľky č.1.

		Slovenský preklad			Český preklad		
č.	**Anglická veta**	**Google Translate**	**Bing Translate**	**PC Translator 2010**	**Google Translate**	**Bing Translate**	**World Lingo**
1.	Where are you from?	Kde ste?	Odkiaľ si?	Odkiaľ ste?	Kde ste?	Odkud jsi?	Odkud jste ?
2.	I have a mother.	Mám matku.	Som matka.	Ja mám matku.	Mám matku.	Mám matku.	Mám matku .
3.	What are you doing in Slovakia?	Čo robítev Slovensko?	Čo robíš na Slovensku?	Čo robíš na Slovensku?	Co děláteve Slovensko?	Co děláš na Slovensku?	Co děláš na Slovensku ?
4.	We will not block other countries.	Nebudeme blokovať iných krajín.	My neblokujeiných krajín.	My nebudeme blokovať iné krajiny.	Nebudeme blokovatjiných zemí.	Nám nebude blokovat další země.	Nebudeme blokovat ostatní země .
5.	There is no agreement.	Neexistuje žiadna dohoda.	Neexistuje žiadna dohoda.	Neexistuje žiadny dohoda.	Neexistuje žádná dohoda.	Neexistuje žádná dohoda.	Neexistuje žádná dohoda .
6.	Chinese Embassy rented the government hotel for celebration.	Čínskeho veľvyslanectva prenajať vládne hotela k oslave.	Čínskeho veľvyslanectva prenajíma vláda hotel pre oslavy.	Čínsky Embassy si prenajal vládny hotel pre oslavu.	Čínského velvyslanectví pronajmout vládní hotelu k oslavě.	Čínské velvyslanectví pronajal hotel vláda pro oslavu.	Do Šanghaje pronajaté vládou hotel na oslavu .
7.	We are in a situation where we have to do things together.	Sme v situácii, kedy musíme robiť veci spoločne.	Sme v situácii, kde sme robiť veci dohromady.	My sme v situáciikdemy musímeurobímeveci spolu.	Jsme v situaci, kdy musíme dělat věci společně.	Jsme v situaci, kdy musíme dělat věci dohromady.	Jsme v situaci , kdy musíme dělat věci dohromady .
8.	Our project is sound and workable.	Náš projekt je zdravý a funkčný.	Náš projekt je zvuk a funkčný.	Náš projekt má dobré jadro a výnosné.	Náš projekt je zdravý a funkční.	Náš projekt je zdravé a proveditelný.	Náš projekt je zvuk a proveditelný .

9.	Eurozone members are in the process of ratifying proposals.	Členovia eurozóny sú v procese ratifikácie návrhov.	Členov eurozóny sú v procese ratifikácie návrhy.	Eurozone členovia sú v procese ratifikujúcich návrhov.	Členové eurozóny jsou v procesu ratifikace návrhů.	Členové eurozóny jsou v procesu ratifikacenávrhy.	Zeměmi eurozóny jsou v procesu ratifikace návrhy.
10.	There has been renewed optimism this week.	Tam bol obnovený optimizmus tento týždeň.	Došloobnovená optimizmus tento týždeň.	Tambol obnovený optimizmus tento týždeň.	Tam byl obnovený optimismus v tomto týdnu.	Je tu nabytého optimismu- tento týden.	Bylo obnoveno optimismus tento týden .
11.	These helped to boost investor sentiment with stock markets.	Tieto pomohol k posilneniu investičného prostredia s akciovými trhmi.	Tieto pomohol oživiť investora sentiment s akciové trhy.	Tieto pomohol povzbudiť investor cit s trhmi s cennými papiermy.	Tyto pomohl k posílení investičního prostředí s akciovými trhy.	To přispělo k posílení smýšlení investorů s akciové trhy.	Tyto přispělo k posílení investor sentiment s akciových trhů .
12.	The company has limited itself to making black and white e-readers.	Spoločnosť sa obmedzila na výrobu čiernobielej e-čitatelia.	Spoločnosť sa obmedzila- tvorby čierne a biele e-- čitatelia.	Spoločnosť limitovaný samo o sebe tvorbe čiernobiely e- čitatelia.	Společnost se omezila na výrobu černobílé e-čtenáři.	Společnost se omezila výrobu černých a bílých e čtenáři.	Společnost se omezila na výrobu černých a bílých E - čtenářů .
Výsledok prekladu		**37,5 / 60 => 63 %**	**29,5 / 60 => 49 %**	**34 / 60 => 57 %**	**39 / 60 => 65 %**	**40,5 / 60 => 68 %**	**33,5 / 60 => 56 %**

