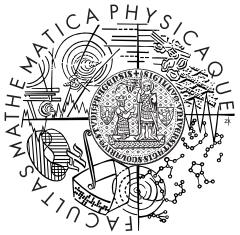
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

Disertační práce



Drahomíra „johanka“ Spoustová

Kombinované statisticko-pravidlové metody značkování češtiny

(Formální popis českých vět a otázky jeho implementace)

Ústav formální a aplikované lingvistiky
Školitel: doc. RNDr. Karel Oliva, Dr.

Největší zásluhu na vzniku této práce má (po autorce :)) bezesporu fenomenální programátor Pavel Květoň, bez jehož programovacího jazyka LanGR by vize zde nastíněné nemohly být nikdy uvedeny v život. Též mi velmi pomohlo, že na mnoho věcí nepřišel již dříve on sám, a zbyly tedy na mě :).

Dalšími, jejichž zásluha na závěrečném výsledku není nikterak abstraktní, nýbrž zcela konkrétní, jsou autoři statistických taggerů použitých v kombinovaných metodách značkování Jan Hajič, Pavel Krbec a Jan Votrubec a autoři lingvistických disambiguačních pravidel Niki Petkevič a Tomáš Jelínek.

Karlovi Olivovi a Janu Hajičovi jsem vděčná zejména za jejich schopnost nadhledu a velmi cenné nápady a rady, Karlovi navíc i za to, že tuto práci několikrát velmi pečlivě přečetl :).

Za podporu odbornou, osobně-mezilidskou i finanční děkuji svým pracovištím a kolegům z nich (Ústav formální a aplikované lingvistiky UK MFF a Ústav teoretické a komputační lingvistiky UK FF). Během svého studia jsem byla finančně podporováma granty Ministerstva školství, mládeže a tělovýchovy MSM0021620838 a LC536.

Nemohu samozřejmě opomenout svého muže Mirka, jenž mě podporoval nejen svou chápavostí (oželení teplých večeří krátce před deadlines apod.), nýbrž často i konkrétní programátorskou („Jakto, že to neznáš, vždyť's to učila!“), debugovací či lingvistickou výpomocí. V neposlední řadě mi svým neustálým tropením si bohapustých žertů zabránil prožívat záležitosti podmíněného světa více, než si zaslouží.

Tuto práci bych ráda věnovala naší dceři Helence, která byla v prenatálním a později kojeneckém věku přítomna sepisování, jakož i růstu úspěšnosti, který kopíroval růst plodu. Doufám, že ji to nikterak nepoznamenalo a že z ní vyroste něco normálního.

Obsah

Úvod

Důležitý je začít a třeba úplně blbě, ale pak negací toho blbýho získáme to správný řešení!

Karel Oliva

Tématem této práce jsou metody pro morfologické značkování (tagging) češtiny, zejména pak nejnovější experimenty s kombinováním pravidlových a statistických metod. Okrajově se zmíníme též o možnostech spolupráce pravidel a statistiky při parsingu.

Autorka se osobně podílela na vývoji pravidly řízené morfologické disambiguace a provedla veškeré v této práci popsané kombinační experimenty, přičemž většinu z nich i navrhla, ve zbylých případech navazovala na dřívější experimenty (zejména Pavla Květoně), které byly prováděny s ranou fází pravidel a jiným statistickým taggerem. Dále navrhla a provedla experiment s rozšířením působnosti systému pravidel do oblasti syntaxe.

V první kapitole zavedeme základní pojmy, specifikujeme rozsah řešeného úkolu a stručně shrneme dosavadní vývoj značkování češtiny. Podrobněji si přiblížíme projekt pravidly řízené morfologické disambiguace, jakož i tagger pracující na principu průměrovaného perceptronu, neboť zejména tyto dvě metody byly pro kombinované značkování využity.

Ve druhé kapitole podrobně popíšeme metodu automatického převodu valenčního slovníku sloves na valenční slovník deverbativních adjektiv, kterou jsme vyvinuli pro účely projektu pravidlové disambiguace, lze ji však využít obecně. Tato kapitola s ostatními souvisí pouze tematicky, není s nimi přímo provázána, lze ji tedy číst zcela samostatně.

Ve třetí kapitole, těžišti práce, popíšeme a vyhodnotíme veškeré provedené experimenty s kombinováním disambiguačních pravidel a statistických taggerů, shrneme i experimenty prováděné dříve jinými autory, a kde je to možné, pokusíme se odvodit obecnější závěry.

Ve čtvrté kapitole popíšeme (nepříliš vydařený) experiment s rozšířením působnosti systému disambiguačních pravidel do oblasti syntaxe. Pokusíme se co nejpřesněji stanovit, proč byl tento pokus odsouzen k nezdaru a na co by si měl dát pozor ten, kdo by se na tuto otázku chtěl zaměřit v budoucnu.

V poslední kapitole shrneme nejdůležitejší výsledky a závěry této práce.

Kapitola 1

Výchozí situace

Morfologické značkování je jedním ze základních kroků při automatickém zpracování přirozeného jazyka, zejména pak jazyka tak morfologicky bohatého, jako je čeština.

Proces je zvykem rozdělovat na dvě samostatné části, *morfologickou analýzu*, tj. přiřazení všech kombinací značek a lemmat přípustných pro daný izolovaně stojící slovní tvar, zpravidla na základě slovníku, a *morfologické zjednoznačnění (disambiguace, tagging)*, tj. zvolení právě jedné dvojice lemma-značka z nabídky morfologické analýzy, a to především na základě kontextu. Tato druhá fáze bývá někdy sama o sobě označována jako morfologické značkování. Pro formální definici morfologické analýzy a taggingu viz [5].

Po provedení morfologické analýzy je možné místo taggingu (nebo jako krok před ním) provádět i disambiguaci částečnou, tj. sice omezovat původní morfologickou nabídku, nikoli však nutně na pouhý jediný výsledek. Blíže tento způsob rozebereme v oddílu popisujícím projekt pravidly řízené morfologické disambiguace.

Morfologickou analýzou jsme se při naší práci nezabývali (její výsledek byl pro nás vstupem), proto ji podrobněji přiblížíme jen v rámci úvodní kapitoly.

Morfologické disambiguaci se naopak budeme věnovat v míře bohaté, neboť je hlavním tématem této práce. Naším úkolem bylo vyzkoušet různé způsoby kombinování již existujících metod značkování (po jejich případné modifikaci), a pokud možno dosáhnout toho, aby kombinace co do úspěšnosti překonala dosud nejúspěšnější metodu samostatnou (tagger Morče, [30]). Tento cíl se nám podařilo splnit.

V této kapitole nejprve stručně popíšeme morfologickou analýzu a následně shrneme situaci, která na poli značkování českých textů panovala před vyvinutím naší kombinované metody. Představíme tedy všechny významnější statistické taggery, které byly a jsou pro češtinu používány, jakož i projekt pravidly řízené morfologické disambiguace. Tento průřez je důležitý nejen kvůli přehledu a srovnání, ale i proto, že většina z těchto metod byla v naší kombinaci nakonec s úspěchem využita.

Náš výčet zdaleka neobsahuje všechny taggery, které kdy pro češtinu existovaly, nicméně nechybí žádný z těch, které jsou aktivně udržované, natrénované na aktuálních datech (PDT 2.0, [7]) a dosahují rozumné úspěšnosti (> 90 %). Uvedené taggery tedy mají všechny předpoklady pro to, aby bylo možné je v současné době používat, a to buď samostatně, nebo v kombinaci.

V závěru kapitoly ještě stručně přiblížíme data, která byla použita pro trénování statistických taggerů a pro testování všech metod, tedy statistických, pravidlových i kombinovaných.

1.1 Morfologická analýza češtiny

Pro češtinu v současné době existují dva morfologické analyzátory: *česká morfologie* Jana Hajiče [5] („pražská morfologie“) a *Ajka* Radka Sedláčka [28] („brněnská morfologie“). Tagsety (množiny možných značek) těchto morfologických systémů jsou na sebe s trochou brutality vzájemně převoditelné, úplná kompatibilita však dosud není vyřešena a k rutinním převodům ani ke spolupráci těchto systémů nedochází. Z tohoto důvodu jsme se v naší práci i v následujícím popisu omezili na pražskou morfologii a disambiguační systémy pro ni dostupné. Pokud je nám známo, žádný z nich není s touto morfologií svázán do hloubky, všechny by bylo možné přizpůsobit jinému morfologickému analyzátoru s odlišným tagsetem, u statistických modelů samozřejmě za nutné podmínky dostupnosti odpovídajícího množství trénovacích dat. Platí samozřejmě i opačná implikace, tedy že teoreticky je možné pro naše účely zkusit využít i brněnský hybridní tagger [25] založený na kombinaci pravidel a skrytých Markovových modelů.

V následujícím textu ve stručnosti popíšeme hlavní rysy české morfologie (tedy morfologického analyzátoru tohoto jména), a to zejména z uživatelského hlediska. Podrobnosti o tvaru slovníku, derivačních vzorech a dalších vnitřních záležitostech lze nalézt v monumentální monografii [5].

1.1.1 Morfologické kategorie

Morfologický analyzátor rozlišuje tyto morfologické kategorie:

Slovní druh (POS)

Základní rozdělení slov: podstatná jména (substantiva, N), přídavná jména (adjektiva, A), zájmena (pronomina, P), číslovky (numeralia, C), slovesa (verba, V), příslovce (adverbia, D), předložky (prepozice, R), spojky (konjunkce, J), částice (partikule, T), citoslovce (interjekce, I). Další možné hodnoty jsou Z – interpunkce a X – neznámý, neurčený, neurčitelný slovní druh.

Toto rozdělení je shodné s rozdělením užívaným v základním jazykových příručkách (např. Česká mluvnice [8]). Pokud je nám známo, morfologická analýza se od něj neodchyluje ani v případech, kde je držení se této normy spíše na škodu – např. zájmena a číslovky tvoří z morfologického a syntaktického hlediska velmi různorodou směs, kterou je pro účely automatického zpracování jazyka vhodné

spíše rozdělit podle toho, zda se jednotlivá slova chovají jako adjektiva, či jako substantiva. Toto rozšíření je implementováno v rámci pravidlového disambiguačního projektu.

