VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MORFOLOGICKÝ ANALYZÁTOR POMOCÍ KONEČNÝCH AUTOMATŮ

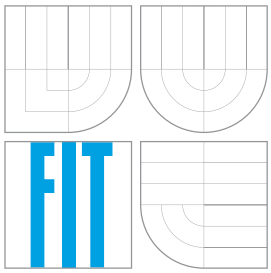
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL RYŠAVÝ

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MORFOLOGICKÝ ANALYZÁTOR POMOCÍ KONEČNÝCH AUTOMATŮ

MORPHOLOGICAL ANALYSER IMPLEMENTED AS FSAS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL RYŠAVÝ

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2013

Abstrakt

Tato práce se věnuje analýze českého jazyka a pokouší se rozšířit zatím omezenou derivativní nadstavbu, kterou disponuje morfologický analyzátor MA. Autor popisuje dosavadní stav tohoto programu a vytváří postupy pro nalezení slovotvorných vazeb, které slouží k vytváření derivačních pravidel, díky kterým je možné automatické rozšiřování znalostí české slovotvorby. Poté ilustruje, jak se data seskupují dle podobnosti, aby vytvořila derivační vzory, které usnadňují budoucí zpracování nových slov. Závěrem jsou výstupy práce zhodnoceny a jsou naznačeny směry možného rozvoje.

Abstract

This thesis deals with analysis of czech language and tries to enlarge limited derivative extension of morphologic analysator MA. Author describes actual state of this program and defines ways to find word formation connections, which serves to create derivation rules, which helps to automatically enrich knowledge of czech word formation. Illustrates how are similiar data grouped to create derivation patterns, which will make future work with new words easier. In the end, outcomes are sumarized and direction of possible future evolution is described.

Klíčová slova

Slovotvorba, derivace slov, TRIE, slovotvorné vzory, morfologická analýza, slovníková seskupení

Keywords

Word formation, derivation of words, TRIE, word formation patterns, morphologic analysis, groups of words

Citace

Michal Ryšavý: Morfologický analyzátor pomocí konečných automatů, bakalářská práce, Brno, FIT VUT v Brně, 2013

Morfologický analyzátor pomocí konečných automatů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením doc. RNDr. Pavla Smrže, Ph.D. Další informace mi poskytl Ing. Kamil Chalupníček. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Ryšavý
15. května 2013

Poděkování

Rád bych poděkoval svému vedoucímu, Pavlu Smržovi, za pomoc při získávání materiálů, za uvedení do problematiky a připravení dat, z nichž bakalářská práce vychází. Dále za cenné rady a nápady, jak řešit problémy, které se v projektu mnohokrát vyskytly. Rád bych také poděkoval Kamilu Chalupníčkovi, který mi byl vždy ochotný poradit s drobnými problémy a pomohl získat cenné informace pro vyřešení zadaných úkolů.

Obsah

Kapitola 1

Úvod a motivace

Slovotvorba patří mezi nejdůležitější prvky jazyka, neboť jen díky ní jsou lidé schopni pojmenovat nové věci, či objevené jevy a skutečnosti, které se kolem nich vyskytují. Díky slovotvorbě se může jazyk vyvíjet a reagovat tak na měnící se svět. Dovoluje nám vytvořit jména z charakteristických vlastností či odvodit nové názvy na základě vnější či vnitřních podobností a tímto rozvojem hranic jazyka umožňuje zlepšovat schopnost lidského vyjadřování.

Stávající systémy pro práci s českou morfologií velmi dobře zvládají zpracovat flektivní složku jazyka, což je vytváření nových tvarů slov ohýbáním. Mezi takové programy patří program LEMMA [12], AJKA [11], MA 3.5.2 a jiné (všechny tyto programy budou shrnuty později). Přes kvalitní práci nad flektivní morfologií, jsou programy při zpracování derivativní morfologie omezeny jen na pár nejprodukti vnějších odvození, jako například vytváření deverbativních adjektiv a substantiv, či vlastností odvozených od adjektiv a jiné.

V této práci bude uveden konkrétní příklad Výzkumné skupiny znalostních technologií na FIT VUT. V této skupině se pracuje s velmi rozsáhlým českým slovníkem, který obsahuje přes 370 tisíc základních tvarů slov (lemmat), ze kterých se odvozuje přes 7 milionů slov. Nad tímto slovníkem pracuje program MA, využívající hlavně knihovnu libma, a nástroje pro práci s konečnými automaty, které byly vytvořeny polským vědcem Janem Daciukem [2]. Principy práce programu a nástrojů, které používá pro popis a derivativních vazeb, budou v této práci objasněny.

Hlavním úkolem praktické části práce bylo vytvořit systém nad stávajícími derivativními vazbami, které jsou zachycené ve slovníku Výzkumné skupiny znalostních technologií na FIT VUT, a tím umožnit automatické rozšiřování derivativních znalostí. Dále měly být prohloubeny znalosti, aby byl systém schopen určit vazby mezi slovy *král*, *královna*, *království*. Automatické zpracování bude možné používat pro určování odvození nad novými slovy. Tím by se mohl slovník efektivně automaticky rozšiřovat i o novotvary, které se v českém jazyce teprve začínají používat (například na internetových fórech). Systém by se měl vypořádat s novotvary jako *dýňování*, které je odvozeno od slova *dýňovat* a to je odvozeno od existujícího slova *dýně*. Tím by mohl být slovník automaticky přizpůsobován vývoji českého jazyka.

Následující text popisuje jednotlivé kroky a pro snadnější pochopení je doplněn mnoha názornými příklady.

Kapitola 2

Derivace v češtině

Nejprve bude shrnut postup, jak derivují lidé svojí přirozenou znalostí českého jazyka. Práce se nesnaží opisovat knihy na tuto problematikou zaměřené, jako například knihu Novočeské tvoření slov od Vladimíra Šmilauera [14] nebo Tvoření slov v současné češtině od Zdenky Rusínové [10], či Mluvnice česká od Bohuslava Havránka [5] a mnohé jiné. Přesto bude užitečné vypsat nejdůležitější pojmy, přímo související s touto prací, aby si nikdo nepotřeboval k přečtení této práce obstarávat i spousty jiných knih.

Po objasnění pojmů, které vycházejí ze zmíněných knih, bude ukázáno, jak jsou derivace přirozeně prováděny člověkem. Tento postup, bude srovnán s postupy počítačů a bude poukázáno na problémy vznikající při automatickém zpracovávání jazyka.

2.1 Základní pojmy

Kořen (root): jedná se o základní část slova, která určuje věcný význam a je společnou částí slov navzájem příbuzných (pře-**voz**-ník). Kořen je slovotvorným základem a při práci se slovem (časování, skloňování, atd.) se v českém jazyce mění jen velmi výjimečně.

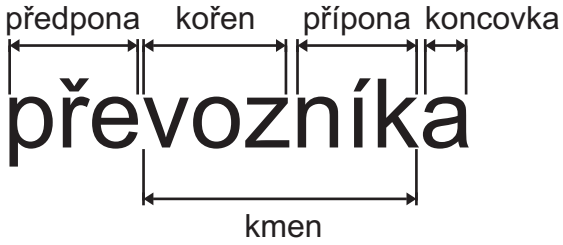
Předpona (prefix): jedná se o část, která stojí na začátku slova před kořenem. Předpony se využívají k obměnám významu kořene (**pře**-voz-ník).

Přípona (suffix): je tou částí slova, která se vyskytuje za kořenem. V případě ohebných slov se za příponou objevuje ještě koncovka (pře-voz-**ník**, pře-voz-**ník**-a).

Koncovka (ending): jedná se o poslední část slova, která je proměnlivá. Při ohýbání slov (skloňování a časování) nese informaci o pádu, číslu, rodu, atd., a proto se při tomto procesu mění (pře-voz-ník-a – 2. pád č.j.). Koncovka může být i prázdná neboli mít nulovou délku a to zejména v 1.pádě č.j. (pře-voz-ník-**-**). Pro odlišení přípony a koncovky se slovo ohýbá a tím se části odliší (převoz-**ník-**, převoz-**ník-a**, převoz-**ník-ovi**).

Kmen: zahrnuje kořen slova a jeho příponu (pře-**vozník**-a).
Vše je přehledně znázorněno na obrázku 2.1.

Základní tvar (lemma): jedná se o tvar, pod kterým je slovo uváděno ve slovnících. Slovesa jsou uváděna v infinitivech. Podstatná jména a další ohebné druhy v prvním pádě čísla jednotného.



Obrázek 2.1: Rozdělení slova dle základních pojmů

2.2 Slovotvorba

2.2.1 Zařazení

Uvedení základních elementů bylo nezbytné, protože s nimi velmi výrazně pracuje slovotvorba (nauka o tvoření slov). Z knihy Zdenky Rusínové [10] je zřejmé, že už samotné zařazení této jazykovědné disciplíny není jednoduché, protože se někdy přiřazuje k lexikologii a jindy k morfologii. Obě zmíněné disciplíny však spadají pod **lingvistiku**, nauku o jazyce.

Lexikologie: nauka o slovním základě. Ve slovotvorbě se vyskytují jisté jazykové šablony, které odpovídají slovním základům. Jednotlivě odvozená slova jsou tedy chápána jako další rozšíření slovních základů.

Morfologie: tvarosloví. Vychází z praxe a tradičního chápání morfologie. Nově vytvořená slova jsou chápána jako tvar původního slova. Onou praxí je zde myšlen fakt, že morfologie i slovotvorba mají slovotvorné prostředky, které se nemísí mezi sebou, aby tím nerušili hranice slovních druhů. Takže je zde popis jak utvářet substantiva, adjektiva či verbativa a tyto prostředky nesmí vést k vytvoření jiného slovního druhu např. adverbií.

Složité zařazení slovotvorby vyústilo do chápání této vědy jako samostatné.

2.2.2 Tvoření slov

Způsobů vzniku nových slov je v českém jazyce několik, jak o tom pojednává kniha Vladimíra Šmilauera [14]. Jednak přejímáním z cizího jazyka, což je nejčastější u technických termínů a prolínání kultur, které se posledních dvou století stávají mnohem četnější. Takto byla čeština obohacena o slova jako termostat, metr, či cool. U přejímání slov z cizího jazyka, může docházet k alternacím, některých písmen a to hlavně neznělého „s“ na znělé „z“ jako jsou například slova gymnasium a gymnázium, či komunismus a komunizmus, atd. Přejímání slov z cizích jazyků nejsou předmětem práce, a proto již nebudou podrobněji vysvětleny.

Dále slova vznikají odvozováním z již existujících slov, což je v češtině nejčastější způsob vytváření nových slov a nakonec skládáním slov dohromady (tento prvek je velmi významný například v německém jazyce).

Odvozování

Odvozování se provádí pomocí změn předpon a přípon. Tento jev je prováděn na základě několika faktorů:

Myšlenkový základ Slovo zachycuje nějaký znak, kterým nám připomene celek. Slova se odvozují na základě vnější vlastnosti (červenohlávek, plameňák, nosatec), či jiné výrazné vlastnosti: zvuky (kukačka), povaha (měkkýš, blboun), původ (kůrovec), způsob obživy (mrchožrout, květopas) a mnohé jiné.

Slovní základ Jedná se o odvození nového slova, pomocí změny slovního základu. Zachovává se význam slova, pokud se pracuje s kmeny (zahřmít, hřmění, hřmící, hřmot), při práci s kořeny se význam mění (hoř-et,hoř-ký; hor-ký,hor-livý).

Přípony Připojením za kořen vzniká nový kmen a tím nové slovo.
Slovotvorné přípony: připojením ke kořeni vzniká nové slovo (sad-ař).
Kmenotvorné přípony: připojením ke kořeni vzniká kmen, který musí být ještě doplněn (boj-ova-).
Zpětné odvozování (retrográdní): *nastává, když se k danému slovu přitváří zdánlivě základní slovo: starší německé wätsac „cestovní brašna“ bylo do češtiny přejato jako váček; k tomu bylo podle poměru ráček:rak mráček:mrak přitvořeno zdánlivě základní vak.* (Novočeské tvoření slov [14], strana 12)

Předpony Tvoření pomocí předpon se liší od přípon, protože předpona nepřistupuje k slovnímu kmeni. Nemění tudíž významovou skupinu, nýbrž ji pouze obměňuje v jejích mezích (zesílení, změna vidu, popření, atd.)

Skládání

Jedná se o tvoření slov spojováním dvou významových slov v jedno slovo, které se nazývá složenina. Jednotlivá stavební slova se nazývají *členy složeniny* a místo styku členů složeniny se nazývá *šev*.

Význam je zpravidla specifičtější, užší a odvozuje se na základě členů složeniny (černozem není libovolná černá půda).

Rozlišujeme dva základní typy složenin:

složeniny vlastní Jedná se o takové složeniny, které se nedají rozložit na samostatná slova. První člen mívá podobu, která se neshoduje se žádným tvarem základního slova, a v druhém členu se vyskytují taková slova, která nemají samostatnou existenci:

zprav-o – daj (mají šev „o“, „i“, atd.)

kaz-i – svět

listo – noš (nemají šev)

kože – luh

složeniny nevlastní Také se jim říká nepravé a dříve se nazývaly spřežkami. Jedná se tedy o složeniny, které se dají (bez změny podoby) rozložit na samostatná slova:

vlastizrádce → zrádce vlasti

divuplný → plný divu

budižkničemu → budiž k ničemu

Existují i případy, kdy se složenina nevlastní mění při skloňování ve složeninu vlastní:

pantáta → pan táta – skloňuje se pantáty, pantátové atd. (již se jedná o složeninu vlastní)

Derivační program, který byl pro slovotvorbu vytvořen, se složeninami zatím nepracuje a tak se jimi dále prace nebude zabývat.

Alternace

Týkají se hláskových změn, které vznikají při skládání nebo odvozování. Podrobně o těchto změnách pojednává historická mluvnice, protože většina alternací byla dříve pravidly, která se přestala používat, a zůstalo jen několik výjimek.

Kniha o Novočeském tvoření slov [14] rozlišuje tři základní typy alternací:

Kvantitativní změny Popisují krácení nebo dloužení samohlásek.

Krácení:

- král → kralovat
- rozpůliti → rozpulovat
- kříž → křižovat

Dloužení:

- hrad → hrádek
- sypat → nasýpat
- mihnout se → míhat se

Kvalitativní změny samohláskové

stupňování: nesu → nosím, teče → točí

stahování: voj vodit → vojvoda; z čehož vzniklo „vévoda“

přesmyknutí: soli → slaný

úžení: řemen → řemínek, nehet → nehýtek

přehláska: svatý → světější, vejce → vaječný

Kvalitativní změny souhláskové Provádí se měkčení:

plačky → plačtivý

opice → opičí

strnad → strnadí

výr → výří

opozdit → opožďovat

hraběte → hraběcí

vysoký → vyšší

hladit → uhlazovat

jih → jižní

čtvrtek → čtvrteční

noc → noční

živočich → živočišný

Nebo tvrdnutí:

běžet → běh

letět → letec

skříň → skřínka

pršet → sprcha

Mezi alternace patří i změny při přejímání slov cizích například

kapitalismus → kapitalizmus

filosofie → filozofie

gymnasium → gymnázium

Jak bude vysvětleno později, vypořádat se s některými alternacemi je pro počítač velmi náročné. A proto zde byly alternace uvedeny podrobněji.