Tabuľka A.1: Preklad anglických viet.

A.3. Zo slovenčiny a češtiny do angličtiny

V predchádzajúcej kapitole boli zadefinované určité anglické vzorky. Ich korektný manuálny preklad bol použitý ako základ pre spätné porovnanie výsledkov. Webové služby World Lingo a MOSES nepodporujú slovenský jazyk a preto bol použitý ich český ekvivalent.

		Anglický preklad				
č.	**Slovenská veta**	**Google Translate**	**Bing Translate**	**World Lingo ***	**MOSES ***	**PC Translator 2010**
1.	Odkiaľ pochádzate?	Where are you from?	From where are you from?	Where did you come from?	Where are you from?	From where come form?
2.	Mám matku.	I have a mother.	I have a mother.	I have a mother.	I have a mother.	Shall I mother.
3.	Čo robíte na Slovensku?	What are you doing in Slovakia?	What do you do in Slovakia?	What are you doing in Slovakia?	What do you do in Slovakia?	What are you about in Slovakia?
4.	Nebudeme blokovať ostatné krajiny.	We will not block other countries.	We do not block the other country.	We don't block the other countries.	We will not block the rest of the country.	Nebudeme block others landscapes.
5.	Neexistuje dohoda.	There is no agreement.	There is no agreement.	There is no agreement.	There is no agreement.	Don't exist agreement.

6.	Čínske veľvysla-nectvo prenajalo k oslave vládny hotel.	The Chinese embassy ren-ted the hotel to celebrate the gover-nment.	The Chinese Embassy prena-jalo to celebrate a Government hotel.	China's embas-sy around to celebrate go-vernment the hotel.	The Chinese Embassy prona-jalo to celebrate the government's hotel.	Chinese em-bassy rent to celebration government hotel.
7.	Sme v situácii, kedy musíme robiť veci spoločne.	We are in a situation whe-re we do things toget-her.	We are in a situ-ation where we have to do things together.	We're in a situ-ation, when we have to do things together.	We are in a situ-ation where we have to do things together.	Be sitionary, when we have to do thing together.
8.	Náš projekt je zdravý a funkčný.	Our project is a healthy and functional.	Our project is healthy and functional.	Our project is a healthy andto the functional.	Our project is healthy and functional.	Our projectbe ingood health and functional.
9.	Členovia eurozóny sú v procese rati-fikácie návrhov.	Members of the Eurozone are in the process of ratification of the draft.	Members of the euro area are in the process of ratification of the proposals.	Euro area members aret-he process of ratification proposals.	The members of the euro zone are in the pro-cess of ratifica-tion of the pro-posals.	Member euro area aretheyin the process of ratification proposals.
10.	Tento týždeň prišlo k obnoveniu optimizmu.	This week there has been renewed op-timism.	This week came to renewed optimism.	This week there was to restore but.	This week there has been a re-newed optimism.	This weekly come by reno-vation opti-mism.
11.	Tie pomohli k posilneniu inves-tičného prostredia s akciovými trhmi.	They help to strengthen the investment climate in equity mar-kets.	You have helped to strengthen the investment environment with stock mar-kets.	You have hel-ped to stren-gthen inves-tment environ-ment the stock markets.	Youhave helped to strengthen the investment environment in the stock mar-ket.	Those have helped to rein-forcement invested set-tings with stock market.
12.	Spoločnosť sa obmedzila na výrobu čiernobie-lych e-čítačiek.	The company is limited to the production of black and white e-readers.	The company is limited to the production of black and white e-readers.	The company has limited itself to the production black and white e-readers.	The company has limited itself to the production of black and white e-čteček.	Society oneself restrict to pro-duction black and white e-scaler.
Výsledok prekladu		**55 / 60 => 92 %**	**51 / 60 => 85 %**	**45 / 60 => 75 %**	**45 / 60 => 75 %**	**10 / 60 => 17 %**

Tabuľka A.2: Preklad slovenských viet.