Detailní určení slovního druhu (SUBPOS)

Podrobněji rozčleňuje některé slovní druhy do podkategorií (například předložky je možné dělit na vokalizované a nevokalizované, zájmena na vztažná, neurčitá, osobní atd.), pro velký rozsah nebudeme uvádět úplný výčet (viz např. [5]), neboť pro práci není podstatný. Slovní druh je jednoznačně určen detailním slovním druhem, opačně to samozřejmě neplatí.

Jmenný rod (GENDER)

Hodnoty jednoznačné: F – femininum (ženský rod), I – maskulinum inanimatum (rod mužský neživotný), M – maskulinum animatum (rod mužský životný), N – neutrum (střední rod).
Hodnoty sdružující více možných variant: H – femininum nebo neutrum (tedy nikoli maskulinum), Q – femininum singuláru nebo neutrum plurálu (pouze u příčestí a jmenných tvarů adjektiv), T – masculinum inanimatum nebo femininum (jen plurál u příčestí a jmenných tvarů adjektiv), X – libovolný rod (F/M/I/N), Y – masculinum (animatum nebo inanimatum), Z – „nikoli femininum“ (tj. M/I/N).

Číslo (NUMBER)

Hodnoty jednoznačné: D – duál, P – plurál (množné číslo), S – singulár (jednotné číslo).
Hodnoty víceznačné: W – pouze v kombinaci s jmenným rodem *Q* (singulár pro feminina, plurál pro neutra), X – libovolné číslo (P/S/D).

Pád (CASE)

Hodnoty jednoznačné: 1 – nominativ (1. pád), 2 – genitiv (2. pád), 3 – dativ (3. pád), 4 – akuzativ (4. pád), 5 – vokativ (5. pád), 6 – lokál (6. pád), 7 – instrumentál (7. pád).
Hodnoty víceznačné: X – libovolný pád (1/2/3/4/5/6/7).

Přivlastňovací rod (POSSGENDER)

Hodnoty jednoznačné: F – femininum (ženský rod), M – maskulinum animatum (rod mužský životný).
Hodnoty víceznačné: X – libovolný rod (F/M/I/N), Z – „nikoli femininum“ (tj. M/I/N).

Přivlastňovací číslo (POSSNUMBER)

P – plurál (množné číslo), S – singulár (jednotné číslo).

Osoba (PERSON)

Hodnoty jednoznačné: 1 – 1. osoba, 2 – 2. osoba, 3 – 3. osoba.
Hodnoty víceznačné: X – libovolná osoba (1/2/3).

Čas (TENSE)

Hodnoty jednoznačné: F – futurum (budoucí čas), P – prézens (přítomný čas), R – minulý čas.
Hodnoty víceznačné: H – minulost nebo přítomnost (P/R), X – libovolný čas (F/R/P).

Stupeň (GRADE)

1 – 1. stupeň, 2 – 2. stupeň, 3 – 3. stupeň.

Negace (NEGATION)

A – afirmativ (bez negativní předpony *ne-*), N – negace (tvar s negativní předponou *ne-*).

Slovesný rod (VOICE)

A – aktivum nebo „nikoli pasívum“, P – pasívum.

Varianta, stylový příznak apod. (VAR)

(Toto není v pravém slova smyslu morfologická kategorie, nicméně morfologická analýza ji alespoň po technické stránce klade na roveň ostatním vlastnostem zpracovávaného slova.)

1 – varianta, víceméně rovnocenná („méně častá“), 2 – řídká, archaická nebo knižní varianta, 3 – velmi archaický tvar, též hovorový, 4 – velmi archaický nebo knižní tvar, pouze spisovný (ve své době), 5 – hovorový tvar, ale v zásadě tolerovaný ve veřejných projevech, 6 – hovorový tvar (koncovka obecné češtiny), 7 – hovorový tvar (koncovka obecné češtiny), varianta k 6, 8 – zkratky, 9 – speciální použití (tvary zájmen po předložkách apod.).

Pro úplnost dodáme, že z morfologického slovníku lze získat i slovesný vid, v současné době však není součástí značky, proto ho standardní implementace morfologické analýzy neposkytuje. Lze použít (nepublikovaný) program Miroslava Spousty, který vid do značky přidá, a to buď na jednu z rezervních pozic, nebo na jinak neexistující šestnáctou. Tento program byl použit mimo jiné při značkování korpusů SYN2005 a SYN2006PUB (viz [1]).

1.1.2 Poziční systém tagů

Poziční tag, standardní výstup morfologické analýzy, se skládá z patnácti znaků, každá kategorie má určeno pevné číslo pozice, na které

se vyskytují znaky reprezentující hodnoty dané kategorie. Přiřazení je následující:

1	slovní druh (POS)
2	detailní určení slovního druhu (SUBPOS)
3	rod (GENDER)
4	číslo (NUMBER)
5	pád (CASE)
6	přivlastňovací rod (POSSGENDER)
7	přivlastňovací číslo (POSSNUMBER)
8	osoba (PERSON)
9	čas (TENSE)
10	stupeň (GRADE)
11	negace (NEGATION)
12	slovesný rod (VOICE)
13	rezervováno (RESERVE1)
14	rezervováno (RESERVE2)
15	varianta, stylový příznak apod. (VAR)

V závorce jsou uvedeny zkratky používané pro dané kategorie.

Coby příklad uvedeme možné poziční tagy pro slovní tvar *zdraví*.

Lemma *zdraví*:

`NNNP1-----A----` `NNNP2-----A----` `NNNP4-----A----` `NNNP5-----A----`
`NNNS1-----A----` `NNNS2-----A----` `NNNS3-----A----` `NNNS4-----A----`
`NNNS5-----A----` `NNNS6-----A----`
(substantivum středního rodu v rozličných číslech a pádech, afirmativ)

Lemma *zdravit*:

`VB-P---3P-AA---` `VB-S---3P-AA---`
(sloveso v singuláru nebo v plurálu, třetí osoba, přítomný čas, afirmativ, aktivum)

Lemma *zdravý*:

`AAMP1----1A----` `AAMP5----1A----`
(adjektivum v mužském životném rodě, plurál, nominativ nebo vokativ, první stupeň, afirmativ)

1.1.3 Kompaktní systém tagů

Kompaktní systém zápisu tagů nemá pevně danou velikost (co do počtu znaků), je starší a v současné době se používá především v morfologickém slovníku. Jeho výhodou je menší velikost oproti pozičnímu systému. Kompaktní tag se skládá ze tří částí:

- **Prefix** určuje slovní druh a rozšířený slovní druh (např. VF pro slovesný infinitiv, DB pro příslovce bez možnosti negace a stupňování).
- **Morfologické kategorie** obsahují hodnoty (pole 3–12 pozičního systému), které jsou pro daný prefix relevantní. Například pro substantiva je to rod+číslo+pád+negace, kompaktní tag pro akuzativ slova *možnost* bude tedy `NFS4A`.
- Pole **varianta, styl** je od předcházející části tagu odděleno pomlčkou.

Mezi oběma systémy existuje vzájemně jednoznačné přiřazení (jsou na sebe vzájemně převoditelné).

1.1.4 Rozšířený tagset

Projekt pravidly řízené morfologické disambiguace mírně modifikuje a rozšiřuje seznam informací poskytovaných morfologickou analýzou, upravuje tedy automaticky její výstup (v pozičním systému), a to do podoby tzv. rozšířeného (wide) tagsetu. Interpret jazyka LanGR [16], který tyto změny provádí, je schopen načítat vstup a produkovat výstup v původním i v rozšířeném tagsetu. Hlavní změny jsou následující:

- zrušení víceznačných pádů, rodů, čísel, osob a časů (nejčastěji reprezentovaných písmenem X) – příslušné značky jsou distribuovány na všechny přípustné kombinace
- u slovních druhů, které mohou být separátory klauzí (spojky a interpunkční znaménka), je informace o tom, zda jimi skutečně jsou, přidána do značky – z každé vstupní značky jsou tedy vygenerovány dvě varianty, separující a neseparující
- číslovkám psaným číslicemi jsou přiřazeny stejné množiny tagů jako jejich slovním ekvivalentům.

Jelikož zejména po první z popsaných úprav volá v současné době více aplikací, očekáváme, že bude časem zahrnuta do samotné morfologie, tato změna ovšem bude vyžadovat opravy ručně anotovaných dat (PDT apod.) pro zajištění kompatibility.

Veškeré výstupy metod prezentovaných v této práci jsou v původním, nerozšířeném tagsetu, rozšířený tagset je používán pouze uvnitř interpretu jazyka LanGR pro potřeby lingvistických pravidel.

1.1.5 Příklad

Jako příklad práce morfologické analýzy uvedeme větu obsahující několik pro češtinu typických homonymií, kterou jsme si dovolili vypůjčit z práce [30].

Příklad 1.1–1: *Na tři hlavní podezřelé byla uvalena vyšetřovací vazba.*

Forma	Lemma	Značky
Na	na	`RR--4---------- RR--6----------`
tři	třít	`Vi-S---2--A---- Vi-S---3--A---4`
	tři	`ClXP1---------- ClXP4---------- ClXP5----------`
hlavní	hlaveň	`NNFP2-----A---- NNFS7-----A----`

	hlavní	`AAFP1----1A---- AAFP4----1A---- AAFP5----1A----`
		`AAFS1----1A---- AAFS2----1A---- AAFS3----1A----`
		`AAFS4----1A---- AAFS5----1A---- AAFS6----1A----`
		`AAFS7----1A---- AAIP1----1A---- AAIP4----1A----`
		`AAIP5----1A---- AAIS1----1A---- AAIS4----1A----`
		`AAIS5----1A---- AAMP1----1A---- AAMP4----1A----`
		`AAMP5----1A---- AAMS1----1A---- AAMS5----1A----`
		`AANP1----1A---- AANP4----1A---- AANP5----1A----`
		`AANS1----1A---- AANS4----1A---- AANS5----1A----`

podezřelé	podezřelý	`AAFP1----1A---- AAFP4----1A---- AAFP5----1A----`
		`AAFS2----1A---- AAFS3----1A---- AAFS6----1A----`
		`AAIP1----1A---- AAIP4----1A---- AAIP5----1A----`
		`AAMP4----1A---- AANS1----1A---- AANS4----1A----`
		`AANS5----1A----`

byla	být	`VpQW---XR-AA---`
uvalena	uvalit	`VsQW---XX-AP---`

vyšetřovací	vyšetřovací	`AAFP1----1A---- AAFP4----1A---- AAFP5----1A----`
		`AAFS1----1A---- AAFS2----1A---- AAFS3----1A----`
		`AAFS4----1A---- AAFS5----1A---- AAFS6----1A----`
		`AAFS7----1A---- AAIP1----1A---- AAIP4----1A----`
		`AAIP5----1A---- AAIS1----1A---- AAIS4----1A----`
		`AAIS5----1A---- AAMP1----1A---- AAMP4----1A----`
		`AAMP5----1A---- AAMS1----1A---- AAMS5----1A----`
		`AANP1----1A---- AANP4----1A---- AANP5----1A----`
		`AANS1----1A---- AANS4----1A---- AANS5----1A----`

vazba	vazba	`NNFS1-----A----`
.	.	`Z:------------`

Velká část slovních tvarů je zde bohatě homonymní, některé i slovnědruhově (*tři* a *hlavní*). I při správném zvolení slovního druhu má *hlavní* stále ještě 27 možností pro výběr značky, stejně tak slovo *vyšetřovací*. Je zde zastoupena i pádová homonymie předložky *na*, která je velmi rozšířená a často i obtížně řešitelná.