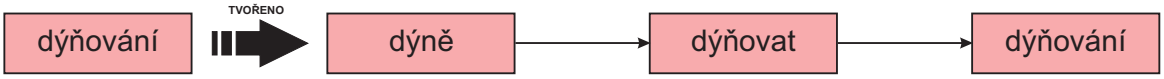
2.3 Aplikace slovotvorby

Slovotvorbu a její pravidla využíváme každý den. Pokaždé když potřebuje pojmenovat něco nového (objev nebo vynález), většinou využijeme pravidel slovotvorby (odvození názvu na základě podobnosti předmětu s jiným (kůrovec, plameňák), či na základě komponent, ze kterých se nová věc skládá (kladkostroj, termoregulátor). Slovotvorba tedy pomáhá jazyku se vyvíjet a držet krok s měnícím se světem. Slovotvorbu také využíváme pokaždé, když se setkáme se slovem, které neznáme, a díky pravidlům slovotvorby jsme schopni význam odvodit.
Rozvoj jazyka: Čeština se řadí mezi přirozené jazyky, jejich definice je obtížná, ale jistě je můžeme charakterizovat **slovníkem**, tedy množinou slov, která jsou srozumitelná a v daném jazyce dávají smysl, a dále pak **gramatikou**, čili souborem pravidel, jak v daném jazyce pracovat se slovy.[9]

Přirozené jazyky nevznikly za nějakým specifickým účelem (na rozdíl od jazyků formálních (programovací jazyky, či esperanto)) a jsou běžně používány lidmi. S vývojem lidské společnosti se jazyky přizpůsobují aktuálním potřebám (některá slova zastarávají stejně tak jako zastarávají předměty, které popisovaly (pluh, cep, řemdih, sudlice), až nakonec

zmizí úplně (láptě - nízká bota z dřevěného lýka či březové kůry). Zaniklé předměty a vynálezy jsou zpravidla nahrazeny novými, které je nutno pojmenovat a k tomu slouží právě slovotvorba. Jazyk se tak mění stejně, jako se mění doba, ve které žijeme.
Odvození neznámého slova: Odvození neznámého slova se děje pomocí zažitých pravidel slovotvorby českého jazyka a využíváme jej pokaždé, když se střetneme s neznámým slovem. U tohoto neznámého slova si určíme kmen a tím zjistíme, co k němu bylo přidáno. Kmen slova člověk určí na základě podobnosti s pro něj známými slovy. Poté nalezne slova, která byla odvozena podobě, či-li přidáním stejných přípon, předpon a koncovek. Jelikož rozeznáme, jak změna slov, která známe, upravila jejich význam, jsme schopni odhadnout, i jak se změnil význam pozměněním kmenu neznámého slova a tudíž odhadnout, co vyjadřuje.

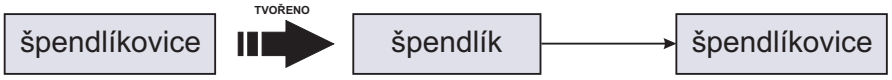
Následně bude uvedeno několik názorných případů, které jistě lépe vysvětlí postup odvozování nových slov (viz obrázek 2.2):



Obrázek 2.2: Cesta k nalezení nového slova

2.3.1 Analýza člověkem

Nejpodobnější slovo pro neznámé dýňování bude dýně. Jelikož je člověk schopen určit původní slovo, určí, k jakým změnám došlo. Člověk snadno pozná příponu -ování. Ze znalosti a podobnosti ostatních slov, je -ování odvozeno od verbativa. Respektive je ze slovesa vytvořeno podstatné jméno označující činnost, pomocí změny -at na -ání (běhat → běhání; plavat → plavání; zpívat → zpívání; atd.). Proto se musí nejdříve propojit podstatné jméno dýně se slovesem. Pomocí známých vazeb mezi slovy doplní člověk k podstatnému jménu -ovat (dýně → dýňovat). Poté se sloveso převede na podstatné jméno vyjadřující činnost (dýňovat → dýňování), čímž je analýza hotova. Význam slova dokáže člověk odvodit z kontextu věty nebo ze slovotvorných postupů, které pro vytvoření slova použil. V tomto případě by se jistě jednalo o pojmenování činnosti, která nějak pracuje s dýněmi.



Obrázek 2.3: Cesta k nalezení nového slova

V případě, který znázorňuje obrázek 2.3, již musí člověk pravidla slovotvorby využívat mnohem obezřetněji. Při určování původního slova, vyjde slovo „špendlík“. Primárním významem tohoto slova je drobný pomocný nástroj používaný hlavně při šití. Je zde, ale i sekundární význam, říkající, že špendlík je plod rostliny (jedná se o žluté plody připomínající švestky rostoucí na stromě).

Při zjišťování, jaký vliv má přidaná část však člověk správně dokáže rozlišit, že slovo vychází od plodu rostliny (švestka → švestkovice, hruška → hruškovice, jablko → jablkovice, a jiné). Bude se tedy jednat o nápoj vytvořený z daného plodu.

Možností ke kterým při poznávání neznámého slova dospějeme, může být mnoho a proto je důležitý i kontext (někdy ani ten nestačí a člověk se musí na pojem zeptat).

2.3.2 Analýza počítačem

Problém nastává při počítačové analýze slov. Kontext je pro tyto programy mnohem náročněji odvoditelný. Musejí existovat programy, které analyzují sémantiku vět velmi podrobně, protože lidské řeč je velmi komplikovaná. Jako příklad bude uvedeno spojení „velký plat“, které vyjadřuje něco pozitivního (pokud v kontextu věty nebude negováno) a druhé spojení „velký notebook“, které naopak značí, že přístroj bude velký a pravděpodobně i těžký, což je spíše chápáno jako negativní informace. Na příkladu lze vidět, že stejné slovo „velký“ může mít více významů. Proto je celková analýza a odvození kontextu nesmírně složitá.

Dále se počítač nemůže nikoho zeptat (dokud slovo není napoprvé zpracováno nějakým jiným strojem, kterého by se zeptal) na upřesnění informace. Jedinou možností je ptát se uživatele, ale pak už nelze mluvit o automatické analýze počítačem.

Určení kmene je opět velmi složité a práce s kořeny může vytvářet nesmyslná slova. Například slovo „pes“ v druhém pádě má tvar „psa“ Kdyby počítač uvažoval, že kořenem bude neměnná část slova, naprosto chybně by určil, že kořenem slova je samotné „p“. Pokud program špatně určí kořeny, začne slova, která spolu nemají nic společného, považovat za stejné. Příkladem tohoto problému jsou velmi podobná slova jako například král a králík, mezi kterými počítač špatně rozezná rozdíl.

Při aplikaci derivačních pravidel by navíc mohlo docházet, a bohužel i dochází, k propojování slov, které spolu významově nesouvisí.

Jako příklad poslouží verbativum „běhat“, pokud se z daného slova vytvoří substantivum, vznikne „běh“. Vazba je tedy běh → běhat nebo v obráceném směru běhat → běh. Když tuto stejnou vazbu program aplikuje na sloveso „rubat“ vznikne slovo „rub“, které již nemá nic společného s těžbou, ale jedná se o slovo popisující stranu (rub a líc). Stejný problém vznikne při aplikaci vazby na substantivum smrk → smrkat, ani v tomto případě významy nemají nic společného.

Dále můžeme uvést vazbu otročit → otročina, která při aplikaci na verbativum pustit, dá špatný výsledek pustit → pustina.

Popsané příklady shrnuje následující přehledný výčet, kde tučné vazby jsou správné a ostatní špatné:

běh → běhat

rub → rubat

smrk → smrkat

otročit → otročina

pustit → pustina

Posledním záludným požadavkem je, aby se při určování kořene nepropojovaly slova, která mezi sebou mají historické odvození, ale v dnešní době jsou již chápána samostatně. Proto není žádoucí, aby programy spojovaly slova jako pře-**voz**-ník a slovo **vůz**.

Všechny uvedené problémy výrazně ztěžují analýzu.

Aplikace více vazeb

Pokud hledané slovo vzniklo aplikací více slovotvorných vazeb (viz obrázek 2.2, je jejich určení pro počítač výrazně náročnější, neboť při objevení mezikroku si počítač nemůže ověřit, jestli se analýza vydala správným směrem.

Jednak hrozí, že nebude nově vzniklé slovo známo, tím nemusí být mezikrok věrohodný:

Jak bylo vysvětleno výše, člověk si propojí slovo dýňování se slovem dýně. Počítač toto propojení provede jen stěží. Bude se snažit slovo dýňování propojit s nějakým lemmatem, které přímo vzniká jedinou vazbou. Slovo dýňování dokáže navázat na slovo dýňovat. Jenže toto slovo také není lemmatem, neboť se opět jedná o nespisovný, nový, či smyšlený tvar slova dýně. Počítač musí provést další krok derivací a přitom nebude mít jistotu, jestli tvar dýňovat, ze kterého vychází, je správný nebo ne. Stejně tak by program chybně určil dýňování → dýňova, podle derivačního pravidla sebezáchova → sebezachování (odstranění ze slova ání pokud se v něm nachází a nahrazení jej za a. Tomuto pravidlu vyhovuje i zadané dýňování). Program by tedy určil obě varianty a obě by nedávaly smysl, pak by musel provádět další derivační krok a ze slova dýňovat by se dostal ke slovu dýně.

Jak je ale naznačeno, někdy je slovo určeno pomocí více než jedné derivace, což značně zvyšuje náročnost správného určení. Je otázkou kolik derivačních kroků by měl program udělat. Měl by se snažit, dokud nedojde k nějakému známému lemmatu? Měl by použít maximálně dvě, nebo tři iterace? Při nízkém počtu pokusů, se program k cíli nemusí dostat, i když šel správnou cestou. Ale s každým dalším pokusem, vzrůstá nebezpečí, že se program dostane k naprosto jiným slovním kořenům (viz příklad pes → psa). Program pak může i špatnými derivačními kroky náhodou narazit na správné slovo.

Nyní bude uveden příklad, který nepovede ke zdárnému konci. Problematika bude stejná jako u slova dýňování → dýňovat → dýně. Program vygeneruje slovo, které bude mezikrokem k nalezení výsledného a toto slovo nebude známo. Pokusí se tedy pracovat s neznámým slovem (stejně jako se pracovalo s neznámým slovem dýňovat a nalezne slovo známé, jenže tentokrát půjde o špatné propojení. Vznikne tak vazba mezi slovy smolit → mol.

Jedná se tedy o chybné spojení, které aplikuje vazbu dle příkladů chvalořeč → chvalořečit, nebo zlověst → zlověstit. Při obrácení pravidla program ze slova smolit dostane na slovo smol. Toto slovo neexistuje a program se tedy pokusí aplikovat další pravidla. Jelikož mají obě slova společnou část mol a liší se pouze v písmenu —myuvs, které patří mezi české předpony, program se ji pokusí odstranit, aby se dostal na kmen slova (při určování slova vymalovat program také odstraní předponu vy- a dostane se ke správnému slovu malovat). Po odstranění „s“ zbude slovo „mol“, které program zná a tak program derivaci určí jako správnou, což je nesprávně.

Druhým typem víceúrovňové derivace slov při slovotvorbě, jsou vzniklé mezikroky, které program bude znát. To ovšem neznamená, že se program vydal v derivacích správným směrem. Může být uvedeno propojení slov slečna → slečinkovský, které opět není jednoduché, neboť se zde nejedná o přímou derivaci a program musí nejprve ze slova slečna vytvořit slovo slečinka a z tohoto slova se teprve dostane na tvar slečinkovský (slečna → slečinka → slečinkovský). Program by ale mohl nalézt i derivaci slečna → slečnin, která je sice správná, ale ke slovu slečinkovský by se ze slova slečnin již nedostal.

2.4 Problémy českého jazyka

Čeština, stejně jako všechny ostatní přirozené jazyky, obsahuje mnoho různých případů a výjimek, které jsou počítačem velmi těžce postihnutelné. Častokrát nalezneme slova, se kterými si nejsme schopni poradit jen běžnou znalostí našeho jazyka a musíme se obrátit na odborníky, kteří se jazykem zabývají do hloubky (například pro určení vzoru či rodu daného slova. Existuje město Litovel, které je ženského rodu, a dále existuje produkt Litovel, který je rodu mužského).

Následuje výčet dalších představitelů problémů.

2.4.1 Změna významu

Jedná se o slova, která vznikají dle pravidel slovotvorby, avšak náhle jsou odvozena z jiného kořene a mají tudíž úplně jiný význam. Původní kořen, ze kterého se přes pravidla slovotvorby k takovému slovu dostaneme, danou vazbu netvoří, nebo jí tvoří jinak, nepravidelně.

Mezi tyto slova patří již zmiňované:

rub → rubat

smrk → smrkat

pustit → pustina

2.4.2 Homonyma

U těchto slov se musí rozlišovat jejich sémantický význam, který udává, jaké derivační vazby jsou pro dané slovo přípustné.

Jako příklad může být uvedeno slovo jeřáb, které vyjadřuje rostlinu, nebo stroj. Od rostliny můžeme vytvořit odvození plodu dané rostliny: jeřáb → jeřabina, zatímco od stroje by tato vazba nebyla korektní naopak jeřáb → jeřábník → jeřábnice jsou odvozeniny možné pouze od stroje. Při práci se slovy je proto nutné kontrolovat i jejich sémantiku a k tomu nám slouží vzory, které program MA využívá (bude vysvětleno později 3.5.4). Program má pro dané slovo jeřáb dva možné vzory jeden pro věc a druhý pro rostlinu.

Dalším takovým slovem může být slovo matka. V případě, že máme na mysli ženu, můžeme ke slovu vytvořit přivlastňovací přídavné jméno matčin. Zatímco pokud mluvíme o věci ze železářství, přivlastnění tvořit nemůžeme. MA má opět pro obě slova rozlišení významu podle vzorů.

2.4.3 Alternace

Jak už bylo řečeno, jedná se o komplikovanou oblast, a pro názornost uveďme příklad, jak vytvářet substantiva verbální (ze slovesa se vytvoří podstatné jméno označující činnost).

Pokud infinitiv slovesa končí na -it je nahrazen za -ení nebo -ění. Dále jsou uvedeny alternace a ani Příruční mluvnice češtiny (**??**, strana 148) někdy neudává přesnou definici, kdy se alternace aplikují.

Pokud slovo končí po odtržení -it na „ď“ bude změněno na „z“: hladit → hlazení; chladit → chlazení; nasadit → nasazení; soudit → souzení; cedit → cezení; ale (a tyto výjimky nejsou vysvětleny) ladit → ladění; řádit → řádění; dědit → dědění; dláždit → dláždění a jiné. Pokud srovnáme podobnost slov, tak každé z nich je ukončeno pomocí „-dit“, přesto se toto ukončení v některých případech mění na „-dění“, v jiných na „-zení“.

Další uvedené alternace ze stejné strany jsou, že se „s“ mění v „š“: hasit → hašení; hlásit → hlášení; křísit → kříšení; ale opět zde jsou výjimky, které se nedají automaticky rozpoznat: spasit → spasení; kosit → kosení; zhnusit → zhnusení.

Tuto problematiku pro automatickou analýzu naštěstí řeší flektivní vzory, které udávají takový tvar slovesa, ze kterého je program schopen odvodit, zdali k alternaci dochází či nikoli. Jak bude vysvětleno později, tyto vzory se plně neshodují se vzory slovotvornými, ale jsou pro aplikaci slovotvorby velmi užitečné.

U jiných typů alternací nám však vzory nepomohou (viz příklady z kapitoly 2.2.2 jako např. jih → jižní; opice → opičí; řemen → řemínek, atd.).

Následující tabulka zobrazuje požadovanou derivaci verbálního substantiva, jejíž alternace korespondují s údaji flektivních vzorů.

Derivace	Flexe
hladit → hlazení	hladit → hlazen
chladit → chlazení	chladit → chlazen
nasadit → nasazení	nasadit → nasazen
ladit → ladění	ladit → laděn
dědit → dědění	dědit → děděn
dláždit → dláždit	dláždit → dlážděn
hasit → hašení	hasit → hašen
hlásit → hlášení	hlásit → hlášen
křísit → křísení	křísit → křísen
spasit → spasení	spasit → spasen
kosit → kosení	kosit → kosen
zhnusit → zhnusení	zhnusit → zhnusen

2.4.4 Možné řešení

Jak bude vysvětleno v kapitole 3.5.6, má program MA seznam slov, která zná. Postihování výjimek je velmi složité a často derivační pravidla na obecné úrovni derivují slovo nesprávně a teprve výjimka vazbu učiní správně. Pak se musí složitým způsobem upravit pravidlo tak, aby nebylo aplikováno na výjimky.

Druhou možností je v seznamu MA postihnout slova, která nemohou existovat. Jak bude teprve ukázáno (v kapitole postihnutí výjimek 3.3), každé slovo v seznamu je ukončeno speciálním ukončujícím znakem. Když se automat dostane k tomuto znaku, prohlásí vstup za správný. Neexistující slova ze vstupu se nepodaří v seznamu vyhledat, a proto budou vstupy považovány za nekorektní.

Bylo by možné provést úpravu, která by výjimky postihovala jiným speciálním znakem, který by dané vstupy zakazoval. Pro derivace by se rub → rubat, dalo uvést do seznamu derivací, ale program by vazbu zakončil jiným znakem, který by vazbu sice správně vyhledal, ale poté zamítl. Stejně tak by se daly postihnout některé alternace.

