Použitie lematizácie vo fulltextovom vyhľadávaní v slovenských dokumentoch *

Stanislav Krajči[1], Róbert Novotný[2], Lucia Turlíková[3]
[1]stanislav.krajci@upjs.sk
[2]robert.novotny@upjs.sk
[3]lucia.turlikova@upjs.sk

[1,2,3]Ústav informatiky, Prírodovedecká fakulta, UPJŠ Košice

Abstrakt Navrhujeme a implementujeme algoritmus na lematizáciu a ohýbanie založený na detekcii najdlhšieho spoločného zakončenia s predlohou existujúcou v slovníku a odvodení gramatických tvarov na základe tejto predlohy. Samotné predlohy slov sa nachádzajú v základnej databáze slov, ktorú je však možné expandovať o novoodvodené slová. Ďalej popisujeme integráciu lematizujúceho nástroja s fulltextovým vyhľadávačom JDBSearch.

1 Úvod

Spracovanie prirodzeného jazyka je v súčasnosti už prepracovanou a do značnej miery preskúmanou oblasťou. Zatiaľ čo anglický jazyk sa teší značnej pozornosti (zrejme aj vďaka „jednoduchým“ aspektom tohto jazyka), v prípade slovenčiny je ešte dostatok priestoru na implementáciou základných lingvistických postupov, ako je ohýbanie slov, stemming a lematizácia. Tieto metódy sú pritom predpokladom pre korektné zabezpečenie podpory slovenčiny v prípade vyhľadávania v textoch.

V anglickom jazyku, ktorý pozná ohýbanie slov len vo veľmi prostej podobe, je možné slovotvorbu vykonať pomerne jednoducho zo spoločného slovného základu. Proces stemmingu je už takpovediac štandardizovaný v Porterovom algoritme [10], ktorý spočíva hlavne v odstraňovaní niektorej z mála prípon (napríklad „-ed“, „-ing“, či „-s“).

Slovenčina však patrí k tzv. flexívnym jazykom, kde môže mať slovo značný počet tvarov (tvoriaci paradigmu). Ukazuje sa, že odvodenie súboru pravidiel na ohýbanie slov môže byť značne zložité a v mnohých prípadoch až nemožné. V mnohých prípadoch je jednoduchšie jednorazovo nájsť všetky tvary slov a uložiť ich vo vhodnej reprezentácii.

V prvotných návrhoch o počítačovú podporu slovenčiny sa tento smer ukázal najmä z hľadiska priestorových požiadaviek ako nepriechodný [9], v súčasnosti však už vtedajšie obmedzenia nepredstavujú význačnejší problém.

* Podporené štátnym projektom výskumu a vývoja Budovanie informačnej spoločnosti, Nástroje na získavanie, organizovanie a udržiavanie znalostí v prostredí heterogénnych zdrojov, 1025/04

Prehľad existujúcich prístupov v počítačovej lingvistike je zhrnutý napr. v [2]. Najčastejšie používané metódy sú založené na manuálnom značkovaní jednotlivých lexém [3], či na vyhľadávaní tvarov s minimálnou Levenshteinovou vzdialenosťou [4].

V rámci projektu NAZOU (Nástroje na získavanie, organizovanie a udržiavanie znalostí v prostredí heterogénnych zdrojov) je pri spracúvaní nových ponúk do vektorového (resp. objektovo-atribútového) modelu, v ktorom sa evidujú rôzne štatistiky výskytu termov (slov) v dokumente, užitočné združiť rôzne tvary toho istého slova do jednej skupiny a vybrať jej vhodného reprezentanta, či už ide o spoločný slovný koreň týchto tvarov alebo ich základný tvar. Naviac sa lingvistické postupy používajú pri predspracovávaní dát (stemming a lematizácia) pri vytváraní indexu na fulltextové vyhľadávanie v textoch pracovných ponúk.