Správné značkování této věty (cíl práce taggeru) by mělo vypadat následovně [1]):

Forma	Lemma	Značka
Na	na	`RR--4----------`
tři	tři	`ClXP4----------`
hlavní	hlavní	`AAMP4----1A----`
podezřelé	podezřelý	`AAMP4----1A----`
byla	být	`VpQW---XR-AA---`
uvalena	uvalit	`VsQW---XX-AP---`
vyšetřovací	vyšetřovací	`AAFS1----1A----`
vazba	vazba	`NNFS1-----A----`
.	.	`Z:-------------`

Tagger by tedy měl rozpoznat, že *tři* je číslovka, *hlavní* adjektivum, předložková skupina *Na tři hlavní podezřelé* je v akuzativu, plurálu a mužském rodě a že adjektivum *vyšetřovací* je v nominativu, singuláru a ženském rodě.

[1]) Bystrý čtenář se možná už teď ozve se zvídavou otázkou, ostatní se dočkají dějových zvratů v průběhu kapitoly.

1.2 Tagger Morče

Tagger Morče, který je podrobně popsán v diplomové práci Jana Votrubce [30], je v současné době nejúspěšnějším samostatným taggerem pro češtinu. Veškeré experimenty založené na kombinaci pouze jednoho statistického taggeru s pravidly jsme ladili primárně na něm, a výsledky těchto experimentů také byly vždy lepší než u jiných taggerů. Z uvedených důvodů mu také budeme věnovat nejvíce pozornosti.

Pro řešení úkolu morfologického značkování zde byla použita statistická učící se metoda založená na koncepci skrytého Markovova modelu (HMM) a průměrovaného perceptronu, která byla popsána v článku Michaela Collinse [2].

1.2.1 HMM

Skrytý Markovův model se obvykle používá tam, kde je třeba jednu sekvenci informací převádět na jinou, přičemž lze předpokládat, že tento převod je určen pouze historií omezené délky. Zde dochází k převodu sekvence slov na sekvenci morfologických značek.

Pro trénování a následné použití tohoto modelu se používá Viterbiho algoritmus, který slouží k nalezení nejlépe ohodnocené výstupní sekvence pro zadanou vstupní sekvenci.

Viterbiho algoritmus se zpravidla aplikuje na HMM, kde ohodnocením jednotlivých přechodů jsou pravděpodobnosti. Algoritmus Morčete, tzv. průměrovaný perceptron, používá místo pravděpodobností váhové koeficienty. Ohodnocením jsou buď celá čísla (ve fázi trénování), nebo čísla reálná (ve fázi testování).

1.2.2 Rysy

Rys může obsahovat jakoukoli informaci, kterou jsme schopni o vstupní větě získat, může tedy popisovat např. značky, slovní formy, lemmata, pořadí ve větě, může tyto informace i libovolně kombinovat. Informace o vstupu se zjišťují pro každou pozici ve větě zvlášť. Obecné předpisy rysů (např. „aktuální značka“) se rozderivují na všechny možnosti (v našem příkladu celý tagset) a každý z těchto individuálních rysů pro danou pozici buď platí, nebo neplatí. Platné rysy tedy můžeme interpretovat jako popis aktuálního kontextu. Další interpretací může být, že rys (s určitou vahou) předpovídá aktuální značku z kontextu.

1.2.3 Průměrovaný perceptron

Úkolem průmerovaného perceptronu je uchovávat váhové koeficienty všech rysů a pro každou pozici v textu sčítat váhové koeficienty rysů platných v daném kontextu. Výsledek předává Viterbiho algoritmu coby pravděpodobnost přechodu. Formálně vyjádřeno:

$$w(C,T) = \sum_{i=1}^{n} \alpha_i.\phi_i(C,T)$$

kde $w(C,T)$ je přechodová pravděpodobnost pro tag T v kontextu C, n je počet rysů, α_i je váhový koeficient $i$tého rysu a $\phi_i(C,T)$ je evaluace $i$tého rysu pro kontext C a tag T.

1.2.4 Učící se algoritmus

Na začátku jsou váhové koeficienty (α) všech rysů nastaveny na nulu. V několika iteracích se procházejí celá vstupní data. Viterbiho algoritmus postupně vybírá nejlepší cestu (tj. nejlepší značky) pro každou větu s použitím aktuálních váhových koeficientů. Po dokončení každé věty dojde k aktualizaci váhových koeficientů. To se opakuje, dokud není dosaženo požadovaného počtu průchodů vstupními daty.

Aktualizace váhových koeficientů probíhá tak, že pro rysy odpovídající dané větě a algoritmem vybraným značkám jsou příslušně váhové koeficienty sníženy o 1, zatímco pro rysy odpovídající správným značkám jsou zvýšeny o 1. Při správném označkování věty tedy zůstávají koeficienty nezměněny.

1.2.5 Volba sady rysů

V rámci adaptace této metody pro češtinu bylo třeba zvolit vhodné rysy. Ačkoli autor taggeru prováděl poměrně rozsáhlé experimenty s automatickým vývojem sady rysů, nakonec se jako úspěšnější ukázal vývoj ruční, tedy volba „nadějných“ rysů na základě lingvistické intuice a následná optimalizace jejich sady na základě dosažených výsledků.

Z mnoha desítek verzí je v současné době nejúspěšnější verze s názvem *hepar*, která pracuje s následující množinou rysů:

- aktuální značka
- značkový bigram
- značkový trigram
- značkový bigram „ob slovo“
- aktuální forma

- bigram forem
- bigram forem ob slovo
- lemma předchozího slova
- forma následujícího slova
- pořadí slova ve větě (max. 7)
- nejbližší předchozí sloveso (lemma a značka)
- nejbližší následující možné sloveso (lemma a značka)
- velikost písmen formy a lemmatu
- částečný n-gram: jen SUBPOS a CASE – unigram
- SUBPOS a CASE – bigram
- SUBPOS a CASE – trigram

Je vidět, že vedle rysů spíše technické povahy pronikly do výběru i rysy lingvisticky podložené – sem patří zejména ohledávání nejbližšího slovesa (shoda), u bigramů ob slovo zase může jít o vynechatelné rozvití adverbiem.

1.3 Ostatní taggery

1.3.1 Feature-based tagger

Feature-based tagger, který byl implementován Janem Hajičem a je popsán mimo jiné v práci [5], je distribuován zároveň s morfologickou analýzou. Využívá exponenciálního modelu v základní formě

$$p_{AC}(y \mid x) = \frac{exp(\sum_{i=1}^{n} \lambda_i f_i(y, x))}{Z(x)}$$

kde $f_i(y, x)$ je binární hodnota předpovídaného jevu a jeho kontextu, λ_i je váha rysu f_i a $Z(x)$ je normalizační faktor.

Váhy λ_i jsou odhadovány pomocí metody maximální věrohodnosti. Odhad je nutný vzhledem k tomu, že možných rysů jsou řádově miliony a proto nelze přímo použít metodu maximální entropie.

Úspěšnost tohoto taggeru je zhruba o 1 % nižší než Morčete, pokud se jedná o celou značku, ale na některých pozicích (např. SUBPOS) jsou jeho výsledky lepší. Konkrétní výsledky a podrobnější porovnání bude následovat v kapitole o kombinovaných metodách značkování.

1.3.2 HMM tagger

Základem HMM taggeru, který byl implementován Pavlem Krbcem, je stejně jako u Morčete skrytý Markovův model (HMM). Oproti standardní implementaci HMM taggeru však obsahuje mnohé změny a vylepšení vycházející ze specifik češtiny a z výsledků průběžných experimentů (přihrádkové vyhlazování, obrácený směr průchodu větou (zprava doleva), počítání pravděpodobnosti na základě koncovky slova, části tagu apod.) Podrobný popis taggeru a jeho vylepšení se nachází v práci [15] a v článku [27].

Ačkoli je tento tagger svou koncepcí nejjednodušší z uvedených tří, dosahuje pozoruhodně dobrých výsledků (cca 0.5 % pod Morčetem).

1.4 Pravidly řízená disambiguace

To nezvládne ani sto Petkevičů! To by jich muselo bejt deset tisíc a dělali by na tom sto let!

Jan Hajič

Projekt pravidly řízené morfologické disambiguace vznikl v roce 2000 jako reakce na tehdejší situaci na poli morfologického značkování češtiny, konkrétněji pak jako reakce na výsledky značkování ČNK SYN2000 [1], které bylo provedeno tehdy dostupnými statistickými metodami.

Při zhruba pětiprocentní chybovosti taggeru to u stomiliónového korpusu znamená asi 5 milionů špatně přiřazených značek. To už je množství, které práci uživatelů korpusu značně ztěžuje, zároveň je to ale natolik rozsáhlý materiál, že v něm lze vysledovat jisté zákonitosti, tedy takové chyby taggerů, kterým by se hypoteticky dalo předzpracováním – částečným zredukováním morfologické nabídky před spuštěním taggeru – předejít.

Takto se zrodil projekt pravidly řízené morfologické disambiguace (viz [22]), na jehož počátku stáli Karel Oliva, Vladimír Petkevič, Milena Hnátková a Pavel Květoň. Základní myšlenky projektu jsou dvě:

- mnohým chybám statistického značkování lze předejít zredukováním morfologické nabídky na základě velmi bezpečných, lingvisticky motivovaných ručně psaných pravidel;
- úplná disambiguace není vždy žádoucí – je-li vstupní věta víceznačná, měla by disambiguace správně ponechat všechny přípustné významy (ale právě jen ty!). To žádný z dostupných statistických taggerů neumožňuje, ale byla-li by pravidlová disambiguace použita samostatně, snadno by to umožnit mohla.