Príloha B – Preklad paragrafov

B.1. Vysvetlenie označenia v prílohách

Pri tabuľkách, ktoré sa vyskytnú v nasledujúcich kapitolách, budú použité symboly. Ich význam je potrebný pre správnu orientáciu vo výsledkov.

Zoznam symbolov:

* - zadaný český výraz rovnakého významu

B.2. Z angličtiny do slovenčiny a češtiny

Sú dané tri ukážky paragrafov v anglickom jazyku, ktoré je potrebné preložiť. Tabuľka bola rozdelená do dvoch častí – slovenský a český preklad. Jej posledný riadok obsahuje percento úspešnosti prekladu jednotlivého paragrafu a aritmetický priemer týchto prekladov.

		Slovenský preklad		
č.	**Anglický paragraf**	**Google Translate**	**Bing Translate**	**PC Translator 2010**
1.	If you are happy to be contacted by a BBC journalist please leave a telephone number that we can contact you on. In some cases a selection of your comments will be published, displaying your name as you provide it and location, unless you state otherwise. Your contact details will never be published. When sending us pictures, video or eyewitness accounts at no time should you endanger yourself or others, take any unnecessary risks or infringe any laws. Please ensure you have read the terms and conditions.	Ak ste radi, že kontaktoval novinára BBC, prosím zanechať telefónne číslo, aby sme vás mohli kontaktovať ~~na~~. V niektorých prípadoch bude výber Vaše pripomienky uverejnené, zobrazí vaše meno, ktoré poskytujete, a miesto, pokiaľ neustanovuje inak. Vaše kontaktné údaje nebudú nikdy zverejnené. Pri odosielaní ~~nás~~ obrázkov, videa alebo očitých svedkov nikdy by ste ohroziť seba alebo iných, prijať akékoľvek zbytočné riziko alebo porušuje zákony.Prosím, aby ste si prečítali podmienky.	Ak máte radi dotknúť BBC novinár zanechajte telefónne číslo, ktoré sme vás kontaktovať na. V niektorých prípadoch sa uverejní výber vaše komentáre, ktoré zobrazuje vaše meno ako poskytnete a umiestnení, pokiaľ ste neurčia inak. Vaše kontaktné údaje nebude nikdy uverejnený. Pri odosielaní nás obrázky, video alebo ocitá svedok účtov v žiadnom okamihu by ste ohroziť sami seba alebo iné osoby, prijať akékoľvek nepotrebným rizikám alebo porušujú zákony. Uistite, ste si prečítali podmienky.	Ak vy ste šťastný, že byť kontaktovaný BBC novinárom prosím zanechajte telefónne číslo ktoré my môžeme kontaktovať vy ďalej. Prípadne výber vašich komentárov bude zverejnený, zobrazovanie vaše meno ako vy poskytujete to a pamäťové miesto, pokiaľ vy udáte cenu inak. Tvoj kontaktné detaily nikdy nebude publikované. Keď zaslanie nás obrazy, video alebo svedecké záznamy nikdy mal by si ohroziť sebe alebo iný, zobrať nejaké nepotrebné riziká alebo porušovať nejaké práva. Prosím vás zabezpečte , že vy čítali podmienky.
2.	At the age of 17 he had got his parents' permission to travel to Yemen to study Arabic. He briefly returned to California but couldn't settle so he headed back to Yemen from where he wrote to ask his father if he could go to Pakistan to continue his studies. Frank Lindh replied: "I trust your judgment and hope you have a wonderful adven-	V 17 rokoch dostal súhlas rodičov k ceste do Jemenu študovať arabčinu. On stručne sa vrátil do Kalifornie, ale nebola schopná~~vysporiadať~~, takže zamieril späť do Jemenu, odkiaľ písal~~sa spý-tať~~ otca, či by mohol	Vo veku 17 mal dostal jeho rodičia dovolenia cestovať do Jemenu študovať~~arabčina~~. Nakrátko vrátil do Kalifornie ale nemohol vyriešiť tak viedol späť na Jemen od kde~~napí-sal~~požiadať svojho otca, ak mu mohol ísť	Vo veku 17 on mal jeho rodičovské oprávnenie cestovať do Jemen študovať arabčinu. Vrátil sa do krátko Kalifornia ale nemôcť účtovať tak on vrátil sa späť do Jemen odkiaľ on písal žiada jeho otca či on môže skočiť na

