Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informačné systémy

Bc. Jozef Hergott

ALGORITMICKÉ ZISŤOVANIE ZÁKLADNÉHO TVARU SLOV V SLOVENČINE

Diplomový projekt I

Vedúci diplomového projektu: RNDr. Michal Laclavík PhD.
Pedagogický vedúci: Ing. Ivan Kapustík

máj, 2008

Obsah

ÚVOD....2

1 ZÍSKAVANIE ZÁKLADNÉHO TVARU SLOVA....4

1.1 Lematizácia....4
1.2 Stemming....4
1.3 Lematizácia vs. stemming....5
1.4 Prístupy k tvorbe lematizátorov a stemmerov....5
1.5 Spracovanie slovenčiny....5

2 SÚČASNÉ RIEŠENIA PRE SLOVENSKÝ JAZYK....7

2.1 Google....7
2.2 Morfeo....7
2.3 Forma....7
2.4 Slovenský morfologický analyzátor (JÚĽŠ SAV)....8
Zhodnotenie....9
2.5 Tvaroslovník (PF UPJŠ)....9
Zhodnotenie....10

3 MOŽNOSTI RIEŠENIA ALGORITMICKÉHO STEMMERA....11

3.1 Lingvistické zdroje....11
3.2 Snowball....12
3.3 Egothor a Stempel....12
Popis algoritmu....12

4 ZÁVER....14

POUŽITÁ LITERATÚRA....15

Úvod

Tento dokument je priebežnou správou o riešení diplomového projektu s témou *Algoritmické zisťovanie základného tvaru slov v slovenčine*. Práca sa zaoberá problematikou vyhľadávania v informačných zdrojoch, ktoré obsahujú slová v gramaticky vyskloňovaných tvarov. V súčasnej dobe nepreberného množstva dokumentov dostupných na Internete sa stáva ich strojové spracovanie nevyhnutnosťou pre získavanie informácií. Väčšina zdrojov na Internete je však dostupná v prirodzenom jazyku, ktorý výraznejšie komplikuje ich spracovanie a vyhľadávanie z dôvodu nejednoznačnosti a istej miery vágnosti.

Zo súčasnej informatickej praxe vyplýva, že problémy, ktoré sú zložité pre človeka (napr. výpočtovo náročné úlohy), sú dobre riešiteľné počítačovým spracovaním. Toto však platí aj naopak – činnosti, ktoré pre človeka nepredstavujú problém (najmä tie, pri ktorých sa využívajú mentálne schopnosti) sú pre počítače stále výrazným problémom. Do tejto oblasti patrí aj porozumenie prirodzeného jazyka. [1]

Prirodzený jazyk ako najdokonalejší nástroj na zdieľanie informácií medzi ľuďmi predstavuje najlepší spôsob komunikácie pre ľudí. V ideálnom prípade by sme s počítačom komunikovali práve v prirodzenom jazyku. Komunikácia s vyhľadávacím systémom by potom prebiehala nasledovne: na otázku v prirodzenom jazyku, napr.: „Koľko obyvateľov má Bratislava?“, by vyhľadávač vrátil jednu konkrétnu odpoveď. Súčasné vyhľadávacie systémy však nemajú schopnosť chápať význam dopytov. Výsledkom vyhľadávania je zoznam množstva stránok zoradených podľa relevantnosti (podľa metodiky vyhľadávača), kde sa táto informácia dá pravdepodobne získať.

Súčasné vyhľadávacie systémy fungujú hlavne na štatistickom princípe – dokumenty sa spracúvajú indexovaním slov dokumentov. Samotné vyhľadávanie je fulltextové – slová nachádzajúce sa v dopyte sa hľadajú v indexovaných dokumentoch. Vzhľadom k tejto skutočnosti nebol predchádzajúci modelový dopyt zadaný najsprávnejšie. Korektnejší dopyt by vyzeral takto: „obyvatelia Bratislava počet“. Tento dopyt je zadaný prostredníctvom kľúčových slov a zohľadňuje pravdepodobný výskyt týchto slov vo vyhľadávaných dokumentoch.

Čo však v prípade, že hľadaná stránka by obsahovala slová „počet obyvateľov Bratislavy“? Ak by vyhľadávací systém nezohľadňoval možnosť výskytu slov v dokumente aj dopyte v rôzne vyskloňovaných tvaroch, hľadanú stránku by nám nenašiel. S tohto príkladu vyplýva jeden z hlavných aktuálnych problémov oblasti vyhľadávania informácií: vyhľadávanie v gramaticky ohýbaných tvaroch.

K získavaniu základného tvaru slova existujú dva rôzne prístupy, ktoré sa líšia výstupom: *lematizácia* produkuje slová v základných tvaroch (napr. pre podstatné mená typicky nominatív jednotného čísla [2]), *stemming* upravuje slová do tvaru slovného základu, pričom výstup stemmingu nemusia byť gramaticky korektné slová. Zvyčajne stačí, aby sa slová s rovnakým slovným základom projektovali na rovnaký koreň [3].

Problém potreby získavania základného tvaru slova sa objavil spolu s prvými systémami vyhľadávania informácií. Prvú prácu ohľadom tejto problematiky publikovala v roku 1968 Julie Beth Lovinsová. Lovinsovej stemmovací algoritmus pre anglický jazyk funguje na princípe orezávania 294 dopredu definovaných koncoviek. Tento algoritmus výrazne

ovplyvnil vývoj v tejto oblasti. Ďalším výrazným počinom v tejto oblasti bolo uverejnenie návrhu stemmovacieho algoritmu Martina Portera v roku 1980. Tento algoritmus orezáva koncovky postupne v piatich krokoch pričom musia byť dodržané zadefinované podmienky orezávania. Porterov algoritmus sa stal pre anglický jazyk najpoužívanejším a je de-facto štandardom pre stemming anglického jazyka.