Projekt se od svého vzniku stále vyvíjí, nyní obsahuje již přes 2000 pravidel psaných ve speciálním jazyce LanGR. Tento jazyk je obecnější než finská Constraint Grammar Freda Karlssona a kolektivu [13] [1]), která může být právem považována za vzor pro pravidlové a kombinované systémy značkování (viz též [29]) a s níž má disambiguační projekt samozřejmě mnoho společných rysů. Existuje i další podobný systém pro češtinu, který je posán v práci [18].

Pravidla popisují rozličná zákoutí české gramatiky a je možné je pouštět všechna najednou (v tom případě se provádějí cyklicky

[1]) Na rozdíl od ní obsahuje např. podmíněné unifikace, obecně má sílu Turingova stroje.

v náhodném pořadí), samostatně či po skupinkách. Autory většiny z nich jsou Vladimír Petkevič a Tomáš Jelínek. Kompilátor a interpret, jakož i samotný návrh jazyka LanGR jsou dílem Pavla Květoně (viz [16]).

Idealisté z řad členů projektu stále věří, že pravidla jednou dosáhnou takového vývojového stadia, že pomocí nich (a bez pomoci jiných metod) bude možné splnit obě výše uvedené základní myšlenky zároveň, tedy bezchybně morfologicky disambiguovat text se zachováním skutečné homonymie. Tuto vizi nepovažujeme za reálnou (k některým z důvodů se dostaneme v kapitole o kombinovaných metodách), proto jsme se pokusili napomoci splnění alespoň prvního bodu. To se nám, byt's netriviálními úpravami a rozšířeními původní myšlenky, podařilo, proto je možné s potěšením konstatovat, že pravidla jsou v tomto směru úspěšná a svůj hlavní účel, tedy vylepšit značkování pro další verze ČNK (SYN2005 a novější) splňují.

1.4.1 Motivační příklad – slovo *se*

V tomto oddílu demonstrujeme na konkrétním příkladu syntax a funkcionalitu disambiguačních pravidel.

Homonymie slůvka *se* (předložka vs. reflexivum) je pro statistické taggery vskutku obtížným oříškem – přitom je to nejčastěji se vyskytující homonymní slovo v češtině. Pravidla mohou napomoci tím, že v některých konkrétních případech nepřípustné čtení odstraní. V současné době má pravidlo pro disambiguaci *se* 28 variant (viz mimo jiné [19]), z nichž několik zde uvedeme na ukázku. Vždy doplníme příkladem věty z korpusu SYN2005, kterou tagger Morče označkoval chybně (čemuž bychom mohli předejít, pokud bychom předřadili dotyčné pravidlo).

- *Se* na začátku věty nemůže být reflexivní

  Příklad 1.4–1: *Se soudci zahájí okamžitě kárné řízení.*

Abychom byli spravedliví, uvedeme zde i protipříklad:

Příklad 1.4–2: *Se mu divím.*

V tomto případě, stejně jako u mnoha jiných užití nespisovného jazyka, pravidla bohužel selžou a odstraní i správnou značku.

- Předložka *se* nemůže stát těsně před interpunkčním znaménkem, spojkou ani slovesem.

  Příklad 1.4–3: *. . . měli zřetelný a profesionální projev a také jednotný styl představení se.*

  Příklad 1.4–4: *Vyhrál mnoho případů, chlubil se, ale tohle byl rozsudek. . .*

Příklad 1.4–5: *Mančaft jako zvon má Sparta, Pardubice se ale budou rvát jako koně.*

Příklad 1.4–6: *Mnozí neměli ani čas se převléci.*

- Předložka *se* nemůže stát těsně před slovem začínajícím na samohlásku (jedna z variant týkajících se vokalizace předložek).

  Příklad 1.4–7: *Vylezla z vany a natřela se olejem.*

Všechna uvedená pravidla jsou tzv. negativní, tedy popisující a odstraňující nepřípustné konfigurace. V jazyce LanGR, který se pro zápis pravidel používá, vypadají takto:

```
1  RuleVariant v6 {
2
3  PRE-SENTENCE ITEM Semicolon;
4  /* začátek věty nebo středník */
5
6  tvarse = ITEM lower form == "se";
7
8  DELETE Pronoun FROM tvarse;
9
10 };
11
12 RuleVariant v8 {
13
14 tvarse = ITEM lower form == "se";
15
16 ITEM (IsSafe Verb or Conjunction) or Colon or SemiColon or Period
 or ExclamationMark or QuestionMark;
17
18 DELETE Preposition FROM tvarse;
19
20 };
21
22 RuleVariant v9 {
23
24 tvarse = ITEM lower form == "se";
25
26 ITEM VowelInitialWord;
27 /* slovo začínající samohláskou */
28
29 DELETE Preposition FROM tvarse;
30
31 };
```

Na těchto triviálních pravidlech je snadno vidět, jak vypadá syntax pravidel obecně. Nejprve je uvedena konfigurační část, v níž je popsána část věty, na kterou se pravidlo má uplatnit, poté je uvedena akce, která se má provést. V uvedených příkladech se vždy hledá slůvko *se* v popsaném kontextu, a když se nalezne, maže se u něj jedno z možných morfologických čtení, tedy buď předložka, nebo reflexivní zájmeno.

Mazání se provádí bez ohledu na to, zda ještě nějaké jiné čtení zbývá, to je jedna ze základních myšlenek negativních pravidel. Pokud se stane, že u nějakého slova po průchodu pravidly nezůstane vůbec žádný tag (to může znamenat buď chybu v pravidlech, nebo vstup odporující gramatice spisovné češtiny), vrátí se na výstup všechny původní tagy, ale se speciální poznámkou, aby bylo možné je odlišit od tagů „nedotčených".

Vedle negativních pravidel existují v systému i pravidla pozitivní, tedy taková, která přímo vybírají správný tag (nebo skupinu tagů). Jako příklad si opět uvedeme jedno z pravidel pro disambiguaci *se*.

- Je-li ve větě s reflexivem tantum (ověřuje se na základě seznamu) jediné *se*, pak to musí být nutně reflexivum.

Pravidlo má čtyři varianty podle pořadí slovesa a reflexiva, uvedeme si jednu z nich:

```
 1  RuleVariant v26 {
 2
 3  ITEM SentenceStart;
 4
 5  SEQUENCE OF IsSafe Punctuation;
 6
 7  SEQUENCE OF ((lower form != "se") and (lower form != "ses"))
or (MustNotBe Pronoun);
 8  /* posloupnost slov, která nejsou reflexivem "se" */
 9
10  verbrefl = ITEM IsSafe VerbReflexiveSeOnly and not PassiveParticiple;
11  /* slovní tvar, který je reflexivum tantum - určení na základě
seznamu */
12
13  SEQUENCE OF ((lower form != "se") and (lower form != "ses"))
or (MustNotBe Pronoun);
14  /* posloupnost slov, která nejsou reflexivem "se" */
15
16  tvarse = ITEM lower form == "se";
17  /* tvar "se" - musí být reflexivum */
18
```

```
19  SEQUENCE OF (lower form != "se") or (MustNotBe Pronoun);
20  /* posloupnost slov, která nejsou reflexivem "se" */
21
22  POST-SENTENCE ITEM IsSafe ClauseSeparator;
23  /* konec věty nebo klauze */
24
25  LEAVE ONLY Pronoun IN tvarse;
26  /* ponech jen reflexivní čtení */
27
28  }; // konec varianty v26
```

Větší část pravidla tvoří obsáhlý popis konfigurace (je třeba popsat všechny úseky věty a vyloučit v nich reflexivní *se*), teprve řádek 25 specifikuje prováděnou akci, tedy výběr přípustné podmnožiny značek pro jediné nalezené *se*.

1.4.2 Homonymní věty

Příklad 1.4–8: *Ženu holí stroj.*

Příklad 1.4–9: *Brňáci čekají na nádraží.*

Příklad 1.4–10: *Nyní je má vrchní sestra v šuplíku.*

Příklad 1.4–11: *Jemnou dětskou pokožku chrání i pěstí.*

Uvedené věty mají krom jistého půvabu společné hlavně to, že jsou homonymní, první uvedená dokonce pětinásobně. Do klubu lze přibrat i větu, na níž jsme demonstrovali morfologickou analýzu a tagging...

Příklad 1.4–12: *Na tři hlavní podezřelé byla uvalena vyšetřovací vazba.*

...neboť tři hlavní podezřelé mohou být i ženy.

Jak má vypadat správný výsledek morfologické disambiguace takovýchto vět? Varianta „vybrat jen jedno náhodné čtení a ponechat značky, které mu příslušejí“ zřejmě poněkud pokulhává. Korektnější možnost je ponechat všechny značky, které příslušejí alespoň jednomu z přípustných čtení, a to buď pohromadě, nebo rozdělené mezi tato čtení. Výstup morfologie a „ideálního taggeru“ pro větu *Brňáci čekají na nádraží.* by tedy mohl vypadat takto:

Morfologie:

Forma	Lemma	Značky
Brňáci	Brňák	NNMP1-----A---- NNMP5-----A----
čekají	čekat	VB-P---3P-AA---
na	na	RR--4---------- RR--6----------
nádraží	nádraží	NNNP1-----A---- NNNP2-----A---- NNNP4-----A---- NNNP5-----A---- NNNS1-----A---- NNNS2-----A---- NNNS3-----A---- NNNS4-----A---- NNNS5-----A---- NNNS6-----A----
.	.	Z:----------

Tagger ve verzi se všemi značkami pohromadě:

Forma	Lemma	Značky
Brňáci	Brňák	NNMP1-----A----
čekají	čekat	VB-P---3P-AA---
na	na	RR--4---------- RR--6----------
nádraží	nádraží	NNNP4-----A---- NNNS4-----A---- NNNS6-----A----
.	.	Z:----------

Tagger ve verzi se značkami zvlášť, první čtení:

Forma	Lemma	Značky
Brňáci	Brňák	NNMP1-----A----
čekají	čekat	VB-P---3P-AA---
na	na	RR--4----------
nádraží	nádraží	NNNP4-----A----
.	.	Z:----------

Druhé čtení:

Forma	Lemma	Značky
Brňáci	Brňák	NNMP1-----A----
čekají	čekat	VB-P---3P-AA---
na	na	RR--4----------
nádraží	nádraží	NNNS4-----A----
.	.	Z:----------

Třetí čtení:

Forma	Lemma	Značky
Brňáci	Brňák	NNMP1-----A----
čekají	čekat	VB-P---3P-AA---
na	na	RR--6----------
nádraží	nádraží	NNNS6-----A----
.	.	Z:----------

Pravidly řízená morfologická disambiguace (na rozdíl od statistických taggerů) teoreticky umožňuje obě tyto varianty výstupu (první vydává automaticky, pro druhou by bylo možné interpret upravit). Problém je v tom, že systém sám o sobě dosud není dostatečně výkonný (spolu se správnými tagy dosud zůstávají ve výstupu i mnohé, které tam nepatří, nebot' je pravidla neumějí odstranit), a při kombinaci s jakýmkoli statistickým taggerem se výhoda možnosti víceznačného výstupu ztrácí.