	ture."	ísť do Pakistanu, aby pokračoval vo svojom štúdiu. Frank Lindh odpovedal: "Dúfam, že vaše rozhodnutie a dúfame, že budete mať nádherné dobrodružstvo."	do Pakistanu pokračovaťv jeho štúdie. Frank Lindh odpovedal: "Som veriť vášho rozsudku a dúfam, že ste krásnym dobrodružstvom."	Pakistan pokračovať jeho štúdie. Bez porta Lindh odpovedal: "Ja dôverujem vášmu úsudoka nádej, ktorú máte zázračné dobrodružstvo."
3.	China is investing billions of dollars in its space programme. It has a strong space science effort under way, with two orbiting satellites having already been launched to the Moon. A third mission is expected to put a rover on the lunar surface. The Asian country is also deploying its own satellite-navigation system known as Bei-Dou, or Compass.	Čína investuje miliardy dolárov do svojho vesmírneho programu.Má silné kozmickej vedy úsilie na ceste, s dvoma satelitov na obežnej dráhe, ktoré už začali na Mesiac.Tretí misie sa očakáva, aby rover na mesačnom povrchu. Ázijské krajiny ~~sa~~ tiež nasadením vlastný satelitný navigačný systém, známy ako Beidou, alebo Compass.	Čína je investujú miliardy dolárov do svojho priestoru programu. Má silné vesmírna veda úsilia prebiehajú, s dvoma obiehajúceho satelitov s už začala na mesiac. Očakáva sa, že tretia misia dať rover na povrchu. Ázijské krajiny~~je~~ tiež nasadenie svoj vlastný satelitný navigačný systém, známy ako BeiDou alebo kompas.	Čína investuje miliardy dolárov v ich priestorový program. To má silnú priestorovú vedeckú snahu v prúde, s dvoma obiehajúci satelity už bývajúc vypustenýMoon. Tretina rozkaz má opený predaj auto - rover na slabom povrchu. Ázijská krajina je taktiež nasadenie svoj vlastný satelitný- navigačný systém známy akoBei-Dou, alebo Compass.
Priemer výsledku prekladu [jednotlivé výsledky]		**86 % [84 % , 84 % , 91 %]**	**56 % [54 % , 54 % , 59 %]**	**50 % [50 % , 71 % , 28 %]**

Tabuľka B.1: Preklad anglických paragrafov do slovenského jazyka.