V seznamu by se pak vyskytovaly záznamy jako „kosit“ → „kosen“ i „košen“ a u slova „košen“ by byl zakazující znak. Dané řešení je názorné na obrázku 3.4 z kapitoly TRIE 3.5.6.

Kapitola 3

Morfologické analyzátory

Poté, co byly objasněny postupy, jakými je slovotvorba aplikována v praxi a bylo vysvětleno, v čem se skrývají problémy při počítačové analýze, nadešla vhodná chvíle objasnit, jakým způsobem automatické programy pracují a jaké postupy jsou zvoleny pro jejich efektivnost. Nejprve budou představeni ostatní zástupci automatické analýzy a teprve poté se práce zaměří přímo na program morfologický analyzátor MA (dále jen MA), pro nějž se data z této práce generují.

3.1 Program LEMMA

Jedná o značkovací program, což znamená, že ke svému seznamu známých slov přidává gramatické značky a jiné informace, aby mohl se slovy pracovat. Výzkumná skupina LEMMA (laboratoř elektronických multimediálních aplikací) [12] s daným programem pracuje, aby podle něj rozhodovala, který záznam zvuku je nejpravděpodobnější. Za pomocí tohoto a jiných programu je pak schopna automaticky vytvářet titulky k nahraným zvukovým stopám.

Jelikož byl pomocí programu LEMMA označkován slovník, s kterým pak měl pracovat program AJKA, jejich výstupy se příliš neliší.

3.2 Morfologický analyzátor pro překlad

Jedná se o morfologický analyzátor vytvořený na ÚFAL MFF UK v Praze [4], který opět využívá značkování. Pracuje se souvislým textem a přidává ke slovům značky. Pomocí takto anotovaného textu a překladových slovníků pak program provádí automatický překlad. Program i jako všechny ostatní pracuje se seznamy slov a mezi nimi vyhledává vhodné značky. Jelikož program pracuje s celým textem, snaží se správně zpracovat i víceslovné výrazy.

3.3 Program AJKA

Jedná se o program, který vytvořil Radek Sedláček [11], a který je programu MA nejpodobnější, neboť je jeho předchůdcem. Program AJKA využívá značkování, takže si ke každému slovu přidává značky o gramatických informacích. Efektivita programu spočívá v rychlém vyhledání kmenových základů slov ve slovníku. Pro toto rychlé vyhledávání využívá program strukturu TRIE, kterou převádí na konečný automat (obojí bude vysvětleno

i u programu MA v kapitole TRIE 3.5.6). Protože AJKA využívá slovníkový přístup, je pro jeho správnou funkci morfologického analyzátoru důležité, aby veškerá data byla uložena ve strojovém slovníku češtiny a v definičním souboru koncových množin a vzorů.

Jelikož je program velmi podobný programu MA, budou jeho postupy vysvětleny při rozboru programu MA.

3.4 Obecně

Všechny značkovací jazyky, včetně MA, velmi dobře zvládají flektivní morfologii a jsou schopné efektivně pracovat se slovem a jeho tvary, které vznikají flexí. Na základě podrobných seznamů slov, mohou být vytvořeny morfologické vzory, které napomáhají rozšiřování slovníků. Programy však postrádají sémantické informace o slovotvorbě a změnách, které přináší.

3.5 Morfologický analyzátor MA

Nyní budou vysvětleny principy programu MA, které jsou velmi podobné i ostatním již zmíněným značkovacím jazykům.

3.5.1 Slovník MA

Slovník MA obsahuje přes 370 tisíc lemmat a více než 7 milionů odvozených tvarů. Dále jsou ke každému slovu přidány důležité informace a kombinací slov s informacemi vzniká téměř 64 milionů záznamů (například slovo jeřáb je ve slovníku uvedeno vícekrát jednou pod vzorem „návod“ a podruhé pod vzorem „slon“. Proč se nejedná o klasické známé vzory, bude vysvětleno později v kapitole 3.5.4).

Záznamy umožňují, převod zadaného slova na jeho základní tvar, nebo o slově říkají jeho mluvnické kategorie, či počítačový vzor. Tyto informace se hodí pro různou práci nad českým jazykem. Mluvnické kategorie, by se dali využívat k analýze české věty (jaké slovní druhy se v ní vyskytují a jestli je korektní jejich pořadí. Schodu podmětu s přísudkem, či jestli se přívlastek vztahuje k nějakému podstatnému jménu a tím určit jeho vhodnost ve větě atd.).

Lemma je využíváno při mnoha jiných zpracováních. Například se využívá v překladových slovnících, kdy se tvar slova nejprve převede na lemma a teprve k němu se vyhledá příslušný překlad. Další využití je právě u derivací mezi základními slovy. Jednak je derivační soubor mnohonásobně menší a navíc by derivace mezi odvozenými slovy přinášela mnoho nebezpečí, protože by hrozilo, že specifický pád, či tvar slova, by již připomínal slovo jiné a program by derivoval mezi prvky, které spolu vůbec nesouvisí. Jako například slova „pes“ a „psaní“. Kdybychom vzali druhý pád čísla jednotného od slova „pes“, vzniklo by „psa“ a toto slovo je již velmi podobné slovu „psaní“. Program by si chybně mohl myslet, že „psaní“ vychází ze slova „psa“ a toto slovo ze slova „pes“, takže by nesprávně určil, že „psaní“ je vytvořeno od slova „pes“.

Další výhodou je, že program musí pracovat pouze nad 370 tisíci lemmaty a ne nad miliony různých slov, což výrazně urychluje analýzu.

Přestože práce s lemmaty výrazně urychluje práci, je nutné zajistit, aby programy lemmata ke slovům rychle našly. Seznamy slov, nad kterými se pracuje (překlady, či analýza textu) jsou velmi dlouhé, a je nutné, aby data byla uložena ve struktuře, kde můžeme velmi rychle vyhledávat. Například pro derivaci je nutno tří vyhledání (vyhledání základního

tvaru, nalezení správné derivace, vyhledání odvozeného tvaru mluvnické kategorie pro nové slovo). Data jsou proto uložena do speciální datové struktury TRIE, která je velmi efektivní a bude popsána níže v kapitole trie 3.5.6.

3.5.2 Program MA

Autorem programu je Stanislav Černý [15] a jak popisuje ve své práci, je program MA multifunkční a nabízí mnoho užitečných zpracování. Kromě vyhledávání lemmat a mluvnických kategorií je program schopen vyhledávat derivace mezi slovy. Ke slovům, která zná, obsahuje jejich popis, ze kterého se dá odvozovat sémantická informace. Obsahuje seznam číslovek, které je schopen převádět ze slovního popisu na číselnou hodnotu a naopak. Dále je program schopen běžet v několika režimech, které usnadňují práci automatickému zpracování. Pro zadané slovo je program schopen jej vrátit ve všech mluvnických kategoriích atd.

Následně bude vysvětlen způsob, jakým si program MA pamatuje změny mezi slovy, protože se přímo využívá ke tvorbě slovotvorných vzorů. Místo, aby si program pamatoval dva dlouhé záznamy „demokrat“ a z něj vytvořené slovo „demokratka“, pamatuje si pouze změnu pomocí „kendings“. Jedná se o způsob zápisu, který opět sníží paměťové nároky slovníku a tím zefektivní práci s ním.

3.5.3 Kendings

Jak bylo naznačeno výše, jedná se o efektivní způsob pamatování si změn mezi slovy. Tento typ zápisu se hodí pro uchování změn koncovek, které jsou v češtině při flexi nejčastější. Při derivaci dochází nejčastěji ke změně přípon, a proto je i zde systém velmi efektivní.

Kendings si neuchovává informaci o tom, co se mění na co. Je zde zakódováno, jak velká část slova se musí ze slova umazat a jaká písmena se mají k zbytku slova přidat. První písmeno zápisu je velké písmeno anglické abecedy. Tím se docílilo možnosti změny až 26 písmen. Zápis tedy vypadá takto A+změna, kde počáteční písmeno abecedy A reprezentuje hodnotu 0, tedy neodstraňovat ze slova nic. B znamená odstranit jedno písmeno, C dvě, D tři atd.

běhat → běhání	změna -at na ání;	kendings: Cání
abdikace → abdikovat	změna -ace na -ovat;	kendings: Dovat
diplomat → diplomatka	přidání -ka;	kendings: Aka

Práce s touto reprezentací změn je velmi jednoduchá a zápis zabírá mnohem méně úložného prostoru. Ovšem přestává být efektivní pro zachycení předpon. Zde se usnadnění ztrácí, protože pomocí kendings, by se vždy mělo odstranit celé slovo a mělo by být nahrazeno za předponu a původně odstraněné slovo.

Proces by při zpracování zbytečně zpomaloval výpočet, protože místo aby program přečetl slovo, které by rovnou vrátil, jako změnu musel by provádět zbytečné operace.

Dalším velkým problémem by bylo automatické zpracování. Program pracuje nad daty s flektivními vzory, které by měli platit obecně pro celou svoji skupinu. Pro tvoření předpon by však obecně platit nemohli a systém pro předpony by neměl smysl.

Z těchto důvodů program MA pracuje jinak. Má seznam častých předpon, a pokud mu na zpracování přijde slovo vymalovat, vyhledá vy- a poté slovo -malovat. Jak je uvedeno v práci Stanislava Černého [15], pokud program narazí na předponu, restartuje svoji pozici

ve vyhledávacím stromu a pokračuje ve vyhledávání zbytku slova od začátku [1]. Tím redundance odpadá a program je schopen předpony řetězit. Toto rozšíření programu je velice efektivní a výrazně snižuje množství záznamů (protože téměř ke všem adjektivům můžeme přidat předpony ne- a nej-).

3.5.4 Seznam vzorů

V tradiční lingvistice se používá 14 vzorů (4 pro rod mužský životný, 2 pro rod mužský neživotný, 4 pro rod ženský a 4 pro rod střední). I v českém slovníku [17] se uvádí u každého slova koncovka v druhém pádě, aby se dalo odvodit, podle jakého vzoru se se slovem pracuje. Například dům má u sebe určeno, že v druhém pádě je domu, jako u vzoru hrad → hradu, a touto podobností i vzor určíme. Vzor je reprezentantem celé skupiny slov, se kterými se pracuje podobně. Jak jde, ale vidět na příkladu vzor hrad by při tvoření druhého pádu měl mít kendings Au (nic neměnit a přidat u). Jenže u slova dům vzniká výjimka. Kdybychom slovo převedli do druhého pádu slepým použitím pravidla (a tak by to udělal počítač), dostali bychom slovo důmu, což je rozhodně nesprávně. Jelikož je člověk zvyklý na svůj rodný jazyk a změny v něm, by byl schopen intuicí či znalostí podobných změn správně slovo pozměnit na domů. Toto počítač neudělá, systém vzorů, běžně používaný lidmi, je pro něj tedy nejednoznačný a tak bude potřebovat vzor pro každou změnu, která by mohla nastat. Pro uvedený příklad musí existovat minimálně dva vzory zachycující změny:

vzor 1: hrad → hradu – Au

vzor 2: dům → domu – Comu

Program MA využívá asi 1800 vzorů, aby postihl všechny možné změny. Když se ke slovu správně přiřadí jeho specifičtější vzor, může s ním program jednoduše pracovat, může bez kontroly provádět změny s koncovkami a jeho výsledky budou korektní.

Vytváření nových vzorů

Vzory, mohou být reprezentovány dvěma způsoby.

- seznamem, změn které se vyskytují v různých pádech
- reprezentativním slovem, na kterém jsou tyto změny viditelné

Tak jako v knize České mluvnice od Bohuslava Havránka (**??**, str. 132) je zvolen druhý systém, protože je názornější a obecně se využívá více (např. ve školách).

Systém vytvoření slov bude předveden nad podstatnými jmény, u ostatních slov je princip podobný. Slova byla seřazena dle podobnosti. Všechna slova, která byla skloňována stejným způsobem, vytvořila skupinu. Za představitele dané skupiny, bylo vybráno slovo s nejvyšší četností výskytu v testovacích datech. To proto, že reprezentativní slovo by mělo být nejčastější, aby se s ním člověk setkal na mnoha místech a snadno si jej zapamatoval (viz hrad, pán, atd.). Jakmile se slova lišila byť jen v jediném tvaru, byla dána do jiné skupiny. Rozdělování vzorů zobrazuje obrázek 3.1

[1] Je vysoce pravděpodobné, že program tak nejspíš činí jen v případě, že první vyhledání není úspěšné. Tím pádem slovo **vymalovat** nebude nalezeno napoprvé, ale až při druhém průchodu (který již kontroluje předpony). Nalezne se předpona vy- a pak se vyhledávání restartuje a program nalezne zbytek slova malovat. Slovo bude určeno za správné a korektní, protože program skončí v koncovém znaku vyhledávacího automatu, a uživateli budou vrácena data odpovídající jeho požadavku. Naproti tomu slovo **předem** bude vyhledáno již napoprvé a úspěšně vráceno. Kdyby se rozložilo na předponu pře-, tak by zbytek -dem nemohl být úspěšně vyhledán.

nominativ	akuzativ	kendings
hrad	hradu	Au
hmat	hmatu	Au
vlak	vlaku	Au
dům	domu	Comu
vůz	vozu	Cozu

Obrázek 3.1: Rozdělování vzorů

Tento systém rozdělení má velké výhody při zpracovávání. Stačí programu předat jakékoliv slovo a k němu správně určený vzor a slovo bude bez zdlouhavé a složité analýzy ihned vygenerováno ve všech tvarech.

Systém má další velký přínos do znalosti českého jazyka a tím je rozlišení životnosti ve všech rodech. Ne jen u mužského na životné a neživotné, jak je běžné, ale u všech rodů. Vznikají tak vzory životné, které mají ve všech tvarech shodné koncovky jako jejich protějšky, vzory neživotné. Rozdíly jsou viditelné až při slovotvorbě, kdy ze vzoru životného můžeme vytvořit přídavné jméno přivlastňovací, zatímco z neživotného ne. Dále nám vzory slouží k rozlišování homonym (viz příklad jeřáb → jeřabina výše).

Vzory matka a aktovka mají všechny koncovky stejné (viz obrázek 3.2).

	singulár		plurál	
nominativ	matka	aktovka	matky	aktovky
genitiv	matky	aktovky	matek	aktovek
dativ	matce	aktovce	matkám	aktovkám
akuzativ	matku	aktovku	matky	aktovky
vokativ	matko	aktovko	matky	aktovky
lokál	matce	aktovce	matkách	aktovkách
instrumentál	matkou	aktovkou	matkami	aktovkami

Obrázek 3.2: Rozdělení vzoru dle životnosti

Kdyby nebyly rozlišeny, tak při tvorbě přídavného jména přivlastňovacího by vzniklo slovo matčin, které je správně a aktovčin, které správně není. Proto je rozlišení životnosti velmi důležité.

Velké množství specifických vzorů, má i své stinné stránky. Pokud budeme mít pro každou možnou změnu jiný vzor, vznikne kolem 800 vzorů jen pro podstatná jména. Je výrazně náročnější určit, pod který vzor slovo spadá. Přidávání nových slov je automaticky

o to složitější, že nemáme informaci o tom, zdali je nové slovo životné, či nikoliv. Počítač by mohl na základě pravidel odvozovat jednotlivé tvary při skloňování, ale došel by k dvěma možným vzorům, z nichž jeden bude životný a další nikoliv. Proto je zde nutné získat další informace o slovu, ze kterého by se dala životnost odvodit, nebo se na životnost zeptat člověka.

Autor sám ve své předchozí praci pro NLP vytvořil program, který zahrnuje jednoduché grafické uživatelské rozhraní pro pohodlné přidávání nových slov do slovníku a přiřazování k nim správných vzorů.

3.5.5 Flektivní / derivatviní vzory

Jak bylo vysvětleno jsou potřeby flektivních a derivativních vzorů rozdílné. Kvůli derivativní morfologii musely být vzory matka a kotva rozděleny, jak ukazuje příklad 3.2 a tím se zvyšuje složitost systému flektivních vzorů. Na druhou stranu mnoho flektivních vzorů spadá pod jeden derivativní vzor (jako např. vzory *nový*, *metrový*, *kupovaný* a *snadný*). Z těchto důvodů bude nejvýhodnější vytvořit slovotvorné vzory, které budou v systému samostatně. Tím by mohly být sjednoceny i některé morfologické vzory, které byly rozděleny jen kvůli potřebám slovotvorby. Bude nutné doplnit funkci, která bude schopná mapovat jeden systém vzorů na druhý, ale každý systém samostatně by se stal jednodušším.