1.5 Použitá data a evaluační metriky

Pro trénování a testování všech popsaných statistických taggerů byl použit pražský závislostní korpus (Prague Dependency Treebank) ve verzi 2.0 [7]. PDT je ručně anotován na morfologické, analytické a tektogramatické rovině, pro naše účely jsme využili pouze anotaci morfologickou. Je nutné podotknout, že veškeré publikace o českém taggingu starší než z roku 2005 (a i některé novější) uvádějí výsledky dosažené při použití dat z PDT 1.0 [26], přičemž kvalitativní rozdíl mezi těmito verzemi je velmi výrazný, proto nelze porovnávat samotná „čísla“ z těchto publikací s výsledky uvedenými v naší práci – aby bylo možné metody porovnávat, je bezpodmínečně nutné nejprve trénovací i testovací data sjednotit. Nicméně většinu významnějších kombinačních experimentů z minulosti jsme na nových datech zopakovali, proto čtenář nebude o možnost srovnání ochuzen.

Pravidlová morfologická disambiguace žádná trénovací data nepotřebuje, kombinační metody, které budou popsány ve čtvrté kapitole, také žádné vlastní trénování (tedy navíc k natrénování použitých statistických taggerů) nevyžadují, ovšem testování jak samotných pravidel, tak všech kombinačních experimentů samozřejmě probíhalo na týchž testovacích datech jako testování samostatných taggerů. Podrobnosti budou uvedeny spolu s výsledky v příslušné kapitole.

Pro testování syntaktického rozšíření pravidel, popsaného ve čtvrté kapitole, jsme využili data anotovaná na analytické rovině z PDT 1.0 [26]. Využití PDT 2.0 nebylo možné z důvodů, které nedokážeme ovlivnit (není možné přetrénovat Charniakův parser apod.)

Pro naše úlohy (tagging, parsing) a zvolená data (PDT) platí, že existuje právě jedna ruční anotace („správný výsledek“), tedy pro každý slovní tvar je přiřazena právě jedna dvojice morfologického tagu a lemmatu, resp. právě jeden otec v syntaktickém stromě. U metod, které vydávají obecně více výsledků (pravidlová disambiguace, některé mezistupně kombinovaných metod), měříme *precision*, *recall* a *F-measure* podle následujících vzorců (srov. např. [12]).

Nechť t je počet tokenů v testovacích datech, c počet všech výsledků vydaných zvolenou metodou (pro všechny tokeny dohromady), h počet tokenů, u nichž se správný výsledek objevil mezi vydanými výsledky. Potom jsou charakteristiky *precision* (p), *recall* (r) a *F-measure* (f) definovány následovně:

$$p = h/c \qquad r = h/t \qquad f = 2pr/(p+r).$$

Pokud evaluovaná metoda vydává vždy právě jeden výsledek, platí $p = r = f = h/t$ a tuto hodnotu nazýváme *accuracy*.

Kapitola 2

Valenční slovník deverbativních adjektiv

V této kapitole popisujeme proces automatického převodu valenčního [1]) slovníku sloves na valenční slovník jim odpovídajících deverbativních adjektiv.

Tento proces byl vyvinut pro účely disambiguačního projektu, je však na něm zcela nezávislý (program i data jsou uvolněny pod licencí GPL) a výsledný valenční slovník deverbativních adjektiv, který je ve stejném formátu jako původní slovník sloves, může vygenerovat a začlenit do svého projektu kdokoli, kdo má k dispozici původní slovník.

Jako zdrojový valenční slovník sloves jsme využili brněnský Brief [24], [10], neboť minimálně v době, kdy v disambiguačním projektu vznikla potřeba využití znalostí o valenci, to byl nejrozsáhlejší slovník povrchové valence, který byl pro češtinu k dispozici. (Ani nyní nám není známo, že by byl překonán.) Výsledný valenční slovník deverbativních adjektiv je taktéž dosud nejrozsáhlejším a nejpropracovanějším dílem svého druhu.

Implementace naší převodní procedury je samozřejmě přizpůsobena konkrétnímu zdrojovému slovníku, nicméně potřebná lingvistická fakta jsou zachycena zcela obecně a domníváme se, že případná adaptace na jiný slovník odlišného formátu by byla otázkou maximálně několika dní.

V následujících podkapitolách nejprve přiblížíme souvislost s disambiguačním projektem a poté podrobně popíšeme jednotlivé fáze převodu slovníku – odvozování adjektiv od sloves a převod valenčních rámců.

[1]) Zde i dále v textu máme na mysli pouze povrchovou valenci.

2.1 Potřeba povrchové valence v disambiguaci

S rozvojem disambiguačního systému začala vznikat pravidla, která neplatí pro celé třídy slov stejného slovního druhu, nýbrž závisejí na konkrétním obsazení, speciálně tedy na povrchové valenci sloves, adjektiv, případně i deverbativních substantiv. Například rozdíl mezi automatickou disambiguací slovního tvaru *Petra* ve větách *Petra měla bílé zuby.* a *Petra měla plné zuby.* je dán pouze informací o valenčním potenciálu slova *plný*. Pravidla využívají valenční informace jak pozitivně (aplikují se, pouze pokud určité slovo má danou vlastnost), tak negativně (je-li v konfiguraci pravidla slovo s určitou vlastností, pravidlo se neaplikuje). Pro více příkladů a podrobností viz [4].

Žádné pravidlo zatím neohledává celý valenční rámec (resp. všechny valenční rámce) slovesa či adjektiva, vždy jde jen o konkrétní vlastnost, tedy zda dané sloveso či adjektivum může mít dané doplnění, případně zda ho musí mít povinně (figuruje ve všech valenčních rámcích), či zda ho naopak nemůže mít nikdy (nefiguruje v žádném z rámců). Disambiguační systém tedy nepotřebuje být (a zatím také není) přímo propojen s valenčním slovníkem, stačí mu pro každou ověřovanou vlastnost seznam sloves (adjektiv), která onu vlastnost mají (např. slovesa s obligatorní akuzativní valencí, bez akuzativní valence, s fakultativní dativní valencí apod.). Tyto seznamy se v současné době automaticky generují ze slovníku, přímé zapojení slovníku do systému je samozřejmě do budoucna také možné.

Pro generování valenčních seznamů sloves využíváme slovník Brief, v případě adjektiv využíváme námi odvozený valenční slovník deverbativních adjektiv, navíc přidáváme i informace o valenci adjektiv, která deverbativní nejsou, a tudíž ve slovníku nefigurují (např. *plný* z příkladu uvedeného výše, srov. [14]). Z uvedených slovníků získáváme i některé informace, které s valencí přímo nesouvisejí, např. seznam reflexiv tantum.

2.2 Odvozování adjektiv od sloves

2.2.1 Teorie

Existuje více typů deverbativních adjektiv, netriviální valenční rámec nesou však jen tři z nich, proto se budeme nadále zabývat pouze jimi:

- adjektiva procesuální – „činná“ odvozená od nedokonavých sloves (*dělat* ⇒ *dělající*)
- adjektiva resultativní aktivní – „činná“ odvozená od dokonavých sloves (*udělat* ⇒ *udělavší*)[1])
- adjektiva trpná – odvozená od dokonavých i nedokonavých tranzitivních sloves (s předmětem) (*dělat* ⇒ *dělaný*, *udělat* ⇒ *udělaný*)

Každé sloveso vytváří alespoň jedno činné adjektivum, jehož typ závisí na vidu slovesa (viz výše), obouvidá slovesa vytvářejí oba typy činných adjektiv – procesuální i resultativní aktivní. Slovesa zařazená do některých specifických vzorů vytvářejí i více adjektiv jednoho typu (*vyjmout: vyjmuvší/vyňavší*).

Sloveso může a nemusí vytvářet trpné adjektivum, záleží na tom, zda toto sloveso má valeční rámec s předmětem (většinou akuzativním), ke kterému se odvozené adjektivum může vztahovat jako přívlastek. Situace ale není vždy takto jednoduchá, trpná adjektiva se mohou tvořit například i od některých reflexivních sloves (*narodit se – narozený*), proto se ve fázi odvozování omezíme na to, že trpná adjektiva vygenerujeme pro všechna slovesa a až později rozhodneme, která z nich do výsledného valenčního slovníku použijeme a která nikoli. Na místě trpných adjektiv někdy generujeme i přídavná jména odvozená od l-ových participií (typ *zemřelý*), neboť mají podobné valenční vlastnosti. Zdůvodnění tohoto kroku a podrobnosti následují v oddílu o převodu valenčních rámců.

2.2.2 Praxe

Při odvozování adjektiv od sloves je třeba znát vid slovesa a vzor pro odvození adjektiva. Vid napoví, který typ adjektiv sloveso generuje, vzor upřesní jejich tvar.

Potřebné informace o videch jsme získali ze dvou různých morfologických slovníků [5] [28] a v místech neshody těchto slovníků

[1]) V korpusu a starší literatuře se vyskytuje též typ adjektiva resultativního aktivního odvozeného od nedokonavého slovesa (*byvší* . . .), tento typ je ovšem vskutku okrajový a ani v odborné literatuře jsme o něm nenalezli zmínku.

rozhodli ručně (na základě obvyklých testů, např. tvar budoucího času). Pokoušeli jsme se i spojovat slovesa automaticky do vidových dvojic (či obecně N-tic – iterativní koncovky typu *-ávávat"*) za účelem vzájemného doplnění a ověření valenčních rámců v celé N-tici, ovšem výsledky nebyly valné, a tak jsme od tohoto nápadu upustili.