Stemmovacie algoritmy sú jazykovo špecifické, preto sa algoritmy pre anglický jazyk nedajú aplikovať na stemming slovenčinu. Na vývoji obdobných algoritmov pre slovenský jazyk sa pracovalo, výskum však narážal na problém prílišnej komplikovanosti slovenského jazyka (množstvo výnimiek z pravidiel skloňovania a pod.). Preto sa vývoj v tejto oblasti ubral smerom k slovníkovému spracovaniu. Na výskumných projektoch z tejto oblasti sa pracuje na Jazykovednom ústave Ľudovíta Štúra Slovenskej akadémie vied (JÚĽŠ SAV) a na Prírodovedeckej fakulte Univerzity Pavla Jozefa Šafárika (PF UPJŠ). Slovníkové lematizátory sú nasadené aj v komerčnej sfére – vlastným riešením slovníkového stemmera sa môže pochváliť firma Forma s. r. o. Ďalej slovníkový lematizátor pre slovenský jazyk využívajú vyhľadávače *google.sk* a *morfeo.sk*. Problémom slovníkového prístupu je však nemožnosť spracovania slov mimo slovníka ako sú bežne napr. vlastné mená[1]. Preto slovníkové lematizátory nie sú najvhodnejšie pre použitie v rámci vyhľadávania informácií.

Cieľom tohto projektu je navrhnúť a implementovať stemmer slovenského jazyka, ktorý by umožnil algoritmické získavanie základného tvaru slov. Zámerom práce je vytvoriť stemmer aplikovateľný aj na spracovanie vlastných mien a novotvarov slov. Ďalej je cieľom projektu poskytnúť výsledky práce vo voľne dostupnej forme na ďalšie využitie v projektoch z oblastí vyhľadávania a extrakcie informácií.

V prvej kapitole práce si povieme o metódach a prístupoch k získavaniu základného tvaru. Prvá kapitola poskytuje prehľad týchto metód, pričom predstavuje ich princípy a definuje základné pojmy. Ďalej sa v tejto kapitole zameriame na porovnanie týchto prístupov a poukážeme na ich výhody a nevýhody.

Druhá kapitola sa venuje opisu súčasných riešení, snaží sa poukázať na nedostatky týchto riešení. V rámci tejto kapitoly sa detailne zameriame na akademické projekty z oblasti lematizácie a stemmingu – Slovenský morfologický analyzátor vyvíjaný na JÚĽŠ SAV a Tvaroslovník z PF UPJŠ.

V tretej kapitole si predstavíme možnosti riešenia a platformy pre implementáciu algoritmického stemmera.

1 http://sk.wikipedia.org/wiki/Vlastn%C3%A9_meno

1 Získavanie základného tvaru slova

Vyhľadávanie v informačných zdrojoch v prirodzenom jazyku komplikuje fakt, že slová rovnakého významu sa vyskytujú v texte v rôznych morfologických tvaroch. Vo všeobecnosti problém nastáva v prípade, že slová vo vyhľadávacej fráze sú ohýbané inak ako slová v hľadaných dokumentoch. Napr. v prípade vyhľadávania frázy „Košické Hlavné námestie" mám záujem aj o výsledky obsahujúce výraz „Hlavné námestie Košíc", „Hlavné námestie (mesta) Košice" a pod. V prípade, že sú slová indexované tak, ako sa nachádzajú v dokumente, vyhľadávač nevráti dokumenty, ktoré neobsahujú presné tvary slov vyhľadávacej frázy.

Preto je výhodné reprezentovať rôzne morfologické tvary rovnakého slova jedným zastupujúcim termom – základným tvarom slova. Metódami pre získavanie základného tvaru slova sú *lematizácia* a *stemming*. [3]

Spracúvanie slov do podoby základných tvarov je nevyhnutné v oblastiach vyhľadávania informácií, extrakcie informácií, sémantickej anotácie, identifikáciu informačných zdrojov v rámci domény a ďalšie. [4]

1.1 Lematizácia

Lematizácia predstavuje metódu získania základného tvaru slova (*lema*). Tento tvar sa tiež dá definovať ako slovníkový tvar slova [2]. V rámci morfológie sa lema chápe ako kanonická forma lexém[2] slova [3].

Lematizátor sa dá vytvoriť dvomi prístupmi – slovníkovou metódou a algoritmicky. V prvom prípade sa získa lema zo slovníka, ktorý priraďuje odvodeným a ohýbaným slovám slová v základnom tvare. Rozsah takéhoto slovníka dnes už nie je problémom. Problémom však ostáva neschopnosť spracovania slov, ktoré sa v slovníku nenachádzajú. Alternatívne sa dá lematizátor riešiť algoritmicky definovaním skloňovacích pravidiel, ktoré upravia slovo do základného tvaru.

1.2 Stemming

Stemming je proces, ktorý umožňuje získať zo slova jeho koreň (*stem*). Táto časť slova je rovnaká pre všetky morfologické varianty slova. Koreň je zvyčajne slovný základ (morfologický koreň, resp. kmeňová morféma). Nie však nevyhnutne – koreň získaný stemmingom nemusí byť zhodný s morfologickým koreňom slova [3]. Koreň nemusí byť ani gramaticky korektným slovom. Dôležité je, aby sa morfologické tvary jedného slova mapovali na jedného reprezentanta. Toto maximalizuje odozvu systému vyhľadávania informácií za cenu nižšej presnosti. [5]

Rovnako ako lematizátor aj stemmer sa dá vytvoriť spomenutými postupmi. Ako vhodnejšie riešenie sa javí použitie algoritmického stemmera, ktorý spracuje slovo do tvaru slovného základu s využitím pravidiel stemmingu, ktoré sú špecifické pre každý jazyk.