3.5.6 TRIE

Datová struktura TRIE, z anglického slova „information retrieval“ je vhodná pro ukládání v asociativních polích. Struktura TRIE, existuje v mnoha variantách, ale program MA využívá typ „vícecestné TRIE“ a proto zde bude rozebrána jen tato varianta. Zmiňovaná struktura je velmi vhodná pro uložení klíčů reprezentovanými řetězci. Těmito klíči budou právě slova ze slovníku MA.

Jelikož struktura TRIE je jednou z variant, vyhledávacího stromu, dovolte mi zopakovat některé základní prvky vyhledávacího stromu, které byly převzaty ze skript předmětu algoritmy od profesora Jana M. Hozíka ([6],78).

Kořenový strom: (*root tree*) je acyklický graf, který má jeden zvláštní uzel (*nod*), který se nazývá kořen (*root*)

Kořen je uzel, pro nějž platí, že z každého uzlu stromu vede jen jedna cesta do kořene

Z každého uzlu vede jen jedna hrana (edge) směrem ke kořeni do uzlu, kterému se říká ”otcovský”uzel a libovolný počet hran k uzlům, kterým se říká ”synovské”.

Výška prázdného stromu je 0, výška stromu s jediným uzlem (kořenem) je 1. V jiném případě je výška stromu dána počtem hran od kořene k nejvzdálenějšímu uzlu + 1

List stromu: je to takový uzel, který nemá synovské uzly.

Vícecestný strom: je takový strom, v němž rodičovské uzly mohou mít více než dva synovské uzly.

Vícecestné TRIE: je vícecestným stromem, který má klíče přidruženy ke každému ze svých listů a je rekurzivně definován takto: TRIE pro prázdnou množinu klíčů je prázdný odkaz. TRIE pro jeden klíč je list obsahující tento klíč. TRIE pro množinu klíčů s mohutností větší než jedna je vnitřní uzel s odkazy ukazujícími do dalších TRIE pro klíče se všemi možnými hodnotami číslic s uvažovaným odstraněním vedoucí číslice pro účely vytváření podstromu [7].

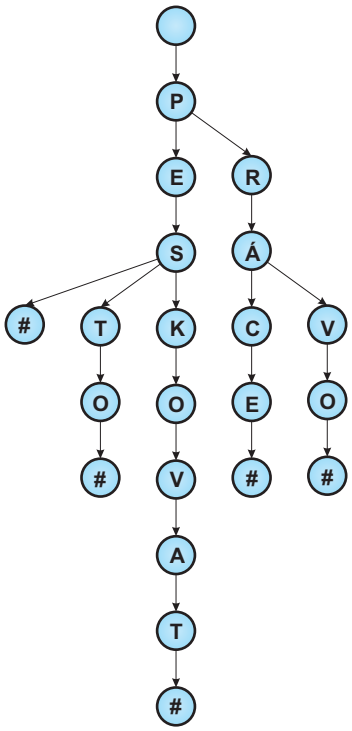
Při uvážení zmíněné definice, budou jednotlivé klíče uloženy v listech stromu. Hodnoty na začátku klíče, odpovídají posloupnosti rodičovských uzlů. Proto se struktura označuje jako prefixový strom, neboť kdybychom měli klíče „z“, „po“, „mal“, „ovat“, které by byly propojen,y jako je vyznačeno na obrázku, byla by cesta od kořene k listu, s daným klíčem celým hledaným slovem. Jednotlivé uzly, před listem (od kořene k listu) jsou tedy prefixy posledního klíče a dohromady tvoří celý hledaný klíč.

Program ale jako klíče využívá jednotlivá písmena, aby si usnadnil analýzu slov. V reálu budou tedy klíče uloženy takto (význam bude znázorněn později):

z → p → o → m → a → l → o → v → a → t

Dále bude ukázáno, jak jsou klíče efektivně uloženy pro slova „pes“, „pesto“, „peskovat“, „práce“, „právo“. Vyhledávání je v takové struktuře velmi rychlé. Pro slova „pes“, „pesto“ a „peskovat“ je společným prefixem slovo PES. Tento prefix bude v grafu uložen jako společná část klíčů a tím se slovo „pes“, se ve struktuře celé ztratí, protože je součástí jiných. Tímto způsobem struktura redukuje potřebné místo k uložení dat.

Na druhou stranu, pohlcením některých slov do prefixů by struktura porušovala svoji vlastní definici. Proto musí být zaveden speciální znak, který bude automaticky přidán za každý hledaný klíč a zároveň se nemůže vyskytovat v abecedě, ze kterých se klíče tvoří. Tím je dosaženo, že výsledný klíč (ukončující znak) bude vždy v listu stromu. Aplikaci ukončujících znaků ukazuje obrázek 3.3.

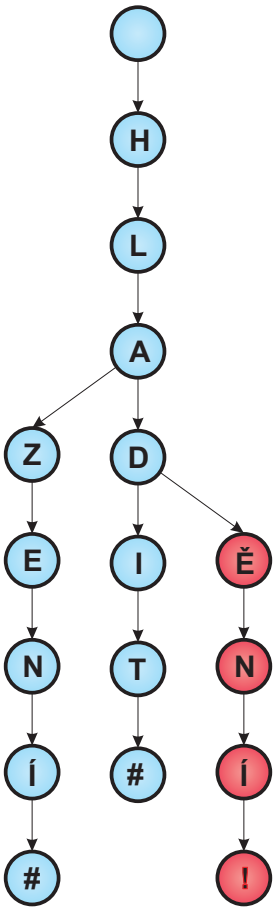


Obrázek 3.3: Uložení klíčů ve struktuře TRIE

Rozdílem při vyhledávání pomocí TRIE je oproti jiným strukturám v tom, že TRIE pro vyhledávání nepracuje s celou hodnotou klíče, ale jen s jistou vybranou částí, která závisí na vzdálenosti porovnávaného uzlu od kořene. Čím hlouběji se ve stromu bude hledat, tím vzdálenější písmeno ze slova se bude používat. Pro první úroveň ve stromu, se bude pracovat s prvním písem slova. V druhé úrovni s druhou atd.

Postihnutí výjimek

Jak bylo naznačeno v kapitole 2.4.4 je zde snadná možnost, jak vyřešit výjimky, které při aplikaci pravidel vznikají. Program by nesprávně vytvořená slova ukládal do struktury a pouze by je zakončil jiným speciálním znakem. Při nalezení takových slov, by program vstup považoval za nesprávný. Následující obrázek 3.4 ukazuje, jak by mohla být ve struktuře uložena slova „hladit“, „hlazení“ a nesprávné slovo „hladění“. Za speciální zneplatňující znak byl v tomto příkladě zvolen vykřičník „!“.



Obrázek 3.4: Návrh řešení výjimek

Souvislost s konečnými automaty

Konečným automatem je v tomto případě myšlen nedeterministický konečný automat, podle definice prof. A. Meduny. ([8], strana 14)
Nedeterministický konečný automat A je pětice A = (Q, T, δ, s, F), kde:

Q je abeceda znaků

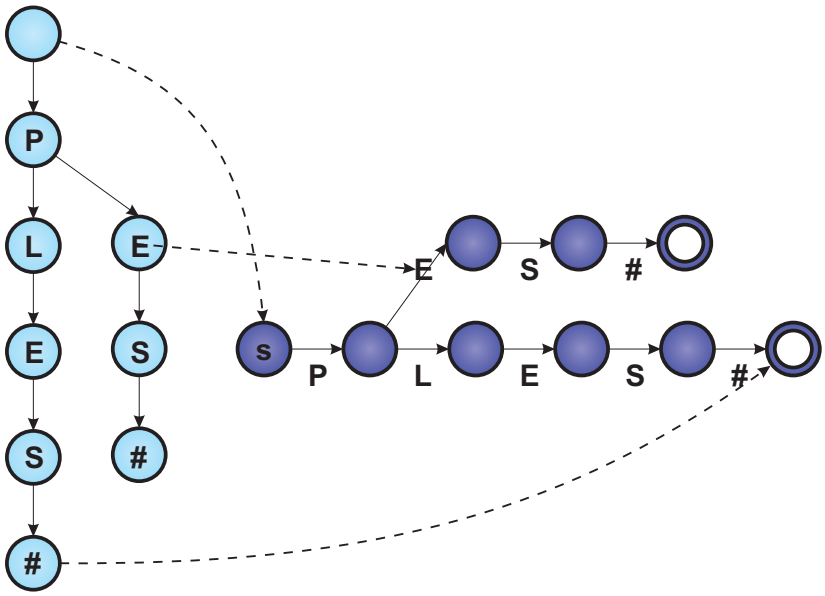
T je abeceda vstupních symbolů

δ: Q x T → 2^Q je množina přechodů

s ∈ Q a je to počáteční stav

$F \subseteq Q$ *je to množina koncových stavů*

Stromovou strukturu TRIE je snadné pomocí algoritmu převést do reprezentace pomocí konečného automatu (viz obrázek 3.5). Jako koncové stavy automatu poslouží speciální znaky přidané za každý hledaný klíč, které jsou v listech stromu. Kořen stromu, bude převeden na počáteční stav automatu. Všechny uzly budou převedeny na přechody mezi stavy.

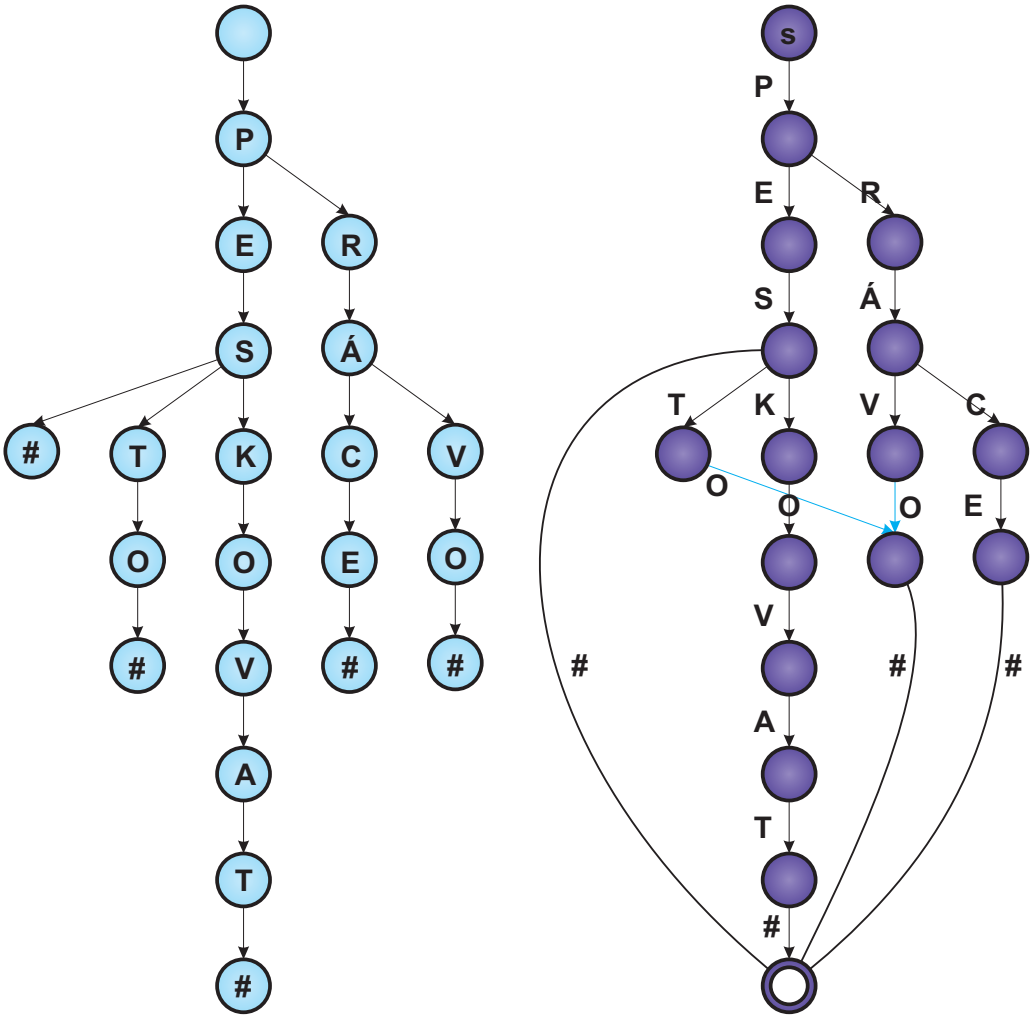


Obrázek 3.5: Převod stromu na konečný automat

Díky převedení do konečného automatu se dosáhne struktury s vyšší volností pro ukládání slov. Je poukazováno hlavně na možnost využití jiných algoritmů schopných provést minimalizaci konečného automatu, čímž se velikost výrazně sníží.

Ve stromu musel být uzel pro každý přidaný ukončující znak od každého slova. V příkladu výše (3.3) muselo být uloženo 5 ukončujících listů pro 5 slov. Jelikož program pracuje s milióny slov, tato redundance by zabírala příliš mnoho paměťového prostoru a proto je výhodné všechny stavy v minimalizaci sjednotit do jednoho.

Takto se mohou sjednotit nejen stavy konečné, ale i prefixy a společné části slov. Sjednocení společných částí ukazuje obrázek 3.6 zvýrazněnými modrými šipkami. Vpravo na obrázku je zobrazena TRIE ukládající několik klíčů a vlevo je minimální automat.



Obrázek 3.6: Minimalizace stromu pomocí konečného automatu

Kapitola 4

Derivační pravidla MA

MA provádělo derivace, podle derivačního souboru, který byl vytvořen a spravován Stanislavem Černým [16]. Tento soubor, obsahuje zhruba 220 tisíc derivačních případů. Data jsou strukturována tak, že je uvedeno, z jakého slova se vychází. K tomuto výchozímu slovu je určen vzor, podle kterého se slovo ohýbá. Dále je určena derivační vazba. Vazba určuje, o jaký typ odvození se jedná (např. že jde o vytvoření podstatného jména ze slovesa, nebo vytvoření přídavného jména z příslovce atd.). Následně je v záznamu určeno, o jaký typ směru se jedná. Písmeno „D“ reprezentuje směr odvozování (z běhat na běhání) a písmeno „B“ reprezentuje zpětné odvozování, takže pro „B“ je v souboru uvedeno běhání → běhat. Po určení směru vazby se v souboru uvádí kendings, které je nutné aplikovat, abychom získali nové slovo a vzor, podle kterého bude nové slovo skloňováno.

Zde pro názornost některé řádky ze souboru:

původní slovo + vzor pův. slova + kód vazba + směr + kendings pro nové slovo + vzor nového slova

abatyše + abatyše + D + 2.2.1.0 + Bin + matčin

abatyšin + matčin + B + 2.2.1.0 + Ce + abatyše

abatyše + abatyše + D + 2.3.0.0 + Bský + kremžský

přeslice + ulice + D + 1.7.1.1 + Cčka + aktovka

přeslička + aktovka + B + 1.7.1.1 + Dce + ulice

Pokud uživatel zadal dotaz na odvození derivací z jím zvoleného slova, program MA vyhledal lemma tohoto slova, a poté vyhledal všechny derivace, které se ze slova dají utvořit. Ač tento systém byl velmi uspokojivý, stále je v něm co zlepšovat.

Jednak se s neustálým vývojem slovníku rozšiřuje počet nepokrytých slov, a proto potřeba vytvořit nástroj, který bude automaticky přidávat derivace i mezi novými slovy.