Vzory pro odvozování deverbativních adjektiv od sloves se dosud nikdo systematicky nezabýval, proto jsme vypracovali jejich podrobnou klasifikaci a do vzniklých 87 tříd poloautomaticky přiřadili všech cca 15000 sloves obsažených v Briefu [1]). Máme ještě rezervu čítající dalších téměř 18000 sloves (figurujících v morfologickém slovníku, nikoli však v Briefu), která může být využita v budoucnu při rozšíření Briefu nebo v případě adaptace na jiný slovník s odlišným obsahem.

Jako základ jsme využili vzory pro časování z morfologického slovníku [5], ty jsme pak pro potřeby odvozování adjektiv dále zjemňovali a větvili. Vzor pro odvození deverbativních adjektiv totiž nelze jednoznačně určit ze vzoru pro časování slovesa, neboť zatímco procesuální a resultativní aktivní adjektiva se chovají konzistentně a vycházejí z tvarů přechodníků, adjektiva trpná (příp. l-ová participia s valencí) se mohou pro jeden časovací vzor tvořit různými způsoby. Následují dva příklady (jsou velmi podobné, leč oba příliš půvabné na to, abychom některý z nich vynechali).

1. Sloveso *sladit* má dva vidově odlišné významy, nedokonavý např. *sladit čaj*, dokonavý *sladit barvy*. Ačkoli v časování není mezi těmito významy rozdíl, nedokonavé sloveso generuje trpné adjektivum *slazený*, zatímco dokonavé *sladěný*.

2. Slovesa *naladit*, *vyudit*, *hodit* se také časují všechna stejně a všechna jsou dokonavá, ovšem *naladit* generuje trpné adjektivum *naladěný*, *vyudit* generuje *vyuděný* i *vyuzený* a *hodit* pouze *hozený*, tato trojice sloves se tedy rozpadá do tří různých vzorů pro generování adjektiv.

V případě nedostatku času či sil na manuální třídění bychom se ovšem pro většinu úloh mohli smířit s přegenerováním, tedy v tomto případě s tvořením trpných adjektiv se zakončeními *-děný* i *-zený* pro všechna slovesa se zakončením *-dit*, analogicky by se daly slít i další skupiny vzorů (např. *zit* a *zzit* nebo *eeci*, *eekci* a *eeci2*, viz seznam v následujícím oddílu).

[1]) Nepatrné množství „mimořádně nepravidelných" sloves nebylo možné přiřadit žádnému vzoru a jsou vedena jako výjimky.

2.2.3 Seznam derivačních vzorů

V této části uvedeme všechny derivační vzory pro odvozování adjektiv od sloves spolu s příklady, procentuálním zastoupením mezi slovesy v Briefu a podílem dokonavých (D), nedokonavých (N) a obouvidých (O) [1]) sloves mezi zástupci daného vzoru. (V celém Briefu je cca 53 % dokonavých, 43 % nedokonavých a 4 % obouvidých či homonymních sloves.) U každého vzoru je v prvním řádku uveden i jeho název, který má ryze technický účel a neobsahuje žádnou lingvistickou informaci.

Každý derivační vzor se skládá ze čtyř částí – zakončení slovesa určeného k utnutí [2]) a zakončení všech tří typů deverbativních adjektiv, která se k torzu slovesa následně přidají. Z procesuálního a resultativního aktivního adjektiva se na základě vidu vybere jedno či obě, trpné adjektivum se, jak již bylo řečeno, vygeneruje vždy a následně se při konfrontaci s valenčním rámcem slovesa buď ponechá, nebo odstraní. Jedno ze zakončení činných adjektiv může (ale nemusí!) chybět, pokud se vzor týká buď pouze dokonavých, nebo pouze nedokonavých sloves. Některá zakončení mohou být zdvojená, v tom případě se generuje více adjektiv stejného typu. Tato adjektiva jsou synonymní a budou mít stejný valenční rámec.

Sedm nejčastějších vzorů popisujících 88 % sloves:

t: -t -jící -vší -ný

29.79% podíl (43.82 % D, 54.70 % N, 1.46 % O)
Příklady: *chytat* (*chytající*, –, *chytaný*), *dodělat* (–, *dodělavší*, *dodělaný*)

o: -ovat -ující -ovavší -ovaný

28.19% podíl (21.48 % D, 66.52 % N, 11.98 % O)
Příklady: *asociovat* (*asociující*, *asociovavší*, *asociovaný*)

i: -it -ící -ivší -ený

14.14% podíl (76.83 % D, 22.92 % N, 0.24 % O)
Příklady: *balit* (*balící*, –, *balený*), *dokončit* (–, *dokončivší*, *dokončený*)

n: -out -oucí -uvší -utý

6.97% podíl (91.04 % D, 8.71 % N, 0.24 % O)
Příklady: *drhnout* (*drhnoucí*, –, *drhnutý*), *klepnout* (–, *klepnuvší*, *klepnutý*)

[1]) Zde mezi ně počítáme i homonymní slovesa s jedním významem dokonavým a jedním nedokonavým.
[2]) Toto zakončení tedy může být jak utnuté, tak uťaté.

w: -it -ící -ivší -ěný

5.93% podíl (74.78 % D, 25.07 % N, 0.14 % O)
Příklady: *leštit* (*leštící*, –, *leštěný*), *nahromadit* (–, *nahromadivší*, *nahromaděný*)

c: -tit -tící -tivší -cený

1.44% podíl (82.24 % D, 17.75 % N)
Příklady: *fotit* (*fotící*, –, *focený*), *chytit* (*-*, *chytivší*, *chycený*)

z: -dit -dící -divší -zený

1.40% podíl (88.41 % D, 10.97 % N, 0.6 % O)
Příklady: *soudit* (*soudící*, –, *souzený*), *nahradit* (–, *nahradivší*, *nahrazený*)

Ostatní vzory:

lt: -t 0 -vší -lý

0.44% podíl (100.00 % D)
Příklady: *zpitomět* (–, *zpitoměvší*, *zpitomělý*)

ett: -t 0 -vší -tý

0.01% podíl (100.00 % D)
Příklady: *naset* (–, *nasevší*, *nasetý*)

jet: -t -doucí -dší -tý

0.18% podíl (95.24 % D, 4.76 % N)
Příklady: *jet* (*jedoucí*, –, –), *zajet* (–, *zajedší*, *zajetý*)

aat: -át -oucí 0 -aný

0.05% podíl (100 % N)
Příklady: *zvát* (*zvoucí*, –, *zvaný*)

aajt: -át -ající -avší -aný

0.31% podíl (58.33 % D, 41.67 % N)
Příklady: *hrát* (*hrající*, –, *hraný*), *rozehrát* (–, *rozehravší*, *rozhraný*)

aawt: -át -ějící 0 -átý

1 ks (N)
Příklady: *smát* (*smějící*, –, –)

aatt: -át -ející -avší -átý

0.32% podíl (92.10 % D, 7.89 % N)
Příklady: *hřát* (*hřející*, –, *hřátý*), *rozehřát* (–, *rozehřavší*, *rozehřátý*)

aaat: -át -ající -avší -átý

0.11% podíl 85.71 % D, 14.28 % N
Příklady: *sát* (*sající*, –, *sátý*), *dopřát* (–, *dopřavší*, *dopřátý*)

iet: -et -ící -ivší -ený

0.45% podíl (3.77 % D, 96.22 % N)
Příklady: *držet* (*držící*, –, *držený*), *zabydlet* (–, *zabydlivší*, *zabydlený*)

iwt: -ět -ící -ivší -ěný

0.40% podíl (4.25 % D, 95.74 % N)
Příklady: *trpět* (*trpící*, –, *trpěný*), *vyhovět* (–, *vyhovivší*, –)

aint: -ít -ající 0 0

1 ks (N)
Příklady: *mít* (*mající*, –, –)

eent: -ít -ející -evší -ený

0.45% podíl (96.22 % D, 3.77 % N)
Příklady: *klít* (*klející*, –, –), *otevřít* (–, *otevřevší*, *otevřený*)

eett: -ít -ející -evší -etý

0.19% podíl (91.30 % D, 8.69 % N)
Příklady: *plít* (*plející*, –, *pletý*), *zasít* (–, *zasevší*, *zasetý*)

wlnt: -ít -ějící -ěvší -ělý

0.05% podíl (71.42 % D, 28.57 % N)
Příklady: *skvít* (*skvějící*, –, –), *odeznít* (–, *odezněvší*, *odeznělý*)

wwnt: -ít -ějící -ěvší -ěný

0.10% podíl (75 % D, 16.66 % N, 8.33 % O)
Příklady: *dít (se)* (*dějící*, –, –), *přiodít* (–, *přioděvší*, *přioděný*)

ient: -ít -ící -ivší -ený

0.12% podíl (53.33 % D, 46.66 % N)
Příklady: *třít* (*třící*, –, *třený*), *pohřbít* (–, *pohřbivší*, *pohřbený*)

iwnt: -ít -ící -ivší -ěný

0.31% podíl (59.45 % D, 40.54 % N)
Příklady: *mstít* (*mstící*, –, *mstěný*), *zneuctít* (–, *zneuctivší*, *zneuctěný*)

iitt: -ít -ující -ivší -itý

0.68% podíl (88.75 % D, 11.25 % N)
Příklady: *bít* (*bijící*, –, *bitý*), *využít* (–, *využivší*, *využitý*)

att: -ít 0 -avší -atý

0.05% podíl (100.00 % D)
Příklady: *počít* (–, *počavší*, *počatý*)

elt: -ít 0 -evší -elý

0.06% podíl (100.00 % D)
Příklady: *vymřít* (–, *vymřevší*, *vymřelý*)

ilt: -ít 0 -ivší -ilý

0.05% podíl (100.00 % D)
Příklady: *opít* (–, *opivší*, *opilý*)

jit: -jít -jdoucí -šedší -šlý

0.16% podíl (94.73 % D, 5.26 % N)
Příklady: *jít* (*jdoucí*, –, –), *ujít* (–, *ušedší*, *ušlý*)

tiit: -tít 0 -ťavší -ťatý

0.05% podíl (100.00 % D)
Příklady: *stít* (–, *sťavší*, *sťatý*)