Výstižne popisuje výhody stemmingu pre slovanské jazyky A. Bialecki na stránkach projektu Stempel:

2 Lexéma je abstraktná jednotka reprezentujúca rôzne formy rovnakého slova [].

„*Stemming ako metóda pre získanie jedného slovného základu pre rôzne vyskloňované tvary slova predstavuje dôležitú súčasť viacerých systémov vyhľadávania informácií. Zlepšuje odozvu systému a zmenšuje veľkosť indexov. Toto platí predovšetkým pre jazyky s bohatou morfológiou ako sú slovanské jazyky (čeština, poľština, slovenčina, ruština, bulharčina, atď.*).“ [6]

1.3 Lematizácia vs. stemming

Výhodou lematizácie je presnosť, cenou za presnosť je však menšie pokrytie. Naopak, vyhľadávanie založené na slovných základoch – stemmingu – má oproti vyhľadávaniu základných tvarov slov nižšiu mieru presnosti (napr. pri slovách, ktoré sa líšia len prítomnosťou koncovej samohlásky: rad – rada), no na rozdiel od lematizácie stemming poskytuje vyššie pokrytie. Nižšia presnosť nemusí byť nevýhodou, lebo relevantnosť výsledku sa v prípade vyhľadávania prostredníctvom viacerých slov dá posúdiť z ostatného kontextu dokumentu.

Ďalším problémom lematizácie je, že bez doplnkovej informácie nie je možné všetkým slovám jednoznačne priradiť lemu, resp. pre niektoré slová je relevantných viacej lem (napr. pre slovo rád sú relevantné lemy rada aj rád). Riešenie tohto problému by si vyžadovalo zapojenie lingvistického (syntaktického analyzátora) alebo logického modulu do procesu lematizácie.

1.4 Prístupy k tvorbe lematizátorov a stemmerov

Ako s predošlého textu vyplýva, k tvorbe lematizátorov a stemmerov existuje viacero prístupov. V zásade sa líšia využívaním sloníka a pravidiel k získavaniu základného tvaru. Najčastejšie používané prístupy sú nasledujúce:

- *brute-force (slovníkový) prístup* – na získanie základného tvaru sa využíva slovník, ktorý má ohýbané slová ako kľúče a lemy ako prislúchajúce hodnoty;
- *odrezávanie koncoviek* – zo spracúvaného slova sa algoritmicky orezávajú koncovky na základe definovaných pravidiel;
- *stochastický algoritmus* – vytvárajú pravdepodobnostný model na základe známych relácií medzi koreňom a morfologickými tvarmi slova [3];
- *morfologická analýza (na základe POS taggingu)* – slová sa spracúvajú na základe informácie o skloňovaní, ktorá je dostupná v podobe morfologickej značky (*tag*).

1.5 Spracovanie slovenčiny

Slovenčina patrí medzi flexívne jazyky (slová sa ohýbajú pomocou koncoviek). S tým súvisí aj bohatá morfológia slovenského jazyka – väčšina slov má viacero rôznych tvarov v závislosti od použitia vo vete.

V angličtine a slovenčine je proces získavania základného tvaru diametrálne odlišný. V anglickom jazyku exituje na ohýbanie slov zo slovného základu niekoľko jednoduchých pravidiel (existujú aj výnimky z pravidiel – napr. nepravidelné slovesá – no je ich relatívne málo). Obvykle sa angličtina stemmuje, čo spočíva v odstraňovaní niektorej z mála prípon

(napr. *-ed*, *-ing*, *-s*). [7]

V slovenčine a iných flexívnych jazykoch sú pravidlá morfológie podstatne zložitejšie a izolácia koreňa sa rieši morfologickou analýzou a komplexným lingvistickým prístupom [8].

Pokus o zachytenie pravidiel skloňovania slovenského jazyka v podobe logického programu, ktorý by súčasne zachytával aj všetky nepravidelnosti skloňovania, vykonal v 90. rokoch bývalý matematik Emil Páleš[3] [4]. Táto práca však nechtiac potvrdzuje dohady o neefektívnosti hľadania pravidiel skloňovania [7]. Na túto prácu nadviazali na PF UPJŠ pri návrhu lematizátora slovenského jazyka. Táto cesta sa však ukázala ako nerealizovateľná vzhľadom na obrovské množstvo výnimiek. Prijali preto inú paradigmu a nástroj Tvaroslovník založili na slovníku tvoriacom rozsiahlu databázu vyskloňovaných slov slovenského jazyka (vychádza z elektrifikovanej verzie Slovníka slovenského jazyka) a algoritme, ktorý zisťuje podobnosť spracúvaného slova s tvarmi v slovníku [4].

Možnosť nájdenia všetkých tvarov slov a uloženia do databázy dlhšie neprichádzala do úvahy. Oficiálna slovná zásoba – Slovník slovenského jazyka obsahuje 150 000 slov ktoré majú spolu zopár miliónov tvarov. Uložiť do databázy zopár miliónov záznamov dnes už nie je nerealizovateľný problém. [7]

Spolu s rýchlym vývojom výpočtovej techniky toto umožnilo realizáciu lematizátorov založených na slovníkovom princípe. Touto cestou sa vydali projekty Slovenský morfologický analyzátor (JÚĽŠ SAV) a Tvaroslovník (PF UPJŠ).

Oba základné prístupy – slovníkový aj algoritmický (pravidlový) – však narážajú v slovenčine na spomenuté problémy. Ideálnym prístupom k tvorbe stemmera pre slovenský jazyk sa preto javí syntéza slovníkového a algoritmického prístupu[4]. Výhodou takéhoto prístupu je, že slovníková časť zabezpečuje vysokú presnosť stemmingu pre slová obsiahnuté v slovníku, jej prítomnosť zároveň rieši aj problém potreby množstva pravidiel pre podchytenie skloňovania v algoritmickom prístupe. Algoritmus zase zabezpečuje správnu funkciu pre slová mimo slovníka, pričom netreba definovať pravidlá.

3 PÁLEŠ, E.: Sapfo – parafrázovač slovenčiny. Bratislava: Veda, vydavateľstvo SAV, 1994.

4 Do tejto kategórie patria napr. stochastické algoritmy, možno sem zaradiť aj algoritmus Tvaroslovníka.

2 Súčasné riešenia pre slovenský jazyk

Hoci zatiaľ neexistujú dostupné otvorené riešenie slovenského stemmera, existujú viaceré komerčné riešenia. Slovenské stemmery na platformách Windows aj UNIX ponúka firma Forma, s. r. o. Stemmery pre vyhľadávanie v slovenčine používajú aj vyhľadávače Google a Morfeo.