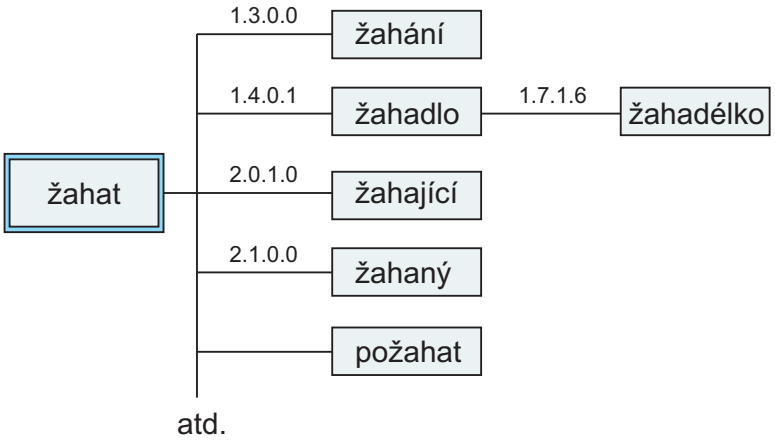
A dále není příliš vhodné používat morfologické vzory u slov pro slovotvorbu. Již bylo řečeno, že vzory se rozlišují na životné a neživotné, což je velmi výhodné, protože tím je rozlišeno například odvozování přídavného jména přivlastňovacího. Dále jsou vzory vhodné pro určování některých alternací. Proto jsou dobrou pomůckou, ale je nutné zavést slovotvorné vzory, které budou ve slovotvorbě fungovat stejně jako aktuální vzory v morfologii.

Splnění ani jednoho úkolu nebude lehké, protože jak bylo popsáno výše, čeština je plná výjimek, na které se těžko aplikují nějaká pravidla.

Kapitola 5

Seskupení slov

Program seskupí všechna slova, která jsou od sebe navzájem nějak odvozena, do propojeného grafu (viz obrázek 5.1):



Obrázek 5.1: Seskupení slov

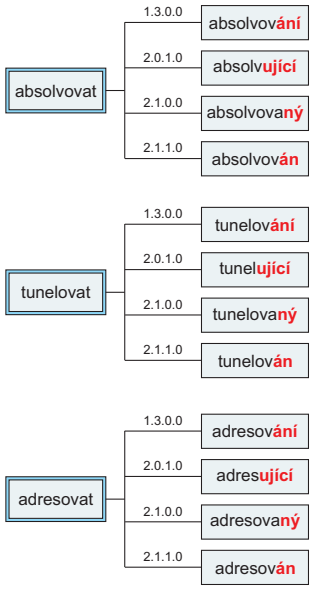
5.1 Vytvoření derivačních vzorů

Seskupení budou sloužit k vytvoření slovotvorných vzorů, zde bude popsán teoretický postup jejich vytváření. Další velmi důležitou funkcí bude hlídání obsazenosti dané vazby (aby nemohla být vytvořena více způsoby) a závěrem budou seskupení sloužit k odvozování vnitřních vazeb.

Tak jak bylo popsáno výše v kapitole „Seznam vzorů“ 3.5.4 bude i nad derivačním souborem snaha vytvořit skupiny stejně se derivujících slov. Bude se jednat o uskupení slov, ze kterých se provádějí stejné vazby stejnými změnami.

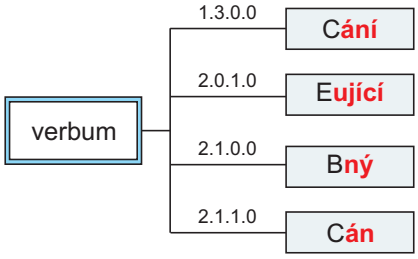
Například seskupením bude každé sloveso, protože z něj vychází odvozeniny podstatných jmen, příslovcí a přídavných jmen. Z těchto slov se pak derivují další slova a celá tato struktura bude tvořit seskupení.

Takováto seskupení se poté převedou pouze na seznamy aplikovaných změn pomocí kendings. Tyto seznamy se pak sjednotí. Seskupení se stejnými vazbami se překryjí a vznikne jen několik různých univerzálních seskupení, které budou navzájem popisovat přesně definované derivace. Po převedení předchozího příkladu (obrázek 5.2) do obecného tvaru vznikne



Obrázek 5.2: Podobnost seskupení

jediné seskupení, které již bude sloužit jako slovotvorný vzor (obrázek 5.3).



Obrázek 5.3: Zobecnění seskupení

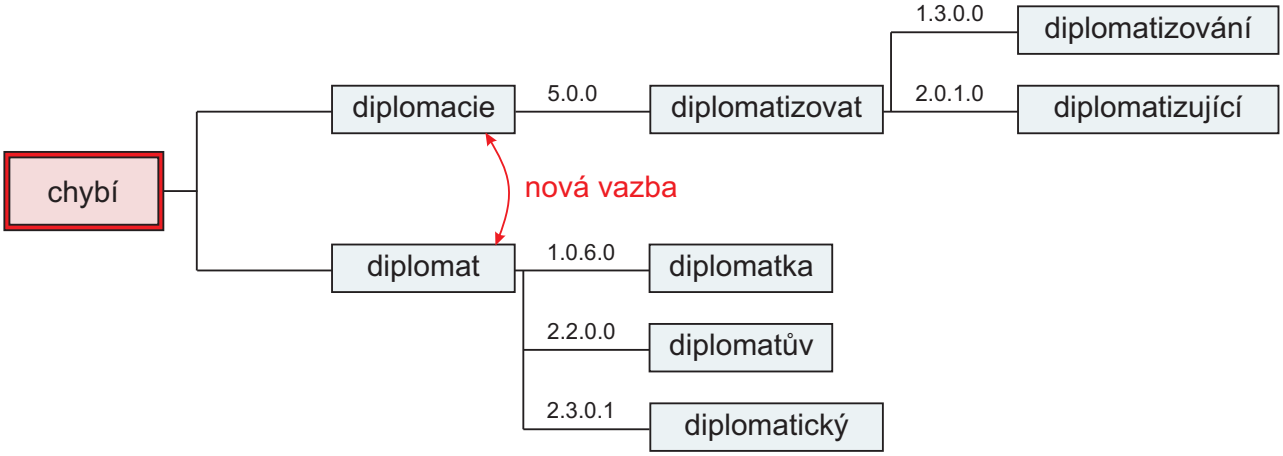
Poté bude práce s novými slovy mnohem jednodušší, protože jediným úkolem bude nalézt vhodný slovotvorný vzor, neboli derivační vzor, podle kterého se budou od slova vytvářet všechny derivace.

K tomuto hledání budou moci sloužit programy spolupracující s uživatelem, aby společně správně rozhodli, kterému derivačnímu vzoru je slovo nejvíce podobné.

Druhou variantou bude automatické rozpoznávání, neboť po určení některých typických vazeb z derivačních pravidel, bude program schopen vypozorovat příslušnost k některému ze vzorů podle projevujících se změn.

5.2 Výběr představitele

Při vytváření seskupení je pro program důležité zvolit prvotní slovo, ze kterého všechna ostatní vycházejí. U přímo propojených slov se vybere takové, které není tvořeno žádným slovem ve struktuře. Je tedy přirozeně první viz obrázek 5.1. Takto se většinou propojí slova, která vycházejí ze sloves. Existují však slova, která společné sloveso nemají (viz obrázek 5.4).



Obrázek 5.4: Sjednoceni seskupeni pomocí nové vazby

Vzniknou dvě seskupení, která by měla být sjednocena v jedno. Při tomto procesu se vybere jeden z představitelů. Program se pokusí jako představitele vybrat sloveso, ovšem v tomto případě to není možné, protože sloveso chybí, a tak vybere nejčetnější ze slov a vytvoří seskupení s hlavním slovem diplomat.

5.3 Propojování vazeb

Velmi důležitou vlastností, kterou seskupení přinese, je snížení nadgenerovávání pomocí hlídání sémantických vazeb. Z derivačního souboru vzniknou pravidla, která pro vytvoření některé vazby se specifickým kódem budou mít, vícero možností (viz vazba 2.3.0.1). Dále jsou sémantické vazby (například vytvoření zdrobněliny z podstatného jména) někdy rozděleny do několika různě číselně kódovaných vazeb. Takto se seskupí všechny dohromady a program z nich vytvoří jen jednu. Takže místo tří možných vazeb, tvořících zdrobnělinu a pěti možných úprav koncovky pro každou z vazeb, program vybere jednu a pak v hledání takovýchto slov skončí.

Bude ovšem důležité, provést bližší prozkoumání vazeb samotných aby se oddělily vazby, které mají dvě různé koncovky a obě jsou správné. Po provedení této práce se množství nadgenerovaných slov sníží.

Příklad, jak se jednou vazbou tvoří dvě správná slova, tato vazba by měla být oddělena:

narcisismus → 2.3.0.1 → narcisistický

narcisismus → 2.3.0.1 → narcistický

Posledním přínosem seskupení bude propojování vazeb mezi sebou. Jak jde vidět na obrázku 5.4 výše, jsou slova diplomat a diplomacie propojitelná. Normálně jsou tato slova propojena v seskupení přes sloveso, které však mezi slovy diplomat a diplomacie chybí. Přesto se dá na základě podobnosti těchto dvou seskupení zjistit, že patří k sobě, a mělo by být možné i rozeznat, kterou sémantickou vazbou.

Díky využívání seskupení budou moci být vytvořeny nové vazby mezi slovy a celá analýza se stane mnohem komplexnější. Programy pak budou schopny mnohem lépe zařazovat nová slova do existujících seskupení a tím se výrazně posune automatická analýza.

Kapitola 6

Implementace

Nyní, když je veškerá problematika slovotvorby vysvětlena a jsou předvedeny i původní data a záměr, jak s nimi pracovat, je vhodný čas objasnit implementační část programu.

Byl zde pokus o vytvoření velmi podrobné analýzy, která by se věnovala, každému slovu a aplikovala by na něj pravidla české slovotvorby. Tento program bohužel skončil neúspěchem, vzhledem k vysoké složitosti a časové náročnosti nutné pro převedení pravidel z češtiny do programu.

Dále byl vytvořen program schopný automatické analýzy z dat, která byla popsána výše. Budou popsány používané klasifikace a sestavování seskupení ze slov, která program zpracoval. Dále bude předvedena aplikace programu na aktuální data a možné budoucí rozšíření daného programu.

6.1 Zpracování českých pravidel

Původně bylo plánováno zpracování pravidel slovotvorby, tak jak je uvádí literatura jako Novočeské tvoření slov, či Mluvnice jazyka českého. Práce by pak byla podobná práci „K počítačové morfologické analýzy češtiny“ od Pavla Šmerka [13]. Jelikož se implementace práce vydala jiným směrem, nebude tu tato možnost zdlouhavě rozebírána. Je však vhodné ukázat, že i tato možnost zpracování českého jazyka je a zdůraznit její výhody a nevýhody.

Pro názornou ukázku zde budou ve zkratce vypsána některá pravidla slovotvorby z Mluvnice českého jazyka [3] a to přesně vytváření substantiv verbativních. Podrobný popis této vazby naleznete na straně 148-150.

...dějová jména se sufixy -(e/ě)ní/-tí teoreticky lze tvořit neomezeně od sloves všech typů. Formově tu jde o připojení koncovky -í, přiřazující k neutrům, vzor stavení, k základu shodnému s participiem pasivním, a to i u těch sloves, která protiklad aktiva/pasiva nevyjadřují...

Tvoření substantiva verbálního je vyvoditelné z infinitivního kmene.

Víceslabičná slovesa na -et/-ět tvoří substantiva verbální zcela pravidelně sufixem -ní: trpění, sázení. (Nepravidelnosti jsou zcela výjimečné: trpění X utrpení)

Víceslabičná slovesa na -at a -ovat dlouží přitom -a-: volání, mazání, kupování.

U sloves na -it se sufix -ení/-ění připojuje k základu zkrácenému o -i- a nad to tu dochází k alternaci posledního konsonantu základu:

ď : z: hlazení, chlazení, slazení, řazení, nasazení, ale ladění, řádění, dědění, bloudění...,

s : š: hašení, kvašení, hlášení pověšení, křišení, nošení..., ale: spasení, řasení, kosení, rosení..., tzn. K alternaci tu nedochází u sloves odvozených ze jmen;

...

Slovesa s jednoslabičnými infinitivy (i jejich prefigované odvozeniny) tvoří substantiva verbální takto:

Slovesa na -át sufixem -aní/-ání s nepravidelnou distribucí obou variant: lání, vání, cpaní, spaní, zdání, psaní...

Slovesa na -řít sufixem -ení: dření, mření,ření, vření

...

Jak jde vidět, popis je pro počítač neuvěřitelně složitý. Proto byl vytvořen způsob, jak popsané změny zanést do počítače pomocí přepsání pravidel do XML formátu. Bylo vytvořeno super pravidlo, které obsahovalo všechna podpravidla pro substantiva verbální.

Super pravidlo, bylo na stejné úrovni jako český obecný vzor hrad, které v druhém pádě čísla jednotného přidává -u, aby vzniklo „hradu“. Přesto pravidlo obsahuje spousty výjimek, stejných jako jsou v příkladu výše (obrázek 3.1).

Podpravidla budou postihovat jednotlivé výjimky, které se v daném derivování vyskytují. V tomto konkrétním příkladě se podpravidla dělila na dvě skupiny, podle toho, kolik slabik muselo slovo obsahovat. Dále podpravidla zahrnovala část, kterou ve slově musejí vyhledat, a pokud se to podaří, čím vyhledanou část slova nahradit. K pravidlům byly přidány i alternace, které se prováděly nezávisle na pozici písmene ve slově, což byla největší výhoda oproti kendings. Přesto se program potýkal s mnoha problémy, které vedly k jeho zavrhnutí.

6.1.1 Slabiky

Zde se začaly objevovat první problémy s určováním slabik. Jistě můžeme říci, že slabiku by měla tvořit samohláska nebo slabikotvorné „l“ či „r“. Přesto byl problém rozeznat, kolik písmen a ze které strany patří do jedné slabiky, jak vidíte na příkladu, je zde mnoho variací.

Přesto → pře-sto, přes-to

zahnat → za-hnat, zah-nat

vlk → vlk

vlček → vl-ček, vlč-ek

vlít → vl-ít, vlít

přeorganizovat → pře-or-ga-ni-zo-vat, pře-o-rga-ni-zo-vat

doopravdy → do-o-prav-dy, do-op-rav-dy, do-op-ra-vdy

Pokud by se program pokusil osamostatnit „o“ tak jako je tomu správně ve slově doopravdy, udělal by program ve slově přeorganizovat chybu. Dále je vysoce pravděpodobné, že by program ke každé samohlásce připojoval nějaké souhlásky, tím by vytvořil druhou a nesprávnou verzi do-op-ra-vdy.

Problémy nastávají i u slabikotvorného „l“ či „r“, zatímco vlk je jedna slabika, protože „l“ nám při vyhledávání slabik může posloužit jako samohláska, tak ve slově vlček by k sobě mohlo připoutat „č“ což je nesprávně. A slovo vlít, se naopak nemá dělit vůbec a písmeno „l“ má být považováno za samohlásku.

Pokud se tedy program na slabiky dívá jako na řetězce bez vnější znalosti, je velmi obtížné slova správně rozdělit a tím určit počet slabik. V příkladu mluvnických pravidel,

by konkréntné tento problém nebyl tak vážný, protože stačí rozlišit jednu slabiku, či více, ale vyskytují se i pravidla rozlišující slova specifičtěji.

Při rozdělování slabik, byla data kontrolovaná dle knihy Dělení slov: slovotvorba v praxi od Aloise Bauera a kolektivu [1]. Kniha obsahuje seznam slov, která jsou rozdělena podle slabik. Po přečtení úvodu této knihy, bude každému čtenáři jasné, jak nelehkým úkolem, rozdělení slov do slabik je.

6.1.2 Alternace

Pravidla dále podrobně popisují alternace ve slovech. Jak jde vidět na úryvku, na některých místech nejsou alternace určeny deterministicky. Nepravidelně se vyskytující, nebo vypsaná pravidlo s uvedením mnoha výjimek, způsobují, že se případy nedají správně postihnout. Při derivování sloves programu jako opora sloužili flektivní vzory, jak již bylo ukázáno v kapitole o alternacích 2.2.2. Přesto se program na dané vzory nemohl vždy spoléhat, jak je ve zmiňované kapitole také ukázáno. Program se tak v mnoha případech po složitém naučení pravidlel musel nakonec stejně uchýlit k náhodnému výběru z vygenerovaných možností.