S tý cieľom prebehla na Ústave informatiky UPJŠ v Košiciach elektronizácia Slovníka slovenského jazyka [11] i Veľkého slovníka cudzích slov [12], čím sa po vyčistení dát získal zoznam 150 000 doteraz oficiálnych slovenských slov doplnených o 60 000 slov cudzieho pôvodu. Tento zoznam dát je síce neúplný (i vzhľadom na uplynutú dobu od vydania Slovníka slovenského jazyka), napriek tomu však slúži ako dostatočne obsiahla východisková databáza slov a predlôh pre nižšie popísané algoritmy.

2 Hľadanie základného tvaru

Ako sme už naznačili, pre slovenský jazyk je prakticky ťažko dosiahnuteľné nájdenie pravidiel analogických postupom z Porterovho algoritmu. Navrhli sme preto iný postup založený na triviálnom pozorovaní, že ohýbanie slova závisí primárne od jeho zakončenia.

Vstupom algoritmu je slovo v ľubovoľnom tvare (z technických dôvodov sa obmedzme len na podstatné mená, uvádzanú myšlienku však možno rovnako aplikovať i na ostatné (ohybné) slovné druhy), výstupom zoznam jeho možných základných tvarov. Ďalej predpokladáme, že máme k dispozícii databázu slovenských slov, pre ktoré je známy ich základný tvar a zoznam vyskloňovaných tvarov. (V ideálnom prípade, o ktorom hovoríme v predchádzajúcej stati, by tento zoznam obsahoval všetky tvary všetkých postatných mien.) Algoritmus má tri fázy:

1. hľadanie zodpovedajúcich *predlôh*, t. j. zodpovedajúcich tvarov slov zo zoznamu vyskloňovaných podstatných mien
2. vytvorenie možných základných tvarov pomocou „podvojnej výmeny“
3. overenie prítomnosti možných základných tvarov v zozname všetkých základných tvarov (včítane kontroly rovnosti rodu s rodom príslušnej predlohy)

Ukážeme priebeh tohto algoritmu na konkrétnom príklade. Prepokladajme, že máme nájsť základný tvar slova „ponúk“ (pričom toto slovo nie je v zozname predlôh). Ďalej predpokladajme, že v základnej databáze existuje slovo „ruka“ a všetky jeho tvary. Zdôraznime, že predlohou nemusí byť žiaden dobre známy „školský“ vzor skloňovania („chlap“, „hrdina“, „dub“, „stroj“, atď.). Všimnime

si tiež, že tu vôbec nerešpektujeme slovné základy, pri skloňovaní totiž nie sú dôležité.

- slovo: $X =$ „ponúk“
- jedna z nájdených predlôh: $Y =$ „rúk“
- spoločný (neprázdny) koniec: $K =$ „úk“
- začiatok predlohy: $Y' =$ „r“ (teda $Y = Y' + K$)
- začiatok slova: $X' =$ „pon“ (teda $X = X' + K$)
- základný tvar predlohy: $Y =$ „ruka“
- koniec základného tvaru ponuky: $K' =$ „uka“ (teda $Y = Y' + K'$)
- základný tvar slova: $X = X' + K' =$ „ponuka“
- overenie existencie slova X v zozname základných tvarov

Rovnako dobrou predlohou (za predpokladu, že by sa nachádzalo v ich zozname) by mohlo byť trebárs slovo „oblúk“, algoritmom v predposlednom kroku odvodený príslušný základný tvar „ponúk“ by však neprešiel kontrolou v poslednom kroku, takýto základný tvar totiž neexistuje.

Ak je predlôh viac, uprednostníme tú, ktorá má s daným slovom najdlhší spoločný koniec. Tento postup, samozrejme, nevylučuje, že základných tvarov daného slova môže byť viacero. Ak sa však žiadny nenájde, dané slovo je vhodne vyskloňovať a doplniť ním zoznam predlôh.