		Český preklad		
č.	**Anglický paragraf**	**Google Translate**	**Bing Translate**	**World Lingo**
1.	If you are happy to be contacted by a BBC journalist please leave a telephone number that we can contact you on. In some cases a selection of your comments will be published, displaying your name as you provide it and location, unless you state otherwise. Your contact details will never be published. When sending us pictures, video or eyewitness accounts at no time should you endanger yourself or others, take any unnecessary risks or infringe any laws. Please ensure you have read the terms and conditions.	Pokud jste rádi~~, že~~ kontaktoval novináře BBC, prosím zanechat telefonní číslo, abychom vás mohli kontaktovat ~~na~~. V některých případech bude výběr Vaše připomínky zveřejněny, zobrazí vaše jméno, které poskytujete, a místo, pokud nestanoví jinak. Vaše kontaktní údaje nebudou nikdy zveřejněny. Při odesílání ~~nás~~ obrázků, videa nebo očitých svědků nikdy byste ohrozit sebe nebo ostatní, přijmout jakákoli zbytečná rizika	Pokud máte rádi kontaktovat novináře BBC prosím zanechtetelefonního čísla, abychom vás mohli kontaktovat. V některých případech výběr vaše připomínky bude zveřejněno, zobrazování vaše jméno jako poskytnout místo a to pokud výslovně jinak. Vaše kontaktní údaje nebudou nikdy zveřejňují. Při odesílání nás obrázky, video nebo svědka účty v čas by ohrozit sebe nebo ostatní, podstupovat zbytečná rizika nebo porušujížádné ~~žádné~~ zákony	Pokud jste spokojeni se být kontaktován BBC novinář nechte prosím telefonní číslo, které můžeme kontaktovat. V některých případech výběr Vaše připomínky budou zveřejněny, zobrazování své jméno v poskytovat a umístění, pokud se stát jinak. Vaše kontaktní údaje nikdy nebude zveřejněno. Když nás posílá obrázky, video nebo očitých svědků v žádném okamžiku by ohrožuješ sebe nebo ostatní , učinit zbytečným rizikům nebo

		nebo porušuje zákony. Prosím, aby jste si přečetli podmínky.	ny.Ujistěte se, že jste si přečetli a podmínky.	porušení právních předpisů. Zkontrolujte si přečteš podmínky.
2.	At the age of 17 he had got his parents' permission to travel to Yemen to study Arabic. He briefly returned to California but couldn't settle so he headed back to Yemen from where he wrote to ask his father if he could go to Pakistan to continue his studies. Frank Lindh replied: "I trust your judgment and hope you have a wonderful adventure."	V 17 letech dostal svolení rodičů k cestě do Jemenu studovat arabštinu. On stručně se vrátil do Kalifornie, ale nebyla schopna vypořádat, takže zamířil zpět do Jemenu, odkud ~~psal~~ se zeptat otce, zda by mohl jít do Pákistánu, aby pokračoval ve svém studiu. Frank Lindh odpověděl: "Doufám, že vaše rozhodnutí a doufáme, žebudete mít nádherné dobrodružství."	Ve věku 17 let dostaljeho rodičepovolení k vycestování do Jemenu studovat arabsky. Krátce se vrátil do Kalifornie ale nemohlavyrovnat tak zamířil zpět do Jemenu z napsal, kde se zeptat svého otce, kdyby mohl jít do Pákistánu pokračovat ve studiu. Frank Lindh odpověděl: "Váš úsudek a doufám, že máte úžasné dobrodružství."	Ve věku 17 měl jeho rodičů povolení k vycestování do Jemenu studovat arabsky. Krátce vrátil do Kalifornie ale nemohl usadit tak zamířil zpět k Jemenu odkud napsal ~~zeptat na~~ jeho otec, kdybych mohl jít do Pákistánu pokračovat ve studiu. Frank Lindh odpověděl: "Věřím tvému úsudku a doufat, že máte úžasné dobrodružství."
3.	China is investing billions of dollars in its space programme. It has a strong space science effort under way, with two orbiting satellites having already been launched to the Moon. A third mission is expected to put a rover on the lunar surface. The Asian country is also deploying its own satellite-navigation system known as BeiDou, or Compass.	Čína investuje miliardy dolarů do svého vesmírného programu. Má silné kosmické vědy úsilí na cestě, se dvěmasatelitů na oběžné dráze, které již byly zahájeny na Měsíc.Třetí mise se očekává~~, abyrover~~ na měsíčním povrchu.Asijské země ~~se~~ také nasazením vlastní satelitní navigační systém, známý jako Beidou, nebo Compass.	Čína investuje miliardy dolarův jeho kosmického programu.Má silnou space science úsilí na cestě, s dvěma orbitální satelity, již byla zahájena na měsíc.Třetí mise se očekává rover na měsíční povrch. Asijské země ~~je~~ také nasazenísvé vlastní satelitní navigační systém, známý jako BeiDou nebo kompas.	Čína je investovat miliardy dolarův jeho kosmického programu. Má silné vesmírné vědě úsilí na cestě , se dvěma obíhající družice s již byla zahájena na měsíc.Třetí mise se očekává~~, že dal~~Rover na měsíčním povrchem . Asijské země rovněž umisťovat vlastní satelitní navigační systém zvaný beidou nebo kompas.
Priemer výsledku prekladu [jednotlivé výsledky]		**85 % [81 % , 85 % , 89 %]**	**72 % [63 % , 80 % , 73 %]**	**63 % [44 % , 89 % , 56 %]**