6.1.3 Pravidla

Nejhorším problémem tohoto postupu je pomalé přidávání pravidel. Je to také důvod, proč byl tento způsob analýzy zavržen a byly hledány jiné a efektivnější způsoby pro postihnutí slovotvorby. Každé pravidlo je nutné podrobně nastudovat a otestovat. Poté se musí přepsat do jednodušší reprezentace, které bude počítač rozumět (například XML). Je ovšem nutné zajistit, aby program podporoval všechny kladené podmínky. Takže pokud některé pravidlo udávalo, že se pracuje s dvouslabičnými slovy, musela být přidána funkce na rozlišování a počítání slabik. Jiné pravidlo provádělo alternace jen před znělými hláskami a tak musel být dodán aparát na rozpoznávání znělých a neznělých hlásek. Rozšiřování programu by bylo velmi složité a musela by být přidaná funkce pro jakoukoliv již nezahrnutou podmínku. Vzhledem k počtu pravidel, by vytváření takového analyzátoru trvalo neúnosně dlouho.

Každá nově přidaná funkce by vyžadovala dlouhodobé testování a podrobnou kontrolu dat, aby se do programu nezavedla chybovost. A v případě, že by se jednalo o výjimečnou podmínku, která je aplikovatelná jen na pár set slov z českého jazyka, bylo by mnohem jednodušší a efektivnější, těchto několik set slov určit ručně, než program učit danou podmínku postihnout.

A závěrem by program stejně musel náhodně generovat výsledné slovo u pravidel, která nejsou jasně popsaná. Mnoho zdlouhavé práce, by tedy vedlo k tomu, že by program nedával jasné výsledky a bylo by třeba dalších zásahů člověka.

Tyto aspekty vedly k práci na analyzátoru jiným směrem.

6.2 Automatická analýza

Poté co byla ruční, byť o něco podrobnější, analýza češtiny zavržena, byl navržen a vytvořen program, který bude muset pravidla slovotvorby odvodit z existujících dat.

6.2.1 Vytvoření pravidel

Z derivačního souboru, byla vytvořena derivační pravidla, na základě kterých se mělo později rozgenerovávat, každé nově přidané slovo. Jelikož se na morfologické vzory nedalo plně spolehnout, program musel analyzovat slova, nad kterými byly vazby vytvořeny.

Například pro vazbu substantiva verbálního, která má kód 1.3.0.0 (ze slovesa na podstatné jméno), lze v souboru nalézt mnoho záznamů (následuje krátký příklad):

karamelov**at** + časovat2 + D + 1.3.0.0 + Cání + stavení

kalívat + dloub**at**_A + D + 1.3.0.0 + Cání + staven

jímat + dloub**at**_A + D + 1.3.0.0 + Cání + stavení

Společnou koncovkou, je pro všechny slovesa -at (slovesa jako „krmit“, které mají koncovku -it, budou seskupena do jiného podpravidla, ale pro jednoduchost se zaměříme jen na pravidelně tvořenou vazbu s koncovkou -at, a toto jiné pravidlo (s koncovkou -it), které se tvoří stejným způsobem zde nebude uvedeno).

Program pak vytvoří pravidlo, aby jakékoliv sloveso končící na -at, bylo převedeno na podstatné jméno pomocí kendings Cání.

Stejně tak, ale může být vytvořeno více pravidel. Jedno, které by hledalo -vat a nahrazovalo tuto koncovku za Dvání a druhé, které bude hledat pouze -at a bude vše nahrazovat za Cání.

Tohle se také děje, protože se program snaží o co nejvyšší přesnost z hlediska části slova, se kterou pracuje. Nové pravidlo ovšem není vytvářeno vždy, ale jen když má již nezanedbatelný podíl na počtu vazeb, které jsou takto tvořeny, tím se vyloučí vytváření pravidel z náhodných výjimek. (V takovém případě by vždy vznikla pravidla nepracující s žádným koncovým písmenem (tato pravidla někde opravdu vznikají, ale vazba 1.3.0.0 má dostatečně výrazné koncovky a proto nejsou prázdné koncovky potřeba)).

Takto se seskupí všechna pravidla, která se týkají jedné vazby (takže včetně koncovky -it měnících se v -ení). Program aplikuje pravidla a nová slova se ukládají do množiny výsledků. Z nich se vybere jen nejpravděpodobnější, aby program negeneroval desítky stejných nebo špatných slov pro jednu vazbu.

Díky tomu, že se pracuje i s koncovkou slova, je program mnohem přesnější a tvoří o 64,2% [1] méně špatných spojení, než když takto v původní části nečinil.

Problémem jsou pravidla s prázdnou koncovkou, tedy že se žádnou koncovkou slova nepracují, a pokusy o automatické nalezení dostatečně výrazné koncovky slova většinou dopadly špatně. Tyto pravidla pouze přidávají ke slovu novou příponu a nezáleží na tom, jaké má slovo zakončení (pomocí kendings se jedná o Axxx). U těchto pravidel, bohužel nejčastěji dochází ke špatnému propojení slov, jejich poměr však není tak minoritní, aby byla ručně přidávána a tak je otázkou zdali pravidla z programu vyloučit, nebo akceptovat jistou chybovost.

6.2.2 Výběr pravidla

Jak bylo řečeno, program si všechny možnosti, na které jsou aplikována pravidla, uloží do seznamu možností a pak z nich vybírá nejpravděpodobnější variantu podle různých kritérií.

Specifičnost

Jedná se o parametr, který říká, s jak velkou částí se pravidlo shoduje se slovem. Pro slovo plavat → plavání bude specifičtější pravidlo „-vat → -vání“, protože pracuje se třemi

[1] Toto číslo bylo ručně spočítáno, program generoval derivace s kontrolou koncovky slova a bez kontroly koncovky. Poté byl ručně zkontrolován vzorek 500 dat, a bylo posouzeno kolik slov je nesprávně propojeno ve kterém z případů.

písmeny, zatímco to obecnější pravidlo „-at → -ání“ jen se dvěma. Kdyby se ovšem program spoléhal jen na nejspecifičtější pravidla, nevygeneroval by příliš mnoho správných výsledků.

Tento jev je způsoben tím, že v derivačním souboru se vyskytují i výjimky, které pracují s delší částí slova než obecné pravidlo. Jedná se právě o případy, kdy se mění i kořen slova. Tyto výjimky by vždy vypadali jako specifičtější než správná pravidla a program by generoval převážně chybná data.

Vzor

Už bylo naznačeno, že vzory mohou být programu dobrým vodítkem k tomu, aby rozhodl, která z možností je správná. Při generování pravidel se stávalo, že některé vzory patřily do všech pravidel, ale také se stávalo, že některá pravidla obsahovala jen omezený výčet vzorů. Nejčastějším výsledkem byly množiny, které se z části překrývaly a z části byly unikátní.

Program zjistí, které vzory se v pravidlu vyskytovaly. Pokud se vzor nového slova, nachází v seznamu vzorů, které jsou pro vazbu typické, je pravděpodobnost nově vzniklé vazby výrazně vyšší než pravděpodobnost ostatních vazeb. Tím program mnohdy odstraní chybné možnosti.

Četnosti

Pokud program stále neurčil, která z možností je správná, výrazně mu k tomu pomohou četnosti, která říkají kolikrát se která pravidla v derivačním souboru vyskytla. Tímto se dají odstranit výjimky, které jsou velmi málo četné a jen náhodou pracují se stejně rozsáhlou koncovkou jako klasická pravidla.

Pokud má jedno pravidlo A četnost 6 tisíc a druhé pravidlo B pouze 8 výskytů, bude vždy vybráno pravidlo A. Tím se sice špatně určí případy, které spadají do výjimek, ale kdyby s výjimkami program pracoval, vygeneroval by 6 tisíc chybných dat na 8 správných slov. Výjimky tedy budou muset být do výsledného seznamu doplněny ručně nebo poloautomaticky, ale to by nemělo být příliš velkým problémem, vzhledem k jejich nízké četnosti. Navíc se oněch pár chybných výskytů dá jednoduše v seznamech MA zakázat, jak bylo vysvětleno v kapitole Postihnutí výjimek 3.5.6.

Program matematicky neporovnává, která četnost je větší. Tudíž u četností s podobnou hodnotou (450 a 480) nebude ani jedna výraznější a bude muset být rozhodnuto na bázi i jiných parametrů.

Rozsah pravidel

Jelikož program i po takto složité analýze možných výsledků stále generoval mnoho nesprávných slov, byly z programu záznamy s minimální četnosti vyřazeny. V derivačním souboru je 440 tisíc popsaných případů, vždy oběma směry, to je 220 tisíc případu derivací. Jestliže se v souboru vyskytovaly vazby, nebo jejich specifické koncovky s četností například 1-4 jednalo se o tak drobné výjimky, jejichž zpracování přinášelo jen chyby, proto byly ponechány pouze častější vazby. Schopnost programu určit správnou derivaci touto změnou nijak nevzrostla (klesla o pár desítek zanedbaných pravidel), ale počet chybně vygenerovaných slov byl téměř poloviční.

Tento fakt je velmi pozitivní, už jen pro nejhorší možný případ, že by někdo musel neznámá slova (a s vysokou pravděpodobností chybně vygenerovaná) procházet ručně, aby rozhodl, které tvary jsou správné.

Program je však možné jednoduše modifikovat, k tomu, aby pracoval s jakýkoliv výskytem vazeb, nebo jen často se vyskytujícími vazbami.

Bylo provedeno mnoho testů, které byly vzájemně porovnány. Čím nižší byla hranice k přijetí pravidla, tím více program vytvořil chybných slov. Počet správných slov, se ovšem příliš nezvyšoval. Proto bylo vyhodnoceno, že nejvhodnější je nízko četné vyzby vyřadit.

A jak bylo zmíněno výše, nepravidelně tvořená slova se automaticky v podstatě nedají postihnout, a proto budou muset být dodávána ručně.

6.2.3 Rozšíření znalostí

Získaná pravidla byla aplikována na všechna slova. Program vygeneroval asi 700 tisíc slov, o kterých pak MA rozhodovalo, která zná a která jsou pravděpodobně nesprávná.

Je pravda, že mezi známými slovy, se nacházejí i taková, která jsou chybně propojená jako pomalovat → zpomalovat, rub → rubat, pustit → pustina atd. Zároveň mezi neznámými slovy, jsou i taková, která nejsou nesprávná a měla by být přidána do slovníku.

K nově vytvořeným slovům musel být určen jejich morfologický vzor a poté byly nad slovy provedeny statistiky. Celkem bylo vygenerováno asi 320 tisíc vazeb, což je nárůst o 100 tisíc, oproti původním 220 tisícům (práce provedena jen ve směru „D“).

Pro slova, která program MA neznal, byla vytvořena funkce, která daná slova kontrolovala s „Google_cout“ seznamem, což je seznam 9 milionů různých slov, která se vyskytovala na různých stránkách, které Google prohledal. Ke slovům je uvedena jejich četnost a tak mohlo být snadno rozhodnuto, zdali se jedná o slovo s překlepem (četnost nižší než 1000), či o skutečně používaná slova, u kterých bývala četnost kolem padesáti tisíc a více.

I tato nově rozpoznaná slova (kterých bylo více než několik tisíc, byla přidána do seznamu).

Seznam všech vazeb poslouží ke komplexnější znalosti pro určování seskupení, ze kterých budou vycházet derivační vzory.

6.3 Seskupení

Jak již bylo naznačeno mnohokrát, všechna zmiňovaná práce, vedla k vytvoření seskupení, která budou sloužit k odvozování slovotvorných vzorů a propojování vazeb.

Program vygeneroval všechna slova a poté pracoval pouze se známými slovy.

6.3.1 Vytvoření seskupení

Prvotní postup měl vytvářet seskupení přesným napojením slov. Byla vytvořena struktura „Telemetn“, která měla tyto atributy: „slovo“ – z jakého slova vychází, „vazba“ – jaká vazba byla na slovo aplikována, „konc“ – s jakou změnou slova byla vazba vytvořena, „nove“ – jaké je nově vzniklé slovo. Poslední atribut se dal jednoduše dopočítat ze „slovo“ a „konc“ ale pro jednodušší porovnávání, byl ke struktuře přiřazen i její cíl. Telement měl ještě jednu důležitou položku a tou byly „potomci“, jednalo se o všechna slova, která vycházela z nově vytvořeného slova. Takto se Telementy propojovaly do ohromné sítě vytvářející seskupení.

```
Struct Telement (

     string slovo
     string vazba
```

```
    string konc
    string nove
    vektor potomci   #seznam potomku
)
```

Program pak vytvářel hashovací tabulku, kde klíči byla původní slova. Pokud se element nedokázal zařadit do některého z prvků (např. na začátku, kdy byla tabulka prázdná), byl do ní element přidán, pod klíčem „slovo“.

Dále každý načtený element, bylo nutné vhodně přiřadit. Pokud nový element vycházel ze starého, byl přidán do potomků starého elementu. V tomto případě se jednalo o případ kdy „nove“ Telementu z tabulky, bylo shodné se „slovo“ nového elementu.

```
if Telement_A.nove == Telement_B.slovo:
    Telement_A.potomci.pridej(Telement_B)
else:
    if Telement_A.potomci != None:
        for each potomek in Telement_A.potomci:
            hledej_napojeni(potomek)
```

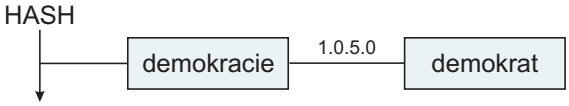
Pro demonstraci funkčnosti mějme následující pravidla (původní slovo → kód vazby → nové slovo):

1. demokracie → 1.0.5.0 → demokrat
2. demokracie → 1.3.1.4 → demokratizace
3. demokracie → 5.0.0 → demokratizovat
4. demokrat → 1.0.6.0 → demokratka
5. demokrat → 2.2.0.0 → demokratův
6. demokratka → 2.2.1.0 → demokratčin

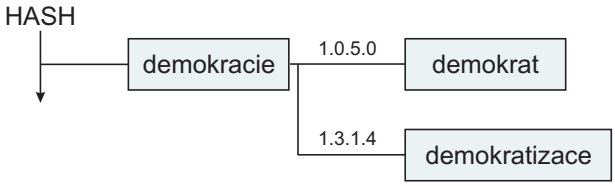
Tvorba struktury
0. před zařazením prvního pravidla (viz obrázek 6.1):

Obrázek 6.1: Prázdná hashovací tabulka

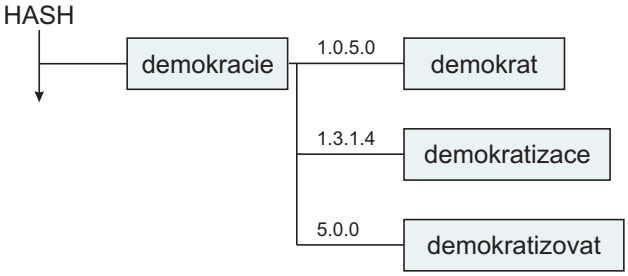
1. slovo demokracie nebude nalezeno ve struktuře a nebude nalezeno ani slovo demokrat a proto bude do struktury přidáno nové seskupení. (viz obrázek 6.2) Dále se program pokusí vyhledat seskupení začínající slovem demokrat, aby jej případně napojil (bude ukázáno v dalším příkladu).
2. Slovo, ze kterého se vychází (demokracie) bude nalezeno a bude k němu připojena nová derivace (viz obrázek 6.3):



Obrázek 6.2: Seskupení po aplikaci pravidla 1

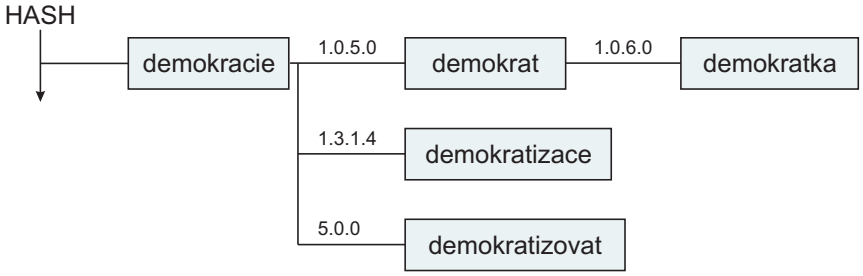


Obrázek 6.3: Seskupení po aplikaci pravidla 2



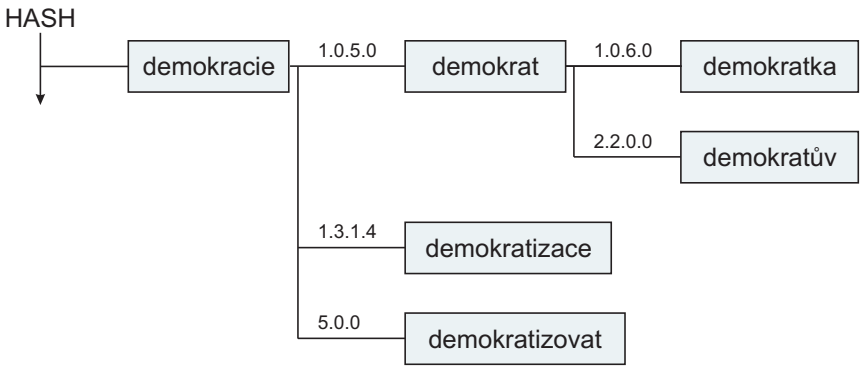
Obrázek 6.4: Seskupení po aplikaci pravidla 3