Na vývoji stemmera pre slovenský jazyk sa pracuje aj na akademickej pôde. Najvýznamnejším pracoviskom na Slovensku, ktoré sa zaoberá spracovaním slovenčiny je Jazykovedný ústav Ľudovíta Štúra Slovenskej akadémie vied. Na tomto pracovisku sa tvorí Slovenský národný korpus a nástroje na jeho automatickú morfologickú anotáciu. Takýto systém sa stal základom pre vytvorenie slovníkového morfologického analyzátora, ktorý súčasne slúži aj ako lematizátor.

V rámci projektu NAZOU je na PF UPJŠ v Košiciach vyvíjaný nástroj Tvaroslovník. Tiež sa jedná o lematizátor, tento však nepracuje na slovníkovom princípe, ale na základe vzorov. Vzhľadom na použitie vzorov na odvodzovanie lemy je pravdepodobná vhodnosť jeho použitia aj na vlastné mená.

Spomenuté nástroje, ktoré vzišli z akademických výskumných projektov, však zatiaľ nie sú voľne dostupné.

2.1 Google

Google implementoval slovenský stemmer v roku 2003. Vzhľadom na neschopnosť spracovať niektoré vlastné mená sa jedná pravdepodobne o slovníkové riešenie. Na vyhľadávanie bežných, frekventovaných slov funguje uspokojivo, no vracia rozličné počty stránok pre rôzne tvary toho istého slova. Google pri vyhľadávaní berie do úvahy stem slova, no pri vyhľadávaní zohľadňuje aj presný tvar slova. Poznať to pri vyhľadávaní rôznych tvarov jedného slova, pri ktorých vracia rôzne počty nájdených stránok a rôzne poradie výsledkov. Výsledky na prvých miestach sú však vo väčšine prípadov relevantné, čo môže byť dôležitejšie ako veľké pokrytie s nepresnými výsledkami.

2.2 Morfeo

Možnosťou vyhľadávania v slovenčine so zohľadnením skloňovania sa pýši aj vyhľadávač Morfeo. Výhodou je, že rovnako ako Google pri vyhľadávaní zohľadňuje aj konkrétny hľadaný tvar slova. Rovnako však naráža na problém vyhľadávania vlastných mien, čo svedčí o použití slovníkového lematizátora.

2.3 Forma

Systém vyhľadávania v dokumentoch v slovenskom jazyku od tejto firmy nie je voľne dostupný na vyskúšanie, je však využívaný na vyhľadávanie na stránke www.zbierka.sk. Prítomnosť stemmera slovenčiny na tomto portáli potvrdilo pokusné vyhľadávanie: pri vyhľadávaní rôznych vyskloňovaných tvarov výrazu „Národná rada“ bol počet nájdených dokumentov v rozsahu 6075 - 6077, pričom výsledky vyhľadávania na prvých priečkach boli totožné. Vyhľadávací systém tejto firmy však nezvláda spracovanie priezvisk – v prípade vyhľadávania priezviska „Hrušovský“ a vyskloňovaného tvaru „Hrušovského“ nevrátil

rovnaké výsledky. V druhom prípade nevrátil dokonca žiadne výsledky. Toto svedčí o použití jednoduchého slovníkového prístupu.

2.4 Slovenský morfologický analyzátor (JÚĽŠ SAV)

Tento systém nie je primárne vyvíjaný pre využitie v oblasti IR. Hlavnou úlohou systému je využitie v procese určovania morfologickej anotácie – slovu sú priradené atribúty lema a tag [2]. Systém sa úspešne používa pri automatickej morfologickej anotácii databázy SNK.

Systém *morphology_levenshtein* je morfologickým generátorom a analyzátorom. Generátor je pre každú lemu je schopný vygenerovať všetky tvary spolu s príslušnými morfologickými značkami. Vychádza z myšlienky, že rovnaká postupnosť Levenshteinových operácií na transformáciu lemy do vyskloňovaného tvaru môže byť použitá na transformáciu lem patriacich do jednej skloňovacej paradigmy. Na základe skloňovacích paradigiem umožňuje systém skloňovanie lem prostredníctvom Levenshteinových operácií[5].

Lematizátor a morfologický analyzátor je z tohto systému získaný vygenerovaním všetkých tvarov slov a následným vložením do databáz *form2lemma.cdb* a *form2taglemma.cdb* – spolu so slovom sa ukladá morfologická značka a informácia o príslušnej leme. [5, 9]

Vyskloňované slová sa ukladajú celkovo do troch konštantných databáz CDB, ktoré tvoria jadro systému:

- *form2lemma.cdb* – tabuľka obsahuje slová ako kľúče a lemy ako príslušné hodnoty;
- *form2taglemma.cdb* – tabuľka obsahuje slová ako kľúče a morfologické značky spolu s lemami ako prislúchajúce hodnoty;
- *lemma2tagforms.cdb* – tabuľka obsahuje lemy ako kľúče a morfologické značky spolu s vyskloňovanými tvarmi ako prislúchajúce hodnoty.

[5]

V aktuálnej verzii systém obsahuje všetky slová z 3. edície Krátkeho slovníka slovenského jazyka a najfrekventovanejšie slová z SNK. Toto tvorí 56 000 lem, ktoré sú skloňované 1 365 paradigmami, čo tvorí približne 600 000 jedinečných slov. [5]

Samotný program sa skladá z dvoch častí. Prvá časť zabezpečuje vytváranie tabuliek skloňovacích paradigiem a zoznamov, ktoré priraďujú jednotlivé slová do týchto paradigiem. Druhá časť zabezpečuje spracovanie používateľských dopytov a predstavuje jednoduchý *wrapper* nad knižnicou na prístup k databáze. Táto časť zabezpečuje aj spracovanie predpôn naj- a ne- pre vytváranie superlatívov a negácií slovies. [5, 9]

Princípy činnosti generátora nebudem bližšie popisovať, spolu s implementačnými záležitosťami (formát vstupných súborov, opis API) sú detailne popísané v zdrojoch [5, 9].