Tabuľka B.2: Preklad anglických paragrafov do českého jazyka.

B.3. Zo slovenčiny a češtiny do angličtiny

Spôsob použitý pri vetách sa teraz aplikuje na časti článku. Vytvorený korektný preklad je použitý ako počiatočná báza. Na preložený odsek sa aplikujú pravidlá a dosiahnuté výsledky sa zaznačia do tabuľky.

		Anglický preklad				
č.	**Slovenský paragraf**	**Google Translate**	**Bing Translate**	**World Lingo ***	**PC Translator 2010**	**MOSES Translate ***
1.	Ak by ste radi kontaktovali	If you would like to contact the BBC	If you would like to contact	If you would like contac-	IF you'd radial road contac-	If you would like to have

	BBC novinára, prosím zanechajte telefónne číslo, aby Vás mohol kontaktovať. V niektorých prípadoch môžu byť Vaše komentáre zverejnené, zobrazí sa meno, ktoré ste poskytli, a miesto, pokiaľ neuvediete inak. Vaše kontaktné údaje nebudú nikdy zverejnené. Poslaním obrázkov, videa alebo účtov očitých svedkov môžete ohroziť seba alebo ostatných a podstupujete nežiaduce riziko ani neporušujete zákon. Prosím, aby ste si prečítali podmienky.	journalist, please leave a phone number so that you can contact. In some cases your comments may be published, the name you provided, and place, unless you indicate otherwise. Your contact details will never be published.The mission of images, video or eyewitness accounts, you could endanger yourself or others, and run the risk of adverse or break the law. I ask that you read the conditions.	a BBC journalist, please leave a phone number that You could contact you. In some cases may be your comments published, you will see the name that you provided, unless you specify otherwise, and location. Your contact details will never be disclosed. The Mission of the images, video, or you can endanger themselves or other witness accounts of the očitých and podstupujete adverse risk nor the neporušujete law. Please read the terms and conditions.	tedthe BBC journalist, please leave your phone number, to you to contact. In some cases, they may be your comments published, displays the name, which you provided, and the place, where do otherwise.Your contact data shall not be never published. The mission pictures, video or accounts eyewitness accounts you can endanger himself or others, and you're taking undesirable risk or smart law. Please, ~~you to~~ read conditions.	ted BBC journalist, please resignation telephone number, that you has been allowed contact. In some cases be able to Your comments made public, will portray oneself the name of, who are offer, and place, as far as nondisclosure otherwise. Your contact poop will not never made public. Sent images, videos or accounts ocular witness may endanger myself or of others and substage objectional risk nor' intact law. Please, so that you yourself read - outs conditions.	contacted the BBC journalist, please leave a telephone number, so that you can contact. In some cases, your comments may be published, shows the name that you have provided, and the place where the options otherwise. Your contact details will never be made public. The mission of the pictures, videos or accounts eyewitnesses can endanger yourself or others, and you run the risk not breaking the law or undesirable. Please, you read our terms.
2.	V 17 rokoch dostal súhlas rodičov k ceste do Jemenu študovať arabčinu. Nakrátko sa vrátil do Kalifornie, ale nebol schopný usídliť sa, takže zamieril späť do Jemenu, odkiaľ	In 17 years, received parental permission to travel to Yemen to study Arabic. Briefly returned to California, but was unable to settle, so headed back to Yemen, where he asked his father if he could go to	In the 17 years he received the consent of the parents on the road to Yemen to study Arabic. Briefly he returned to California, but was not ca-	In 17 years he got consent of the parents to the road to Yemen to study Arabic.For a short while he went back to California, but he was not able	Volts 17 ages got parental consent to ways to Jemenu study Arabic. Briefly oneself handed back to in Kalifornie, but he was unablesettle, ac-	In the 17 years he got parental consent to the trip to Yemen to study Arabic. Briefly returned to California, but was unable to settle ~~elsew-~~