3. Přidání nového pravidla proběhne stejně jako u pravidla 2 (viz obrázek 6.4):
4. slovo ze kterého se vychází (demokrat) nebude nalezeno přímo v tabulce nejvyšších prvku (zde bude demokracie). Ale bude nalezeno v některém prvku struktury (viz pseudokód) a proto se vazba napojí sem (viz obrázek 6.5):



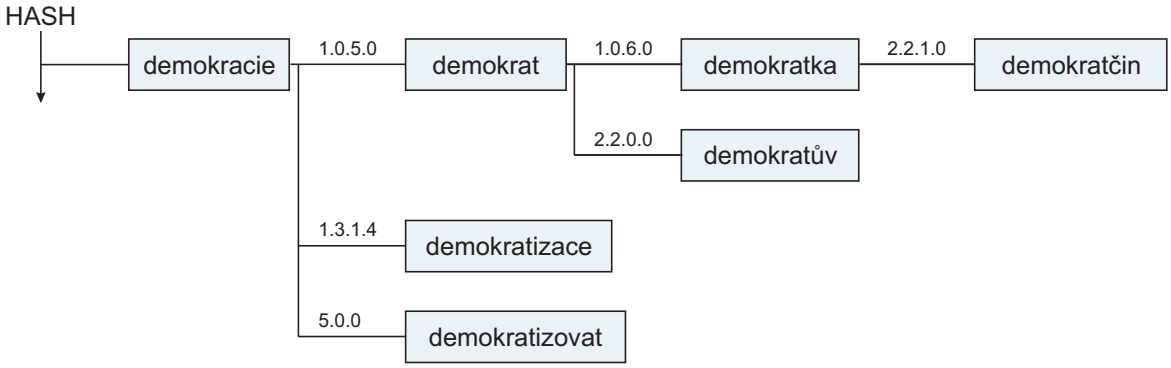
Obrázek 6.5: Seskupení po aplikaci pravidla 4

5. proběhne stejně jako 4 (viz obrázek 6.6):



Obrázek 6.6: Seskupení po aplikaci pravidla 5

6. proběhne stejně jako 4 a 5, ale slovo bude nalezeno hlouběji ve struktuře (viz obrázek 6.7):



Obrázek 6.7: Seskupení po aplikaci pravidla 6

Pro tento druh napojení, bylo nutné projít všechny prvky v tabulce a všechny jejich potomky. Takže pro každý element, se musela hledat shoda se všemi jeho potomky a potomky jejich potomků. Program tedy rekurzivně procházel celou strukturou a zanořoval se do každého potomka:

```
funkce Telement.hledej(hledany):

    if self.nove == hledany.slovo:
        return self # vratí sebe jako místo kam se má připojit
    for potomek in self.potomci:
        x = potomek.hledej(hledany)
        if x != None:
            return x
```

Takto se funkce zanořovala a vyhledávala všechny možná místa, kam se má nový element přiřadit.

Další možností bylo, že nový přiřazovaný element je předchůdce starého. Zde pak byl nový element přidán do hashovací tabulky, a starý element byl napojen na nově přidávaný a poté byl z tabulky odstraněn. Dělo se tak tehdy pokud stary_element.slovo == novy_element.nove.

```
if stary_element.slovo == novy_element.nove:

    novy_element.potomci.přidej(stary_element)
    hash.pridej(novy_element)
    hash.odstran(stary_element)
```

příklad:

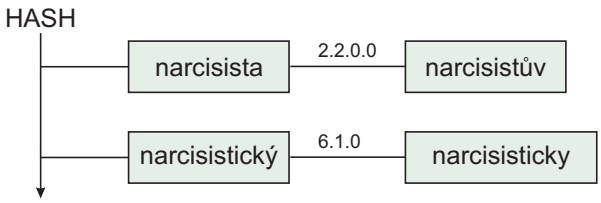
Pro vysvětlení mějme derivační vazby:

1. narcisista → 2.2.0.0 → narcisistův
2. narcisistický → 6.1.0 → narcisisticky
3. narcisismus → 1.0.5.3 → narcisista
4. narcisismus → 2.3.0.1 → narcistický
5. narcistický → 6.1.0 → narcisticky
6. narcisismus → 2.3.0.1 → narcisistický

Jak program bude zpracovávat jednotlivá pravidla:
1. Program se pokusí nalézt slovo „narcisista“ ve svém seznamu seskupení. Dané slovo nebude nalezeno ani ve slovech reprezentujících seskupení (prvotní slovo, ze kterého všechna ostatní vycházejí), a nenalezne jej ani v žádném z potomků. Poté se program pokusí vyhledat vznikající slovo narcisistův, aby mohl případně již vzniklé seskupení napojit na tuto vazbu, ovšem opět neuspěje. Proto bude založeno nové seskupení s jedinou vazbou narcisista → narcisistův a bude přidáno do seznamu (viz obrázek 6.8).

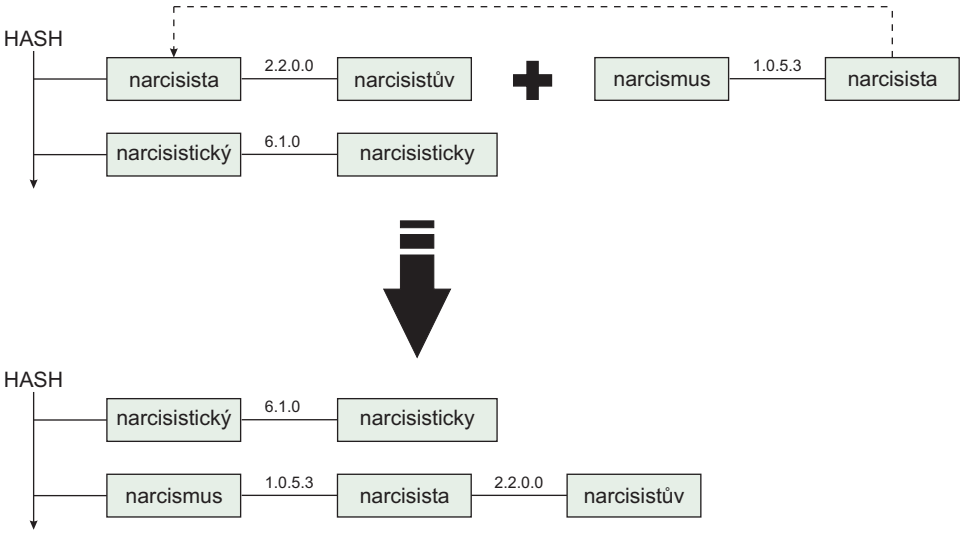
Obrázek 6.8: Seskupení po aplikaci pravidla 1

2. Po neúspěšném pokusu o vyhledání slova narcisistický i vznikajícího slova narcisisticky bude opět založeno nové seskupení a bude přidáno do struktury (viz obrázek 6.9):

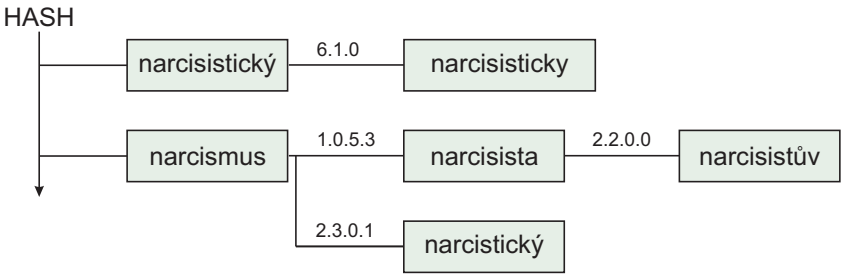


Obrázek 6.9: Seskupení po aplikaci pravidla 2

3. Program se pokusí nalézt slovo narcisismus ve struktuře seskupení a opět neuspěje, pokusí se tedy najít vznikající slovo narcisista a nyní uspěje. Slovo se nachází přímo ve struktuře, proto je program správě propojí a seskupení narcisista zanikne (viz obrázek 6.10):
4. Program se pokusí nalézt slovo narcisismus, což se mu povede, a proto do něj přidá novou vazbu narcistický. Poté zjistí, jestli neexistuje seskupení, které by vznikalo z nově vzniklého slova narcistický (viz obrázek 6.11):

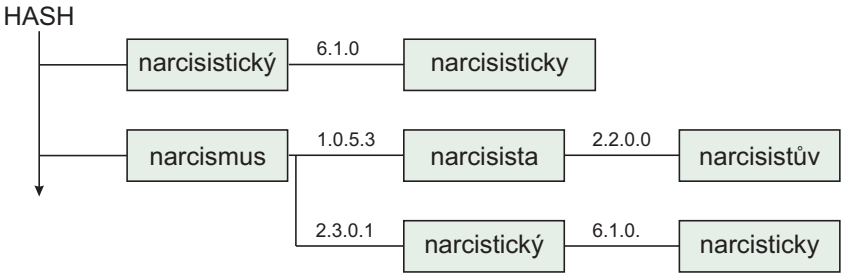


Obrázek 6.10: Seskupení po aplikaci pravidla 3



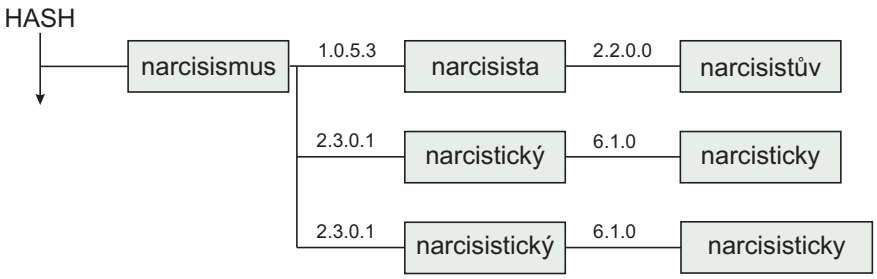
Obrázek 6.11: Seskupení po aplikaci pravidla 4

5. Program se pokusí najít v seznamu slovo narcistický, což se mu zdaří a slovo připojí. Následně se standardně hledá seskupení, které by začínalo slovem narcisticky (viz obrázek 6.12):



Obrázek 6.12: Seskupení po aplikaci pravidla 5

6. Program úspěšně vyhledá seskupení narcisismus. Přidá do něj nově vznikající vazbu narcisistický a poté se standardně pokusí vyhledat jiné seskupení ze seznamu, které by bylo odvozeno od nově vytvořeného slova narcisistický. Takové seskupení bude nalezeno a tak se připojí ke slovu narcisistický a jeho pozice v seznamu seskupení zanikne. Výsledkem bude jedna velká struktura (viz obrázek 6.13):



Obrázek 6.13: Seskupení po aplikaci pravidla 6

Program takto vytvořil seskupení, která nyní ještě nebyla obecná. Jednalo se o konkrétní derivace, z daného slova. Které se museli seskupit do slovotvorných vzorů. Příkladem může být obrázek 5.2 a jeho obecná forma obrázek 5.3

6.3.2 Propojení

Před propojením se musela seskupení zobecnit. Z elementu se vyřadilo vše neopdstatné a zbyla jen vazba a koncovka. Byl tak vytvořen obecný vzor pro každé seskupení zvlášť, podle kterého bylo seskupení vytvořeno. Už tímto sjednocením vzniklo asi osm tisíc vzorů, které se nadále budou sjednocovat na bázi jejich podobnosti.

Propojování seskupení, pracuje velmi podobně, jako jejich sestavování. Rekurzivně se prochází celá struktura a hledají se podobné znaky. Pokud bude vyhodnoceno, že jsi jsou dvě seskupení podobná, program je propojí a vznikne jedno seskupení obsahující vazby z původních dvou seskupení. Podobnost se vyhodnocuje z celkového počtu vazeb obsažených v seskupení a z počtu vazeb, které jsou stejné.

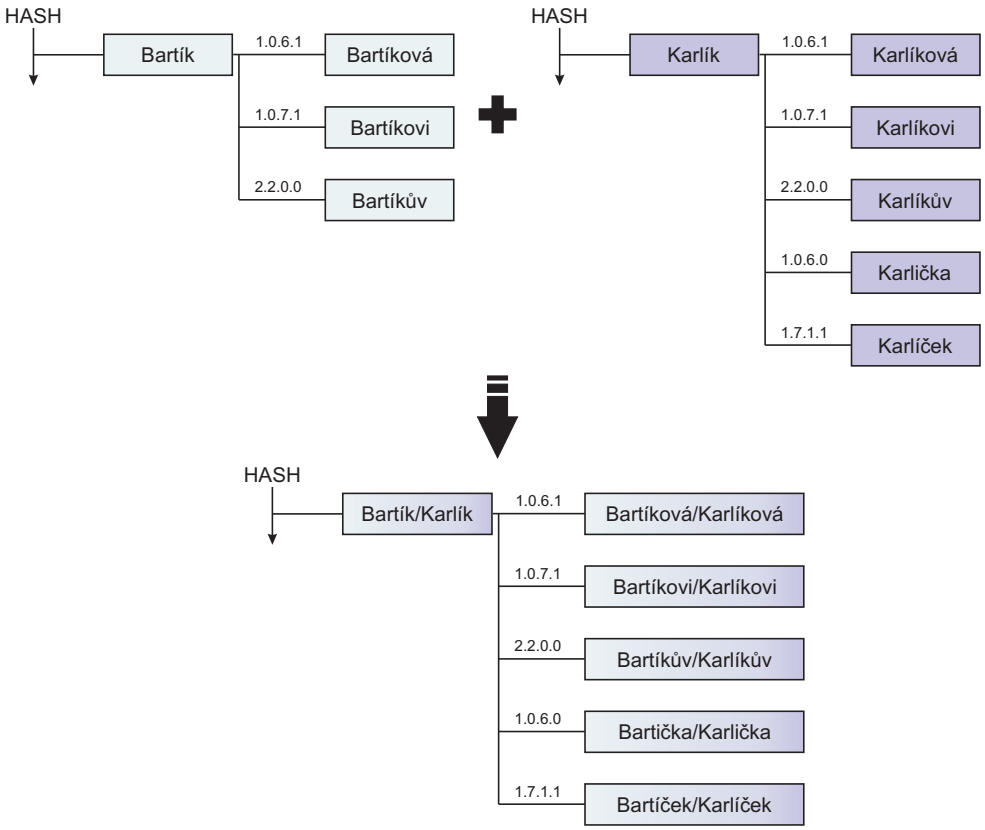
Pokud bylo vyhodnoceno, že jsou si seskupení podobná, byla navzájem sjednocena a to tak, že se vždy vyhledalo (opět rekurzivně) místo, kam se má vazba přiřadit. Přiřazením chybějících vazeb se seskupení rozšiřovala a vznikala super seskupení, jako na obrázku 6.14.

Vznikaly však problémy s uspořádáním seskupení. Například pokud ze slovesa vznikala nová slovesa pouze změnou předpony, program si pamatoval i veškerý rozvoj těchto sloves. Do struktur se zanesla jistá redundance (viz obrázek 6.15), u které se pak ukázalo, že je velkým problémem. Při sjednocování takto rozsáhlých seskupení, která byla stejná, co se derivací z původního slovesa týče, ale různá v používaných předponách vznikalo seskupení, které obsahovalo kompletní definici slovesa třikrát. Každé nové pravidlo se pak zaznamenávalo na všechna tři místa. Struktura seskupení se pak rozrůstala do takových rozměrů, že nebylo možné s ní pracovat.