3 Implementácia – nástroj Tvaroslovník

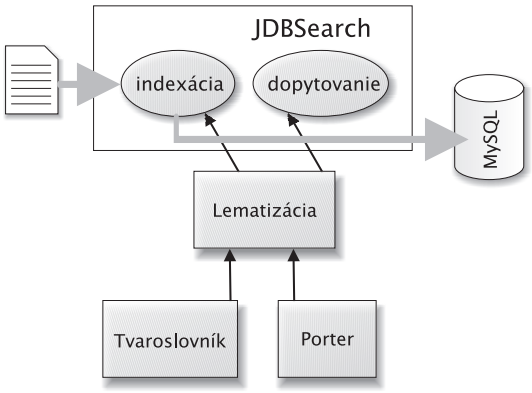
Algoritmus spomínaný v predošlej stati bol implementovaný v jazyku Java do podoby nástroja integrovateľného s ostatnými nástrojmi projektu NAZOU. Prvotné experimenty prebehli nad predlohami uloženými v databáze MySQL. Žiaľ, ukázalo sa, že z hľadiska výkonu i prenositeľnosti vstupnej databázy je lepšie zvoliť vlastnú reprezentáciu dát, a to v podobe retrográdneho zoznamu predlôh načítavaného z disku pomocou štandardných mechanizmov serializácie jazyka Java. Experimenty ukázali približne 90% úspešnosť lematizácie.

Dodajme, že Tvaroslovník je v rámci projektu NAZOU úspešne zintegrovaný s nástrojmi OnTEA [6] a JDBSearch.

4 Fulltextové vyhľadávanie s podporou slovenčiny

Fulltextové vyhľadávanie je typickým problémom v oblasti získavania dát a jednou z najčastejšie používaných metód na vyhľadávanie relevantných dokumentov pre používateľa (v prípade projektu NAZOU zodpovedá dokumentu text pracovnej ponuky). Vstupom pre vyhľadávanie je používateľov dopyt v podobe viacerých kľúčových slov (spojených logickými spojkami) a odpoveď je reprezentovaná množinou relevantných dokumentov.

V tejto oblasti je k dispozícii viacero rozličných prístupov k reprezentácii indexu výskytov kľúčových slov v dokumentoch. Jeden z najznámejších projektov Apache Lucene ho udržiava v podobe vlastnej dátovej štruktúry v súborovom



Obrázok 1. Schéma kombinácie nástrojov Tvaroslovník a JDBSearch

systéme. Alternatívny prístup poskytuje projekt NAZOU [7,8], ktorý používa na ukladanie indexu relačnú databázu MySQL.

Z hľadiska formálnej reprezentácie indexu je použitý vektorový model. Výskyty termov v dokumente sú uložené v invertovanom zozname. Ten obsahuje zoznam termov, pričom každý term v sebe nesie odkaz na všetky dokumenty, v ktorých sa vyskytuje. Samotný proces indexácie textu v dokumentoch spočíva v napĺňaní invertovaného zoznamu, ktorý je priamo namapovaný na jednotlivé databázové tabuľky.

Samotný výpočet frekvencií výskytov termov v dokumentoch používa zvyčajné vzťahy IDF

$$\mathrm{idf}_j = \log\left(\frac{n}{\mathrm{df}_j}\right),$$

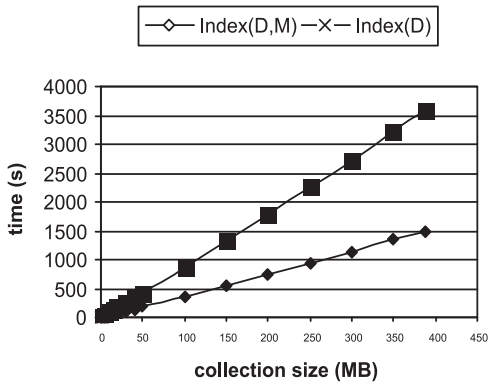
(df_j je počet dokumentov obsahujúcich term t_j). Váha j. termu v i. dokumente je potom určená ako

$$w_{ij} = \mathrm{tf}_{ij} \cdot \mathrm{idf}_j, \tag{1}$$

kde tf_{ij} predstavuje počet výskytov j. termu v i. dokumente.

Proces vyhľadávania reprezentuje spracovanie používateľovho dopytu, ktorý je reprezentovaný zoznamom kľúčových slov oddelených logickými spojkami AND (konjunkcia), resp. OR (alternatíva).