yt: -ýt -yjící -yvší -ytý

0.33% podíl (84.61 % D, 15.38 % N)
Příklady: *krýt* (*kryjící*, –, *krytý*), *rozrýt* (–, *rozryvší*, *rozrytý*)

lyt: -ýt 0 -yvší -ylý

1 ks (D)
Příklady: *zbýt* (–, *zbyvší*, *zbylý*)

aast: -ást -asoucí -ásší -asený

0.04% podíl (80.00 % D, 20.00 % N)
Příklady: *pást* (*pasoucí*, –, –), *vypást* (–, *vypásší*, *vypasený*)

dast: -ást -adoucí -adší -adený

0.11% podíl (85.71 % D, 14.28 % N)
Příklady: *klást* (*kladoucí*, –, *kladený*), *rozkrást* (–, *rozkradší*, *rozkradený*)

tast: -ást -atoucí -átší -atený

0.02% podíl (66.66 % D, 33.33 % N)
Příklady: *mást* (*matoucí*, –, *matený*), *zmást* (–, *zmátší*, *zmatený*)

eest: -ést -esoucí -esší -esený

0.17% podíl (95.00 % D, 5.00 % N)
Příklady: *nést* (*nesoucí*, –, *nesený*), *přednést* (–, *přednesší*, *přednesený*)

aest: -ást -esoucí -ásší -esený

0.10% podíl (91.66 % D, 8.33 % N)
Příklady: *třást* (*třesoucí*, –, *třesený*), *setřást* (–, *setřásší*, *setřesený*)

dest: -ést -edoucí -edší -edený

0.12% podíl (93.33 % D, 6.66 % N)
Příklady: *vést* (*vedoucí*, –, *vedený*), *podvést* (–, *podvedší*, *podvedený*)

test: -ést -etoucí -etší -etený

0.19% podíl (86.95 % D, 13.04 % N)
Příklady: *plést* (*pletoucí*, –, *pletený*), *zamést* (–, *zametší*, *zametený*)

ltest: -ést 0 -etší -etlý

0.05% podíl (100.00 % D)
Příklady: *rozkvést* (–, *rozkvetší*, *rozkvetlý*)

ezt: -ézt -ezoucí -ezší -ezený

0.21% podíl (92.00 % D, 8.00 % N)
Příklady: *vézt* (*vezoucí*, –, *vezený*), *přivézt* (–, *přivezší*, *přivezený*)

lezt: -ézt -ezoucí -ezší -ezlý

0.01% podíl (100.00 % D)
Příklady: *zalézt* (–, *zalezší*, *zalezlý*)

dist: -íst -edoucí -edší -edený

0.15% podíl (88.88 % D, 11.11 % N)
Příklady: *přist* (*předoucí*, –, *předený*), *dojíst* (–, *dojedší*, *dojedený*)

tist: -íst -toucí -etší -tený

0.17% podíl (95.23 % D, 4.76 % N)
Příklady: *číst* (*čtoucí*, –, *čtený*), *připočíst* (–, *připočetší*, *připočtený*)

wist: -íst -ětoucí -ětší -ětený

1 ks (N)
Příklady: *hnís*t (*hnětoucí*, –, *hnětený*)

wdist: -íst 0 -ědší -ědený

1 ks (D)
Příklady: *sníst* (–, *snědší*, *snědený*)

ust: -ůst -ostoucí -ostší -ostlý

0.13% podíl (93.75 % D, 6.25 % N)
Příklady: *růst* (*rostoucí*, –, –), *přerůst* (–, *přerostší*, *přerostlý*)

yzt: -ýzt -yzoucí 0 -yzený

1 ks (N)
Příklady: *hrýzt* (*hryzoucí*, –, *hryzený*)

lyzt: -ýzt 0 -yzší -yzlý

0.02% podíl (100 % N)
Příklady: *prohrýzt* (–, *prohryzší*, *prohryzlý*)

zwt: -dět -dící -divší -zený

0.10% podíl (91.66 % D, 8.33 % N)
Příklady: *sedět* (*sedící*, –, –), *odpovědět* (–, *odpovědivší*, *odpovězený*)

wz: -dit -dící -divší -děný/-zený

0.01% podíl (50 % N, 50 % O)
Příklady: *sladit (N+D)* (*sladící (N)*, *sladivší (D)*, *slazený (N)*, *sladěný* (D)), *udit* (*udící*, -, *uděný/uzený*)

sit: -sit -sící -sivší -šený

0.63% podíl (85.13 % D, 14.86 % N)
Příklady: *hlásit* (*hlásící*, –, *hlášený*), *nabrousit* (–, *nabrousivší*, *nabroušený*)

zit: -zit -zící -zivší -žený

0.24% podíl (89.65 % D, 10.34 % N)
Příklady: *kazit* (*kazící*, –, *kažený*), *přerazit* (–, *přerazivší*, *přeražený*)

zzit: -zit -zící -zivší -žený/-zený

0.02% podíl (100.00 % D)
Příklady: *zmrazit* (–, *zmrazivší*, *zmrazený/zmražený*)

eeci: -éci -ekoucí -ekší -ečený/-eklý

0.04% podíl (100 % D)
Příklady: *napéci* (–, *napekší*, *napečený/napeklý*)

eekci: -éci -ekoucí -ekší -eklý

0.13% podíl (93.33 % D, 6.67 % N)
Příklady: *téci* (*tekoucí*, –, –), *odtéci* (–, *odtekší*, *odteklý*)

eeci2: -éci -ečící -ekší -ečený

0.16% podíl (89.47 % D, 10.53 % N)
Příklady: *vléci* (*vlekoucí*, –, *vlečený*), *obléci* (–, *oblekší*, *oblečený*)

eect: -éct -ekoucí -ekší -ečený

0.06% podíl (87.50 % D, 12.50 % N)
Příklady: *přivléct* (–, *přivlekší*, *přivlečený*)

iici: -íci -ekoucí -ekší -ečený

0.09% podíl (90.90 % D, 9.09 % N)
Příklady: *dořící* (–, *dořekší*, *dořečený*)

iict: -íct -ekoucí -ekší -ečený

0.03% podíl (75.00 % D, 25.00 % N)
Příklady: *dořící* (–, *dořekší*, *dořečený*)

oci: -ci -houcí -hší -žený

0.11% podíl (92.85 % D, 7.14 % N)
Příklady: *moci* (*mohoucí*, –, –), *přemoci* (–, *přemohší*, *přemožený*)

oct: -ct -houcí 0 -žený

1 ks (N)
Příklady: *moct* (*mohoucí*, –, –)

ouci: -ouci -ukoucí -oukší/-ukší -učený

(Tento vzor, stejně jako následující, je poněkud nejistý, neboť se nám (ani osobám povolanějším) nepodařilo najít žádné doklady pro ani jednu z variant, proto raději vytváříme obě, neboť není jasné, která je správná.)

0.11% podíl (92.30 % D, 7.69 % N)
Příklady: *tlouci* (*tlukoucí*, –, –), *zatlouci* (–, *zatloukší/zatlukší*, *zatlučený*)

ouct: -ouct -ukoucí -oukší/-ukší -učený

0.01% podíl (100.00 % D)
Příklady: *přitlouct* (–, *přitloukší/přitlukší*, *přitlučený*)

out: -out -ující -uvší -utý

0.24% podíl (86.20 % D, 13.79 % N)
Příklady: *kout* (*kující*, –, *kutý*), *proplout* (–, *propluvší*, *proplutý*)

lout: -out 0 -uvší -ulý

0.07% podíl (100.00 % D)
Příklady: *zahynout* (–, *zahynuvší*, *zahynulý*)

jmout-ja: -mout 0 -muvší/-avší -mutý/-atý

0.06% podíl (100.00 % D)
Příklady: *pronajmout* (–, *pronajmuvší/pronajavší*, *pronajmutý/pronajatý*)

jmout-na: -jmout 0 -jmuvší/-ňavší -jmutý/-ňatý

0.01% podíl (100.00 % D)
Příklady: *vyjmout* (–, *vyjmuvší/vyňavší*, *vyjmutý/vyňatý*)

jmout-ena: -ejmout 0 -ejmuvší/-ňavší -ejmutý/-ňatý

0.02% podíl (100.00 % D)
Příklady: *odejmout* (–, *odejmuvší/odňavší*, *odejmutý/odňatý*)

dnout: -nout -noucí -nuvší -ený

0.03% podíl (100.00 % D)
Příklady: *přepadnout* (–, *přepadnuvší*, *přepadený*)

ln: -nout -noucí -nuvší -nutý/-lý

0.56% podíl (100.00 % D)
Příklady: *sežehnout* (–, *sežehnuvší*, *sežehnutý/sežehlý*)

l: -nout -noucí -nuvší -lý

0.51% podíl (98.33 % D, 1.66 % O)
Příklady: *padnout (N+D)* (*padnoucí (N)*, *padnuvší (D)*, *padlý (D)*)

nout-ja: -nout 0 -nuvší/-javší -nutý/-jatý

0.10% podíl (100.00 % D)
Příklady: *napnout* (–, *napnuvší/napjavší*, *napnutý/napjatý*)

nout-ze: -hnout 0 -hnuvší -hnutý/-žený

0.41% podíl (100.00 % D)
Příklady: *přestřihnout* (–, *přestřihnuvší*, *přestřihnutý/přestřižený*)

nout-aze: -áhnout -áhnoucí -áhnuvší -áhnutý/-ažený

0.25% podíl (96.66 % D, 3.33 % N)
Příklady: *táhnout* (*táhnoucí*, –, *táhnutý/tažený*), *dosáhnout* (–, *dosáhnuvší*, *dosáhnutý/dosažený*)

nout-t: -tnout 0 -tnuvší/-ťavší -tnutý/-ťatý

0.03% podíl (100.00 % D)
Příklady: *utnout* (–, *utnuvší/uťavší*, *utnutý/uťatý*)

nout-et: -etnout 0 -etnuvší/-ťavší -etnutý/-ťatý

0.01% podíl (100.00 % D)
Příklady: *rozetnout* (–, *rozetnuvší/rozťavší*, *rozetnutý/rozťatý*)

nout-ce: -knout 0 -knuvší -knutý/-čený

0.09% podíl (100.00 % D)
Příklady: *odemknout* (–, *odemknuvší*, *odemknutý/odemčený*)