Stemmovací algoritmus funguje nasledovne: z lemy, ktorú získa z databázy *form2lemma.cdb* odreže samohlásky, ktoré nasledujú za poslednou spoluhláskou. V prípade slovies najprv odstráni koncovku *-ť*, potom pokračuje odrezávaním samohlások podobne ako v prípade ostatných slov. [5]

Generátor je vyvíjaný v jazyku Python. Prístup k exportovaným databázam je možný prostredníctvom vyššieho API v jazyku Python alebo C. [9]

5 Levenshteinove operácie sú vloženie, zmazanie alebo zámena znaku; podrobnejšie popísané v [5, 9].

Na systéme sa stále pracuje – dopĺňa sa slovná zásoba. Cieľom do budúcnosti je doplniť do slovníka najfrekventovanejšie skratky a vlastné mená (geografické názvy a mená ľudí). Na zvýšenie pokrytia pri hľadaní morfologických značiek slov bude v ďalšej verzii implementovaný tipovací modul, ktorý bude odhadovať pravdepodobný tag pre neznáme slová na základe podobnosti koncoviek so slovami v slovníku. [5]

Zhodnotenie

Prístup k tvorbe tohto lematizátora je akoby opačný v porovnaní so štandardným prístupom k tvorbe algoritmického stemmera. Nedefinujú sa pravidlá získania základného tvaru z vyskloňovaného tvaru, ale definované sú pravidlá pre generovanie vyskloňovaných tvarov. Tieto pravidlá sú vzájomne inverzné, čiže by bolo z definovaných paradigiem možné získať stemmovacie pravidlá. Problémom by však bolo určenie podmienok pre aplikáciu pravidiel, čo sa dá však obísť použitím všetkých aplikovateľných (napr. rovnaká koncovka) a kontroly výsledku v slovníku základných tvarov[6]. Slovníkový prístup má však v tomto prípade aj výhodu v podobe šetrenia strojového času – vyhodnotenie dopytu v konštantnej databáze je menej časovo náročné ako spracúvanie reťazca.

Výraznejším nedostatkom stemmera je neschopnosť spracovať slová, ktoré sa nenachádzajú v databáze. Riešením bude plánovaná implementácia tipovacieho modulu – tento bude odhadovať tagy pre slová, ktoré sa nenachádzajú v slovníku [5]. Rovnako bude vítaným rozšírením aj doplnenie slovníka o vlastné mená. S týmito rozšíreniami má tento analyzátor možnosť stať sa univerzálne použiteľným nástrojom pre stemming slovenčiny.

2.5 Tvaroslovník (PF UPJŠ)

V rámci projektu NAZOU sa tvorí nástroj na spracovanie informačných zdrojov z internetu. Pilotná aplikácia – spracovanie pracovných ponúk – sa potýkala s problémom výskytu slov v dokumentoch vo vyskloňovaných tvaroch. Problém tvorili hlavne názvy miest, obcí, firiem, pracovných pozícií alebo kategórií pracovných ponúk. Rovnako aj meno a priezvisko uchádzača sa môže v informačných zdrojoch vyskytovať vyskloňované. Tu sa ukázala nevyhnutnosť využitia modulu pre získavanie základného tvaru slov pri spracovaní informačných zdrojov. Na riešenie tohto problému sa vytvára nástroj na jednoduchú lematizáciu slovenčiny – Tvaroslovník. [4]

Prvá verzia Tvaroslovníka bola pokusom o definovanie a využitie pravidiel skloňovania na získavanie základného tvaru. Lematizačný algoritmus fungoval nasledovne: najprv sa vytvoril zoznam aplikovateľných pravidiel pre slovo – ak slovo, ktoré spracúval, končilo na koncovku, ktorá sa vyskytovala v pravidle, pravidlo bolo pridané do zoznamu aplikovateľných. V ďalšom kroku sa aplikovali jednotlivé pravidlá zo získaného zoznamu a výsledok sa kontroloval v zozname základných tvarov. Ani takmer 900 pravidiel nestačilo na dostatočný popis slovenskej morfológie – pritom tieto pravidlá riešili len podstatné mená. Tento prístup sa ukázal ako neefektívny. Výsledkom tohto pokusu bolo zistenie, že hľadanie a aplikácia pravidiel ohýbania slov je v slovenčine zložitejšie ako nájdenie všetkých tvarov slov. Vývoj lematizátora sa preto priklonil k druhej paradigme.

Aktuálna verzia systému je teda založená na slovníku ohýbaných tvarov slov, je však realizovaná ako algoritmický lematizátor. Algoritmus lematizátora je postavený na jednoduchej myšlienke, že ohýbanie slova v slovenčine sa v najväčšej miere prejaví na jeho

6 Obdobná technika bola použitá v prípade prvej verzie Tvaroslovníka.

koncovke. Vstupom algoritmu je slovo v ľubovoľnom tvare, výstupom je zoznam možných základných tvarov.

Program pracuje s dvomi slovníkmi – slovníkom základných tvarov slov a slovníkom vyskloňovaných tvarov (vo formáte slovo – lema). S cieľom získania databázy slov slovenského jazyka spracovali na Ústave informatiky UPJŠ Slovník slovenského jazyka a Veľký slovník cudzích slov do elektronickej podoby. Výsledkom je zoznam 150 000 slovenských slov a 60 000 slov cudzieho pôvodu. [7]

Lematizačný algoritmus pracuje v troch krokoch: najprv sa hľadajú zodpovedajúce predlohy – vyberie sa slovo, ktoré má so spracúvaným slovom najdlhší spoločný koniec. Pomocou základného tvaru predlohy sa podvojnou zámenou odvodí perspektívny základný tvar slova. Nakoniec sa kontroluje, či sa získaný tvar nachádza v slovníku základných tvarov (kontroluje sa tiež rodová príslušnosť slova). [7]

V prípade vynechania tretej fázy – kontroly získaného základného tvaru v slovníku – sa rozširuje funkčnosť algoritmu aj na slová mimo slovníka (napr. vlastné mená). Toto je plánované doplniť v ďalšej fáze projektu.