Lingvistické procesy sa v procese indexácie a dopytovania využívajú pri predspracovaní termu, a to pri čítaní vstupného textu (v prípade indexácie) a pri tokenizácii užívateľovho dopytu (v prípade dopytovania). Pôvodná implementácia nástroja JDBSearch zameraná na anglický jazyk používala na oboch miestach spomínaný Porterov algoritmus. Jednoduchou modifikáciou bolo možné použiť alternatívu v podobe nástroja Tvaroslovník (pozri obr. 1), ktorý zabezpečuje vyhľadávanie základného tvaru (v prípade indexácie i v prípade dopytovania). Alternatívnym prístupom je možnosť automatickej expanzie používateľovej otázky o všetky tvary daného slova.



Obrázok 2. Porovnanie medzi algoritmom JDBSearch a tradičným algoritmom

Nástroj JDBSearch bol podobne ako Tvaroslovník implementovaný v jazyku Java. Vykonali sme viaceré experimenty, ktoré ukázali, že samotná výmena algoritmu na vyhľadávanie základného tvaru za algoritmus s podporou slovenčiny nepredstavuje výrazné zníženie výkonnosti a indexácia naďalej vyžaduje menej času než klasický prístup uvedený v [5].

5 Záver

Ako sme už povedali, úspešnosť lematizácie je okolo 90%. Treba si uvedomiť, že vzhľadom na nejednoznačnosti prirodzeného jazyka toto číslo nikdy nemôže dosiahnuť 100%, napriek tomu však má význam pokúšať sa o jeho atakovanie. Kvalitatívne zlepšenie možno očakávať od kontroly rovnakosti slovného druhu (a v prípade podstatných mien i rodu) rozoberaného slova (ak sa nachádza v zozname) a jeho predlohy, ktorá doteraz v Tvaroslovníku nie je zapracovaná. Za kvantitatívne možno považovať zlepšenie súvisiace so spomínaným dopĺňaním predlôh. Azda dosiahnuteľným ideálom je zoznam všetkých tvarov všetkých oficiálnych spisovných slov.

Referencie

1. Apache Lucene. A high-performance, full-featured Java text search engine. [online] URL: `http://lucene.apache.org`
2. Ciglan, M., Laclavík, M.: Dostupné zdroje a výzvy pre počítačové spracovanie informačných zdrojov v slovenskom jazyku. Proceedings of 1st Workshop on Intelligent and Knowledge oriented Technologies (WIKT 2006)
3. ISpell. An interactive spell-checking program for Unix. [online] `http://ispell.hq.alert.sk/`

4. Garabík, R.: Slovak morphology analyzer based on Levenshtein edit operations, Proceedings of 1st Workshop on Intelligent and Knowledge oriented Technologies (WIKT 2006)
5. Grossman, D. A., Frieder, O., Information Retrieval: Algorithms and Heuristics, Kluwer Acamedic Publishers, 2000. ISBN 1-4020-3004-5
6. Laclavík M., Ciglan M., Šeleng M., Hluchý L.: Ontea: Empowering Automatic Semantic Annotation in Grid [Slides] to appear in proceedings of PPAM 07, Springer-Verlag
7. Lencses, R., Indexing for Information Retrieval System supported with Relational Database, Conference Sofsem 2005, Slovakia, January 2005, Proc. Vojtáš et al. (ed.): Sofsem 2005 Communications, Bratislava 2004, 81-90
8. Lencses, R., Dopytovanie v systéme zameranom na získavanie informácií s podporou relačnej databázy, Proceedings of Datakon 04, Brno, Czech Republic, Masaryk University, Brno, 2004, p. 271-280
9. Páleš, E.: Sapfo, parafrázovač slovenčiny, Veda, Bratislava, 1994. ISBN 80-224-0109-9.
10. Porter, M. F.: An algorithm for suffix-stripping, Program, 14 (3), pp. 130–137, 1980.
11. kol.: Slovník slovenského jazyka, Vydavateľstvo SAV, Bratislava, 1959–1968
12. Šaling, S., Ivanová-Šalingová, M., Maníková, Z.: Veľký slovník cudzích slov, SAMO, 2003