Program tedy vytvářel jen pár ohromných super seskupení, které pojaly všechny derivační vazby a možnosti koncovek a tím vzniklo super seskupení s mnoha derivacemi, u kterých nebylo jasné jaké pravidlo se mělo kdy použít.

Při nastavení nulové tolerance podobnosti (seskupení musela být stejná) bylo vytvořeno 8 152 seskupení, což je příliš mnoho na to, aby byla používána jako vzory. Bude nutné program poupravit, aby měl lepší rozhodovací mechanizmy a tím poznal, která seskupení k sobě opravdu patří a která ne, poté bude spojování dostatečně kvalitní a program bude moci vytvořit derivativní vzory.

Nakonec program pracuje pomocí o nastavení maximální hladiny zanoření, a sumarizování vznikajících seskupení. Program vytvořil jisté skupiny, které dále nerozváděl a dále si pouze pamatoval, jak se tyto skupinky propojují. Jednalo se tedy o jistý způsob matema-



Obrázek 6.14: Sjednocení dvou seskupení a rozšíření znalostí

tické substituce (viz obrázek **??**).

Tím se snížili paměťové nároky i hloubka rekurzivního zanoření. Vzniklá seskupení sice nejsou natolik specifická a mají tedy teoreticky větší náchylnost k nadgenerovávání (mohly by v některých případech tvořit vazby, které se tvořit neměly). Ale i přesto bylo dosaženo dobrých výsledků.

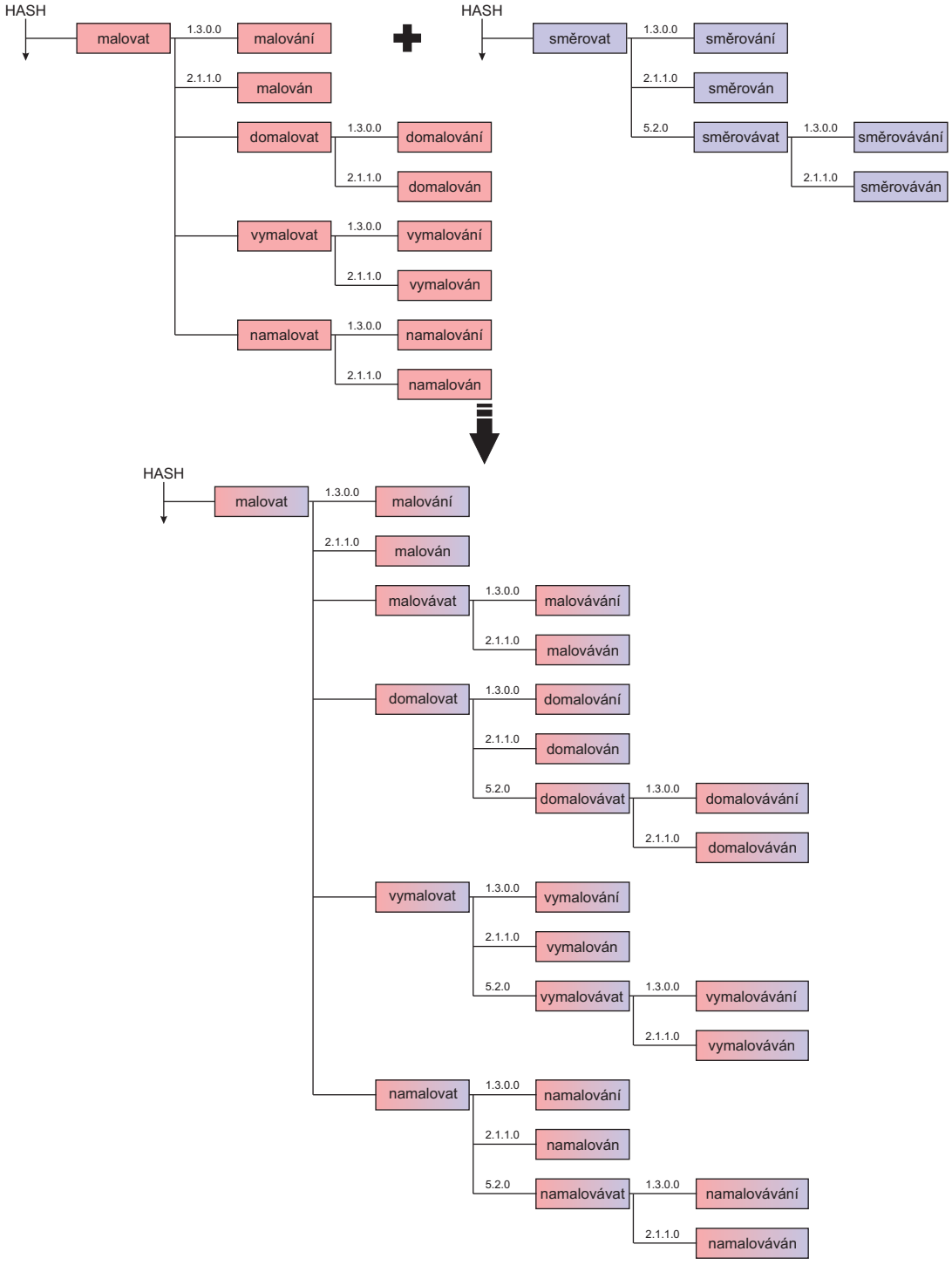
6.4 Aplikace seskupení

Poté co program dokončil seskupení, mohl rozgenerovat chybějící slova. Pokud slovo ze seskupení netvořilo některou vazbu, byla dogenerována. I přes nedokonalý způsob seskupování program vytvořil dalších 25 tisíc slov. Očekávám, že při lepším principu propojování slov, bude program mnohem přesnější, a proto na něm bude vhodné i nadále pracovat.

6.5 Vyhodnocení

Program tedy rozšířil derivační znalosti programu MA o 100 tisíc slov a dovoluje automatickou analýzu i na slova nově přidávaná, takže slova budou moci být vzájemně propojována i při rozšiřování slovníku MA.

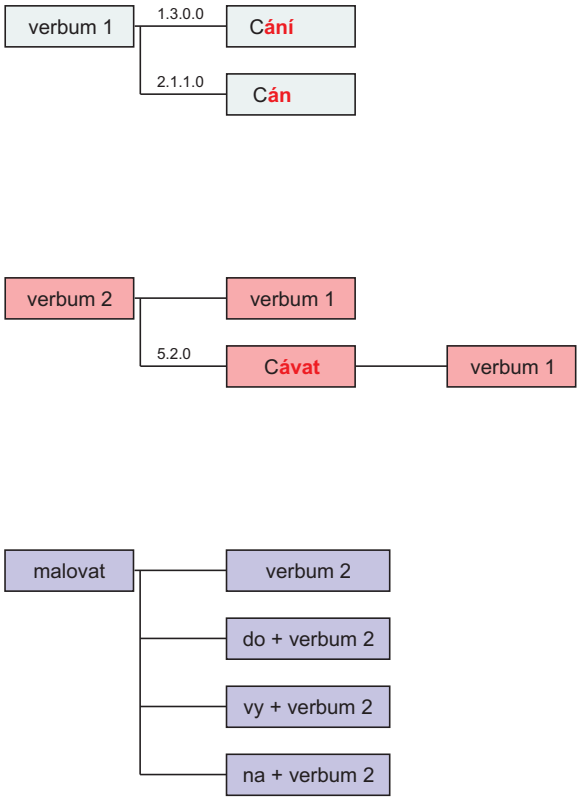
Celkem se ve slovníku MA nachází kolem 370 tisíc lemmat, a program jich dokázal propojit okolo 345 tisíc. Při zpřesnění seskupování slov, bude program schopen obsáhnout téměř všechna slova a poté bude jeho analýza velmi kvalitní. Přesto již nyní se program



Obrázek 6.15: Sjednocení seskupení (vznik redundance)

přibližuje k plnému propojení slovníku.

Program pracuje s lineární paměťovou složitostí. Vzhledem k vysokému počtu zpracovávaných dat vyžaduje zhruba 500 MB paměti. Již méně příznivá je časová složitost programu, protože je při seskupování vazeb kvadratická, a tak program pracuje několik hodin. Aby se tyto nároky pro další spuštění minimalizovali, ukládají se jednotlivé důležité mezivýsledky



Obrázek 6.16: Využití substituce

do souborů a program je tedy nepočítá znovu.

6.6 Směry rozvoje

Jak již bylo mnohokrát řečeno, analýza českého jazyka není věc jednoduchá. Program sám, vykazuje množství chyb, způsobené množstvím výjimek, která se v pravidlech nacházejí. Bude třeba další práce, aby se schopnosti programu staly přesnějšími.

Pravděpodobně bude potřeba lidského aparátu, aby vytřídil chybně vytvořené vazby ze seskupení, podle kterých se pracuje. Dále by bylo vhodné upravit systém seskupování, aby mohl být použit výše zmiňovaný postup. Bude třeba podrobná analýza vztahů a vlastností, která zaručí, že se prvky v seskupení nebudou opakovat (ani, pokud budou vytvořeny přes podobné skupiny).

Tyto a jiné úpravy již na práci nestihly být dokončeny. Slovotvorba je rozsáhlá oblast českého jazyka a bude třeba letité, práce než bude plně hotova. Přesto doufám, že práce přinesla mnoho úspěchů a že na ni bude možné navazovat.

Kapitola 7

Závěrem

Jak bylo naznačeno v úvodu celé této práce, je slovotvorba důležitá k rozpoznávání neznámých slov, která byla vytvořena pomocí pravidel slovotvorby. Lidé, tak mohou pojmenovat činnost odvozenou z jakéhokoli předmětu, nebo naopak. Jelikož je český jazyk, tak jako jakýkoliv jiný přirozený jazyk, proměnlivý, je důležité vytvořit aparát, který by byl schopen se změnami pracovat. Budou potřeba programy, které budou z nespisovných slov vytvářet spisovná, nebo z cizích slov česká, či měnit slovní druhy, aby mohli spolehlivě rozhodnout, co které slovo znamená. Je třeba slova dokonale pochopit a analyzovat, aby bylo možné v budoucnu to stejné udělat s větami. Znalost vět, dovolí pracovat se stále většími celky, až možná jednoho dne budou počítače odvozovat z kontextu celé strany textu, a nakonec třeba vyvodí postoj jednoho díla či knihy vůči druhému.

Na jednu stranu jsou tyto cíle příliš odvážné a smyšlené, na druhou stranu přesně tohle je směr, kterým se analýza jazyka pomalu posouvá. Je nemožné určit kdy, ale je pravděpodobné, že jednou budou počítače schopny změny v jazyce pochopit.

Tato práce se zabývala slovotvorbou. Snažila se počítači vštěpit sémantické znalosti a derivativní vazby mezi slovy. Sice se nepodařilo vytvořit program, schopný plně pracovat se všemi derivacemi, které čestina umožňuje, ale přesto věřím, že tato práce je krokem ke komplexní schopnosti slovotvorby a automatického chápání jazyka.

Literatura

[1] BAUER, A.: *Dělení slov: slovotvorba v praxi.* Nakladatelství Olomouc, první vydání, 1997, ISBN 80-7182-035-0.

[2] DACIUK, J.: *Incremental Construction of Finite-State Automata and Transducers, and their Use in the Natural Language Processing.* Diplomová práce, Gdansk: Technical University of Gdansk, Politechnika Gdanska, 1998.

[3] GREPL, M.; KARLÍK, P.; NEKULA, M.; aj.: *Příruční mluvnice češtiny.* Nakladatelství Lidové noviny, Praha, druhé vydání, 2008, ISBN 978-80-7106-980-5 (váz.).

[4] HAJIČ, J.: *Anotace víceslovných jednotek.*
URL <http://ufal.mff.cuni.cz/pdt2.5/cs/documentation.html>

[5] HAVRÁNEK, B.; JEDLIČKA, A.: *Česká mluvnice: vysokoškolská učebnice.* Státní pedagogické nakladatelství, Praha, Šesté vydání, 1988.

[6] HONZÍK, J. M.: IAL Algoritmy, studijní opora. [cit. 2013-05-13][Online] <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IAL-IT/texts/Opora-IAL-2012-verze-12-M.pdf>.

[7] MATOUŠEK, P.: ISA Síťové aplikace a správa sítí, studijní opora. [cit. 2013-05-13][Online] <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ISA-IT/texts/kapitola9.pdf>.

[8] MEDUNA, A.; LUKÁŠ, R.: IFJ Formální jazyky a překladače, studijní opora. [cit. 2013-05-13][Online] <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IFJ-IT/texts/OporaIFJ.pdf>.

[9] ROŽNOVSKÝ, L.: *Přirozené vs. umělé jazyky.*
URL <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IAL-IT/texts/Opora-IAL-2012-verze-12-M.pdf>

[10] RUSÍNOVÁ, Z.: *Tvoření slov v současné češtině.* Masarykova univerzita, Brno, druhé vydání, 1990, ISBN 80-210-0133-X.

[11] SEDLÁČEK, R.: *Morfologický analyzátor češtiny.* Diplomová práce, Brno: Masarykova univerzita, Fakulta informatiky, 1999.

[12] SOJKA, P.; kolektiv: *Laboratoř elektronických multimediálních aplikací.*
URL <http://www.fi.muni.cz/lemma/index.html>

[13] ŠMERK, P.: *K počítačové morfologické analíze češtiny.* Diplomová práce, Brno: Masarikova univerzita, Fakulta informatiky, 2010.

[14] ŠMILAUER, V.: *Novočeské tvoření slov: vysokošk. příručka.* Státní pedagogické nakladatelství, Praha, první vydání, 1971.

[15] ČERNÝ, S.: *Morfologický analyzátor pomocí konečných automatů.* Diplomová práce, Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, 2008.

[16] ČERNÝ, S.: *Analýza slovotvorných vazeb v češtině.* Diplomová práce, Brno: Vysoké učení technické v Brně, Fakulta informačních technologií, 2010.

[17] ČERVENÁ, V.; FILIPEC, J.: *Slovník spisovné češtiny pro školu a veřejnost s dodatkem Ministerstva školství, mládeže a tělovýchovy České republiky.* Academia, druhé vydání, 1998, ISBN 80-200-0493-9.

Příloha A

Obsah CD

K práci je přiloženo CD, na kterém se nalézá:

Zdrojové kódy programu

Soubor README vysvětlující základní rozhraní programu

Soubor BP_xrysav17_2013.pdf, což je tato práce v elektronické podobě

Soubor *deriv.data.orig*, ze kterého se v práci vycházelo a je důležitý pro běh programu

Složka *doc* obsahující soubory v latexu pro vygenerování práce v elektronické podobě

Složka *plakat* obsahuje motivační plakát v různé kvalitě

Složka *Vystupy*, která pro ilustraci obsahuje některé výstupní soubory programu

Příloha B

Seznam použitých vazeb

Zde bude vysvětleno, co všechny zmíněné vazby sémanticky znamenají:

1.0.5.0 Odvození substantiv z názvů oborů, stylů nebo směrů. Přidáním přípony -t místo -cie

1.0.5.3 Odvození substantiv z názvů oborů, stylů nebo směrů. Přidáním přípony -ista místo -ismus

1.0.6.0 Přechylování

1.0.6.1 Přechylování. Pojmenování ženy, podle jména muže nebo jeho pozice

1.0.7.0 Jména rodin

1.0.7.1 Jména rodin

1.0.7.2 Jména rodin

1.3.0.0 Substantiva verbální. Označení názvu činnosti, při které se vykonává činnost popsaná výchozím slovesem

1.3.1.4 Jména činností postupů. Z podstatného jména popisující politický systém, či disciplínu končící na -ie

1.7.1.1 Zdrobněliny navzájem složené

2.0.1.0 Přídavná jména z přechodníků přítomných

2.1.0.0 Přídavná jména slovesná tvořená z příčestí činného

2.1.1.0 Přídavná jména slovesná tvořená z příčestí trpného

2.2.0.0 Přídavná jména rodu mužského

2.2.1.0 Přídavná jména rodu ženského

2.3.0.0 Přídavná jména se vztahem k místu nebo druhu vytvořená z podstatných jmen příponou -ský

2.3.0.1 Přídavná jména se vztahem k místu nebo druhu vytvořená z podstatných jmen příponou -cký

5.0.0 Slovesa odvozená z podstatných jmen

5.2.0 Slovesa odvozená ze sloves

6.1.0 Příslovce odvozené z přídavných jmen

Příloha C

Plakát

Obrázek C.1: Inspirační plakát