Tvaroslovník je implementovaný v jazyku Java. Slovníky sú dostupné v dvoch verziách – súborovej a databázovej. Súborový slovník ako serializovaná štruktúra dát z programu sa ukázal byť vhodnejší kvôli prenositeľnosti a rýchlosti pri menej frekventovaných dopytoch. Databázová verzia slovníka (implementované v relačnej databáze *MySQL*) má zase výhodu v rýchlosti pri vysokej frekvencii opakovaných prístupov k slovníku.

Zhodnotenie

Metóda Tvaroslovníka má veľkú výhodu v algoritmickom charaktere lematizácie. Toto ju robí použiteľnejšou v rámci IR v porovnaní so slovníkovým lematizátorom z JÚĽŠ SAV. Má však aj niekoľko menších nevýhod.

V prvom rade naráža Tvaroslovník na problém s produkciou nekorektných lem, ktorá súvisí s podobnosťou niektorých tvarov slov, aj keď sa slová neskloňujú rovnako (napr. ponúk, oblúk). Ak sa pre lematizované slovo zvolí zvolí ako základ podobné slovo no s rozdielnym skloňovaním, výsledkom je nesprávna lema. Tento problém je riešený tak, že nekorektné lemy sú zachytávané v tretej fáza algoritmu – kontrola výsledku v slovníku základných tvarov. Opätovné hľadanie predlohy, čiže iteračný charakter algoritmu, však robí proces časovo náročným.

V druhom rade je problémom rýchlosť a efektivita algoritmu. Pri lematizovaní slov, ktoré sú obsiahnuté v slovníku, vráti výsledok okamžite. Problém však nastáva pri slovách, ktoré sa v sloníku nevyskytujú – opakované hľadanie predlohy je časovo náročné a pri slabšom hardvéri môže hľadanie jednej lemy pokojne presiahnuť 1 sekundu, čo je pri spracovaní rozsiahlej zbierky dokumentov neprijateľný čas. Na optimalizácii algoritmu sa pracuje.

Ako ďalšie zlepšenie sa ponúka rozširovanie databázy vyskloňovaných tvarov. V prípade, že by táto databáza obsahovala všetky vyskloňované tvary slov zo Slovníka slovenského jazyka, hľadanie podobnej koncovky by sa mohlo obmedziť na hľadanie presného tvaru a tretia fáza algoritmu – overovanie v slovníku – by stratila zmysel, lebo algoritmus by vracal len slová zo slovníka. Časovo náročné hľadanie predlôh by sa využívalo len na hľadanie základného tvaru slov mimo databázy a fáza overovania v slovníku základných tvarov by sa teda mohla celkom vypustiť. Toto by umožnilo lematizáciu slov mimo slovníka.

3 Možnosti riešenia algoritmického stemmera

3.1 Lingvistické zdroje

Pri návrhu riešenia možno vychádzať z viacerých zdrojov. Dobrým základom pre tvorbu stemmera slovenčiny je Slovenský národný korpus[7] (SNK).

„*Korpus textov predstavuje špecifický súbor jazykových dát, ktorý sa buduje v elektronickej podobe. Jeho základom sú texty zvyčajne rôznych štýlov a žánrov, ku ktorým sa pridávajú lingvistické informácie na úrovni slova (textovej jednotky), vety aj celého textu. Výkonné vyhľadávacie nástroje umožňujú vyhľadávanie a triedenie skúmaných jazykových prostriedkov a informácií. Lingvisti na základe autentického jazykového materiálu opisujú významy a funkcie slov i ďalších jazykových javov, ich štatistiky, spájateľnosti a pod. Bežným používateľom jazyka môže korpus poslúžiť ako zdroj praktického poznania systému jazyka a overenia či doplnenia jednotlivých poznatkov o reálnom fungovaní jazykových prostriedkov v praxi.*“ [10]

V rámci tohto projektu je k dispozícii množstvo textov rôznych štýlov a žánrov, ktoré sú morfologicky anotované (časť „ručne“, časť nástrojom Slovenský morfologický analyzátor) – majú doplnkovú lingvistickú informáciu o gramatických kategóriách slov. Toto môže poslúžiť ako základ pre extrakciu pravidiel skloňovania v prípade návrhu algoritmického lematizátora. Rovnako sa z SNK dá získať informácia o frekvencii použitia jednotlivých slov – výskyt frekventovaných slov treba zohľadniť pri návrhu slovníka v prípade slovníkového alebo kombinovaného prístupu.

Ďalej sa dáta z SNK dajú použiť ako báza slov – všetky tvary slov sa dajú získať vygenerovaním pomocou nástroja *morphology_levenstein*.

Ďalším dobrým východiskom pre získanie lingvistických dát sú slovníky dostupné v rámci projektu *sk-spell*[8], ktoré sa využívajú v kancelárskych programoch *OpenOffice* na kontrolu pravopisu.

Najrozsiahlejšie lingvistické zdroje v elektronickej podobe vlastní JÚĽŠ SAV – disponuje elektronickými verziami viacerých svojich lingvistických príručiek. Elektronický slovník dostupný na stránke *slovnik.juls.savba.sk* využíva nasledujúce zdroje:

- Krátky slovník slovenského jazyka (4. edícia)
- Pravidlá slovenského pravopisu
- Slovník cudzích slov (akademický)
- Synonymický slovník slovenčiny
- Slovník slovenského jazyka
- Historický slovník slovenského jazyka V (R – Š)
- Názvy obcí Slovenskej republiky
- Databáza priezvisk na Slovensku

7 http://korpus.juls.savba.sk/
8 http://sk-spell.sk.cx/

Táto zbierku publikácií považujem spolu s SNK za najkompletnejšiu elektronickú databázu slov slovenského jazyka.