	sa spýtal otca, či by mohol ísť do Pakistanu, aby pokračoval vo svojom štúdiu. Frank Lindh odpovedal: "Verím v tvoje rozhodnutie a dúfam, že zaži-ješ nejaké dob-rodružstvo."	Pakistan to contin-ue their studies. Frank Lindh said: "I believe in your decision and hope that you'll expe-rience an adventu-re."	pable of ta-king up resi-dency, headed back to Ye-men, from where he asked his fat-her if he could go to Pakistan to continue their study. Frank Lindh replied: "I believe in your decision and I hope that zaziješ some adven-ture."	to settle, so he went back to Yemen, where he asked his fat-her, whether he could go to Pakistan, to continue the studio. Frank 8, and said: "I believe in your decision, and I hope I find some adventure."	cordingly he-aded back to Jemenu, from where oneself questioningly father, or might be go-ing to Pakista-nu, that he kept on at of his stu-dio.Franconia n Lindh rep-lied: "Believe inyour deci-sion and hope that the expe-rience some adventure."	~~here~~, so he headed back to Yemen, where he said his father, whether he could go into Pakistan to continued his studies. Frank Lindh said: "I believe in your decision and I hope that desire Being an ad-venture."
3.	Čína investuje miliardy dolárov do svojho ves-mírneho prog-ramu. Silné úsilie kozmickej vedy je v plnom prúde, dva sate-lity na obežnej dráhe už vyštar-tovali na Me-siac. Od tretej misie sa očaká-va auto na me-sačnom po-vrchu. Ázijské krajiny tiež nasadzujú vlast-ný satelitný navigačný sys-tém, známy ako Beidou, alebo Compass.	China is investing billions of dollars into its space prog-ram. Strong efforts space science is in full swing, two satellites alreadyin orbit to the moon vyštartovali. From the third car of the mission is expec-ted on the lunar surface.Asian co-untries also deploy its own satellite navigation system known as Beidou, or Compass.	People's Re-public of Chi-na is investing billions of dollars into its space prog-ramme. The strong efforts of the interna-tional space science is in full swing, two satellites in orbit already vyštartovali for a month. From the third mission to the lunar surface is expected to be a car. Asian countries also are rolled out its own satelli-te navigation system, known as Beidou, or Compass.	China invests billions of dollars into the space program. Strong efforts space science is in full swing, two satellites in orbit for I to the moon. From the third mission, the expected the car in the lunar surface. Asian coun-tries also dep-loys its own satellite navi-gation system, known as beidou, or compass.	China invests zillion dollars in your cosmic program. Strong effort cosmic scien-ce be under-way, two sa-tellites on circulating road for be off on moon. Away from the the third station one-self expects car on mon-thly surface. Asiatic lan-dscapes too risk own satel-litic nautical the system, known as Beidou, or Compass.	China is inves-ting billions of dollars into its space prog-ram. Strong efforts of spa-ce science is in full swing, two satellites in orbit ~~longer~~ took off for the moon. The third mission is expected from a car on the surface of the moon. Asian coun-tries also dep-loys its own satellite navi-gation system, known as Beidou, or Compass.

Priemer výsledku prekladu [jednotlivé výsledky]	**84 % [87 %, 93 %, 72 %]**	**68 % [64 % , 86 % , 55 %]**	**65 % [42 % , 86 % , 67 %]**	**7 % [0 % , 21 % , 0 %]**	**87 % [81 % , 88 % , 93 %]**

Tabuľka B.3: Preklad slovenský paragrafov do anglického jazyka.