3.2 Snowball[9]

Vzhľadom k faktu, že Porterov algoritmus na stemming angličtiny nemal správnu, efektívnu implementáciu, čo zhoršovalo výsledky stemmingu vyhľadávacích systémov, v ktorých bol implementovaný, sa Martin Porter rozhodol, že zverejní *Open Source* implementáciu svojho algoritmu. V roku 2000 v rámci tohto projektu vytvoril jazyk na zápis stemmovacích algoritmov spolu s kompilátormi do viacerých programovacích jazykov. Výhodou tohto systému je aj export zásuvného modulu do *Open Source* vyhľadávacieho systému *Apache Lucene*.

Pre viacero európskych jazykov existujú v rámci tohto projektu voľne dostupné *stemmery*. Stemmer slovenčiny však zatiaľ v rámci tohto projektu vytvorený nebol. Otázne však je, či sa oplatí vytvárať stemmer slovenčiny v tomto jazyku, keď sa počet skloňovacích pravidiel pohybuje okolo tisíc[10]. Toto naráža na problém prehľadnosti a udržiavateľnosti algoritmu. (Z tohto hľadiska treba pochváliť prehľadnú hierarchickú štruktúru skloňovacích paradigiem generátora *morphology_levenstein*.)

3.3 Egothor[11] a Stempel[12]

Projekt Egothor predstavuje vysokovýkonný *Open Source* vyhľadávací nástroj napísaný v Jave. Technológia Egothoru je vhodná takmer na každú aplikáciu požadujúcu fulltextové vyhľadávanie. Egothor obsahuje nástroje na sťahovanie, indexovanie a ohodnotenie (*ranking*) dokumentov a nástroj na vyhľadávanie nad indexovou bázou. V rámci projektu Egothor bol implementovaný aj univerzálny algoritmický stemmer na spracovanie ľubovoľného jazyku.

Projekt *Stempel* pozostáva z vysokokvalitných stemmingových tabuliek pre poľštinu implementovaných nad univerzálnym algoritmickým stemmerom projektu Egothor. Stempel prebral stemmer Egothoru bez zmien, dopracované boli triedy *Stemmer* a *Benchmark* na jednoduchý prístup k funkciám stemmeru. Trieda *Benchmark* umožňuje testovanie stemmeru na textových dokumentoch pričom počíta štatistiku charakteristík stemmera.

Popis algoritmu

Algoritmus je detailne popísaný v publikácii Lea Galamboša [11]. Tu je krátky výťah:

„*Cieľom je separácia exekučného kódu stemmeru od dátových štruktúr. Inými slovami musí byť vyvinutý statický algoritmus konfigurovateľný dátami. Transformácia slov, ktorú stemmer vykoná musí byť zakódovaná do dátových tabuliek.*

Skrytým vstupom našej metódy je vzorová množina (tzv. slovník) slov (ako kľúčov) a ich stemov. Každý záznam môže byť ekvivalentne uložený ako kľúč a záznam transformácie kľúča na perspektívny stem. Transformačný záznam sa nazýva "plátací príkaz" (P-príkaz). Musí byť isté, že P-príkazy sú univerzálne, a že P-príkazy môžu transformovať ktorékoľvek slovo na

9 http://snowball.tartarus.org/

10 Počet pravidiel definovaných v prvej verzii Tvaroslovníka bol okolo 900, počet pravidiel v generátore *morphology_levenshtein* sa pohybuje okolo 1300 [5].

11 http://www.egothor.org/

12 http://www.getopt.org/stempel/

jeho stem. Naše riešenie je založené na Levenshteinovej metrike, ktorá produkuje P-príkazy ako cestu s minimálnou cenou v orientovanom grafe.

P-príkazy si môžeme predstaviť ako algoritmus pre operátor (editor), ktorý prepisuje jeden reťazec na druhý. Operátor môže použiť tieto inštrukcie (PP-príkazy):

- *odstránenie - zmaže postupnosť znakov začínajúc na súčasnej pozícii smerníka posúvajúc smerník na ďalší znak. Dĺžka tejto postupnosti je parameter;*
- *vloženie - vloží znak ch bez posunu smerníka. Znak ch je parameter;*
- *zámena - prepíše znak na súčasnej pozícii smerníka na znak ch a posunie kurzor na ďalší znak. Znak ch je parameter;*
- *prázdna operácia (NOOP) - vynechá postupnosť znakov začínajúc na súčasnej pozícii smerníka. Dĺžka tejto postupnosti je parameter.*

P-príkazy sa aplikujú od konca slova (sprava doľava). Táto zásada umožňuje redukovať množinu P-príkazov, pretože posledná operácia NOOP, ktorá presúva kurzor na koniec reťazca bez vykonania zmien, sa nemusí ukladať.“ [11]

Dátová štruktúra využitá na uloženie slovníka (slová a ich P-príkazy) je znakový strom (trie). Sú aplikované niektoré optimalizácie s účelom zredukovať a optimalizovať inicializačný strom eliminovaním nepotrebných informácií a skrátením ciest v strome.

Nakoniec na získanie stemu vstupného slova necháme slovo prebehnúť zodpovedajúcou cestou v strome (aplikujúc P-príkazy uložené v uzloch). Výsledkom je stem slova. [6]

Existenciou tohto algoritmu sa problém implementácie stemmera zužuje na definovanie a kompiláciu transformačných tabuliek. Formát súboru s transformáciami je dostupný v dokumentácii Egothoru[13] - prvé slovo v riadku je stem, zvyšok pozostáva zo všetkých vyskloňovaných variantov tohto slova. Riadky transformačných tabuliek by mali byť usporiadané podľa názvov stemov (toto je však potrebné len kvôli prehľadnosti, projekt skompiluje aj tabuľky, ktoré nedodržiavajú presne tento formát).

Formát súboru hodnotím ako jednu z výhod tohto stemmera – usporiadanie v tabuľkách je transparentné a uľahčuje ich modifikáciu.

Využitím slovníka tento stemmer pripomína prístupy využité pri tvorbe Slovenského morfologického analyzátora a Tvaroslovníka. Líši sa však v tom, že z vypracovaných tabuliek dokáže abstrahovať pravidlá aj pre slová mimo slovníka. Preto v porovnaní s týmito prístupmi ponúka výhodu v podobe adaptabilnosti. Toto je hlavná výhoda tohto stemmera.

13 http://www.egothor.org/book/bk01ch01s06.html

4 Záver

V tejto práci sme sa venovali problematike získavania základného tvaru slov v slovenčine. V úvodnej časti sme si predstavili problém a motiváciu, prečo tento problém treba riešiť. V ďalších častiach sme sa venovali prístupom k riešeniu tohto problému, pričom ako najvhodnejší prístup k stemmingu slovenčiny sa ukázal kombinovaný prístup využívajúci aj slovník, aj algoritmus pracujúci na základe tohto slovníka.

Práca poskytuje aj prehľad existujúcich riešení. Zamerali sme sa na akademické riešenia, pričom sme sa snažili na poukázanie výhod a nedostatkov oboch riešení. Napriek ich existencii je tu ešte stále široký priestor na riešenie stemmingu slovenčiny s využitím iných prístupov.

V ďalšom riešení projektu plánujem implementovať slovenský stemmer na základe algoritmu Lea Galamboša [11]. Ťažisko práce bude spočívať v extrakcii slov z lingvistických zdrojov a naplnení stemmovacích tabuliek. Ďalej je cieľom vykonať detailné testovanie na množinách slov a dokumentov a navzájom porovnať všetky dostupné riešenia. V prípade úspešnej implementácie je cieľom integrovať stemmer s vyhľadávacím systémom a testovať výkon systému a vzťah medzi počtom a relevanciou dokumentov pri použití stemmera a bez jeho využitia.

Použitá literatúra

[1] ČEREŠŇA, M.: *Výpočtový model na analýzu viet slovenského jazyka*. Bratislava: FMFI UK, 2002. Diplomová práca. Dostupné na webe: <http://www.dbai.tuwien.ac.at/staff/ceresna/ling/nl-parsing-model.pdf>

[2] GARABÍK, R. et al.: *Tokenizácia, lematizácia a morfologická anotácia Slovenského národného korpusu*. 2004. Interný materiál. Dostupné na webe: <http://korpus.juls.savba.sk/publications/block2/2004-garabik-gianitsova-horak-simkova-tokenizacia/2004-garabik-gianitsova-horak-simkova-tokenizacia.pdf>

[3] LACLAVÍK, M.: *Vyhľadávanie informácií*. 2007. Dostupné na webe: <http://ikt.ui.sav.sk/vi/vi_laclavik.pdf>

[4] LACLAVÍK, M. et al.: *Dostupné zdroje a výzvy pre spracovanie informačných zdrojov v slovenskom jazyku*. In: LACLAVÍK, M., BUDINSKÁ, I., HLUCHÝ, L.: *1st Workshop on Intelligent and Knowledge oriented Technologies*. 2006, s. 92 – 98. Dostupné na webe: <http://conference.ui.sav.sk/wikt2006/WIKT2006.pdf>

[5] GARABÍK, R.: *Slovak morphology analyzer based on Levenshtein edit operations*. In: LACLAVÍK, M., BUDINSKÁ, I., HLUCHÝ, L.: *1st Workshop on Intelligent and Knowledge oriented Technologies*. 2006, s. 2 – 5. Dostupné na webe: <http://conference.ui.sav.sk/wikt2006/WIKT2006.pdf>

[6] BIALECKI, A.: *Stempel - Algorithmic Stemmer for Polish Language*. Dostupné na webe: <http://www.getopt.org/stempel/>

[7] KRAJČI, S., NOVOTNÝ, R.: *Hľadanie základného tvaru slovenského slova na základe spoločného konca slova*. In: LACLAVÍK, M., BUDINSKÁ, I., HLUCHÝ, L.: *1st Workshop on Intelligent and Knowledge oriented Technologies*. 2006, s. 99 - 101. Dostupné na webe: <http://conference.ui.sav.sk/wikt2006/WIKT2006.pdf>

[8] FURDÍK, K.: Získavanie informácií v prirodzenom jazyku s použitím hypertextových štruktúr. Košice: FEI TU, 2003. Doktorandská dizertačná práca. Dostupné na webe: <http://web.tuke.sk/fei-cit/furdik/publik/Dizertacia_furdik_2003.pdf>

[9] GARABÍK, R.: *Levenshtein Edit Operations as a Base for a Morphology Analyzer.* In: *Computer Treatment of Slavic and East European Languages*. Zborník z medzinárodnej vedeckej konferencie Slovko 2005. Red. R. Garabík. Bratislava: Veda 2005, s. 50 – 58. Dostupné na webe: <http://korpus.juls.savba.sk/publications/block1/2006-garabik-levenshtein%20edit%20operations/2006-garabik-levenshtein%20edit%20operations.pdf>

[10] *Slovenský národný korpus : Čo je korpus?*. JÚĽŠ. [cit. 14. 05. 2008]. Dostupné na webe: <http://korpus.juls.savba.sk/about/index.sk.html>

[11] GALAMBOŠ, L.: *Semi-automatic Stemmer Evaluation*. In: KLOPOTEK, M. A. et al.: *Intelligent Information Processing and Web Mining* : *Proceedings of the International IIS: IIPWM'04 Conference held in Zakopane, Poland, May 17-20, 2004*. Advances in Soft Computing Springer 2004, ISBN 3-540-21331-7.

[12] GALAMBOS, L., SHERLOCK, W.: *Getting Started with ::egothor.* 2004. Dostupné na webe: <http://www.egothor.org/book/>