nout-ece: -éknout 0 -éknuvší -éknutý/-ečený

0.06% podíl (100.00 % D)
Příklady: *svléknout* (–, *svléknuvší*, *svléknutý/svlečený*)

nout-st: -sknout -sknoucí -sknuvší -sknutý/-štěný

0.08% podíl (90.00 % D, 10.00 % N)
Příklady: *tisknout* (*tisknoucí*, –, *tisknutý/tištěný*), *vytisknout* (–, *vytisknuvší*, *vytisknutý/vytištěný*)

rat: -rát -eroucí 0 -raný

0.04% podíl (100.00 % N)
Příklady: *prát* (*peroucí*, –, *praný*)

rat2: -rat 0 -ravší -raný

0.01% podíl (100.00 % N)
Příklady: *nežrat* (*nežeroucí*, –, *nežraný*)

slet: -slet -slící -slevší -šlený

0.07% podíl (88.88 % D, 11.11 % N)
Příklady: *myslet* (*myslící*, –, *myšlený*), *rozmyslet* (–, *rozmyslevší*, *rozmyšlený*)

slit: -slit -slící -slivší -šlený

0.10% podíl (91.66 % D, 8.33 % N)
Příklady: *myslit* (*myslící*, –, *myšlený*), *vymyslit* (–, *vymyslivší*, *vymyšlený*)

stit: -stit -stící -stivší -štěný

0.46% podíl (90.74 % D, 9.25 % N)
Příklady: *mastit* (*mastící*, –, *maštěný*), *pohostit* (–, *pohostivší*, *pohoštěný*)

zdit: -zdit -zdící -zdivší -žděný

0.10% podíl (91.66 % D, 8.33 % N)
Příklady: *jezdit* (*jezdící*, –, *ježděný*), *rozjezdit* (–, *rozjezdivší*, *rozježděný*)

zzdit: -zdit -zdící -zdivší -žděný/-zděný

0.05% podíl (83.33 % D, 16.66 % N)
Příklady: *brzdit* (*brzdící*, –, *brzděný/bržděný*), *zabrzdit* (–, *zabrzdivší*, *zabrzděný/zabržděný*)

2.3 Převod valenčních rámců

2.3.1 Základní algoritmus

Převod valenčních rámců je dobře definovatelný. „Činné“ typy adjektiv (procesuální a resultativní aktivní) přebírají valenční rámce sloves kompletně a beze změny[1]), zachovají i případnou reflexivitu. Trpná adjektiva se ve většině případů chovají tak, že vezmou v úvahu pouze nereflexivní a *si*-reflexivní variantu slovesa (tedy nikoli případnou variantu se *se*) a z jejich valenční informace převezmou všechny rámce obsahující akuzativ, kde však onen akuzativ vynechají. Vzniklé adjektivum nebude reflexivní.

Jako první příklad poslouží obouvidé sloveso *dokumentovat*. Jeho valenční rámce ve formátu Brief vypadají takto:

```
dokumentovat <v>hTc4,hTc4-hTc6r{na},hTc4-hTc7
```

a v okem čitelném (verbose) formátu takto:

```
dokumentovat
= co
= co & na čem
= co & čím
```

Valenční rámce odvozených adjektiv potom vypadají následovně:

```
dokumentující <v>hTc4,hTc4-hTc6r{na},hTc4-hTc7
dokumentovavší <v>hTc4,hTc4-hTc6r{na},hTc4-hTc7
dokumentovaný <v>,hTc6r{na},hTc7

dokumentující
= co
= co & na čem
= co & čím
dokumentovavší
= co
= co & na čem
= co & čím
dokumentovaný
= ; prázdná valence
= na čem
= čím
```

[1]) Podmět v rámcích zdrojového slovníku nefiguruje.

Druhý příklad demonstruje zacházení s reflexivitou. Sloveso *doplňovat* existuje ve všech třech variantách:

```
doplňovat <v>hPTc4,hPTc4-hTc7,hTc4,hTc4-hTc2r{do}
doplňovat se <v>hPTc7r{s},hTc7
doplňovat si <v>hTc4,hTc4-hTc7

doplňovat
= koho|co
= koho|co & čím
= co
= co & do čeho
doplňovat se
= s kým|čím
= čím
doplňovat si
= co
= co & čím
```

Procesuální adjektivum jednoduše převezme všechny varianty reflexivity i jejich valenční rámce:

```
doplňující <v>hPTc4,hPTc4-hTc7,hTc4,hTc4-hTc2r{do}
doplňující se <v>hPTc7r{s},hTc7
doplňující si <v>hTc4,hTc4-hTc7

doplňující
= koho|co
= koho|co & čím
= co
= co & do čeho
doplňující se
= s kým|čím
= čím
doplňující si
= co
= co & čím
```

Trpné adjektivum výše popsaným způsobem převezme a upraví pouze rámce varianty nereflexivní a varianty se *si*.

```
doplňovaný <v>,hTc7,hTc2r{do}

doplňovaný
= ; prázdná valence
```

```
= čím
= do čeho
```

Zde z varianty se *se* ani není co přebírat, ale i kdyby bylo, ignorujeme to, neboť od těchto variant sloves se trpná adjektiva obecně netvoří (*dozvědět se* $\not\Rightarrow$ **dozvěděný*), o výjimkách se zmíníme v následujícím oddílu.

2.3.2 Výjimky

Základním algoritmem lze odvodit a opatřit valenční informací většinu trpných adjektiv, nikoli však všechna. Existují totiž i trpná adjektiva odvozená od (jen některých!) sloves s předmětem v jiném pádě než akuzativu, a dále trpná adjektiva a l-ová participia, která popisují nikoli předmět, nýbrž podmět děje, a tvoří se od (opět jen některých) sloves bez předmětu, a to jak reflexivních, tak nereflexivních.

Objevit tyto výjimky se nám podařilo zejména díky konfrontaci raných verzí generovaného slovníku adjektiv s vyhledávkami v Českém národním korpusu [1], které v době před zapojením slovníku „nahrubo“ aproximovaly valenční seznamy (podrobnosti viz [4]). Adjektiva, která se v korpusových seznamech vyskytovala, ale ve slovníku nikoli, byla právě ta, která jsou od svých základových sloves odvozena méně běžnými způsoby.

Příklad na převod rámců slovesa s předmětem v jiném pádě (zde v dativu):

```
polichotit <v>hPTc3,hPTc3-hTc6r{v},hPTc3-hTc7,hTc6r{v},hTc7
polichotivší <v>hPTc3,hPTc3-hTc6r{v},hPTc3-hTc7,hTc6r{v},hTc7
polichocený <v>,hTc6r{v},hTc7

polichotit
= komu|čemu
= komu|čemu & v čem
= komu|čemu & čím
= v čem
= čím

polichotivší
= komu|čemu
= komu|čemu & v čem
= komu|čemu & čím
= v čem
= čím
```

```
polichocený
= ; prázdná valence
= v čem
= čím
```

Převodní algoritmus v tomto případě funguje téměř stejně jako u běžných sloves, pouze akuzativ je nahrazen dativem – trpné adjektivum tedy převezme všechny rámce obsahující dativ a z nich tento dativ vynechá.

Vypečenějším typem výjimky je případ, kdy trpné adjektivum nepopisuje předmět, nýbrž podmět slovesa, začasté reflexivního (*narodit se* ⇒ *narozený*). Zde trpné adjektivum přibírá stejně jako činné (kterému se významově velmi blíží) všechny valenční rámce bez jakýchkoli úprav, ztrácí pouze reflexivitu.

```
narodit se <v>,hPTc7,hPc3
narodivší se <v>,hPTc7,hPc3
narozený <v>,hPTc7,hPc3

narodit se
= ; prázdná valence
= kým|čím
= komu
narodivší se
= ; prázdná valence
= kým|čím
= komu
narozený
= ; prázdná valence
= kým|čím
= komu
```

Podobně se chovají např. dvojice *vyrůst* ⇒ *vyrostlý*, *zmrznout* ⇒ *zmrzlý*, *vydařit se* ⇒ *vydařený*. Zde mezi deverbativní adjektiva zařazujeme i adjektiva odvozená z l-ových participií, neboť stejným způsobem přebírají valenční chování slovesa (*vyrostlý z dětských kalhot*, *zmrzlý na kost* apod.) To ovšem neplatí pro všechna adjektiva odvozená z l-ových participií – v některých případech dochází k významovému posunu a ztrátě valenčního chování (např. *vleklý*, *přepadlý*), takováto adjektiva jsme se snažili vynechat.

Žádnou z těchto výjimek bohužel nelze odhalit automaticky pouze na základě tvaru, vzoru či valence slovesa, vše vychází z konfrontace s daty či z lingvistické intuice a muselo být ručně poznamenáno ke slovesům, výjimek všeho druhu jsme dosud odhalili asi 400.

Příkladem nepredikovatelnosti chování sloves budiž dvojice synonymních sloves se stejným vzorem a stejnými valenčními rámci *dotázat se* a *zeptat se*. Zatímco *dotázat se* tvoří trpné adjektivum popisující genitivní předmět (*dotázaný*), žádné *zeptaný* bohužel neexistuje.

2.4 Shrnutí

Závěrem pouze dodáme, že výsledný valenční slovník obsahuje cca 25000 deverbativních adjektiv, vzhledem k pečlivému rozdělení vzorů pravděpodobně s žádným nebo zcela minimálním přegenerováním.

Správnost dat nelze ve větším měřítku ověřit vzhledem k neexistenci dat referenčních, ovšem během více než ročního intenzivního používání v disambiguačním projektu jsme dosud nenarazili na žádný problém, který by neměl svůj původ již ve zdrojovém valenčním slovníku sloves, proto se dá předpokládat, že převodní procedura sama o sobě žádné chyby nevnáší. Problém může přinést pouze nesprávné přiřazení vzorů ke slovesům, které vzhledem k značnému podílu manuální práce nemůžeme vyloučit.

Vytvořili jsme obecně využitelný zdroj dat, který může do svého projektu snadno zapojit každý, kdo využíval původní slovník sloves Brief (a samozřejmě i kdokoli jiný, má-li původní slovník k dispozici).

Kapitola 3

Kombinované metody značkování

Kombinované metody (nejen pro značkování) odedávna vzrušují mysl všech osvícenějších počítačových lingvistů. Základní myšlenka je jednoduchá: máme-li více různých metod řešení jedné úlohy, pravděpodobně se pro každou z nich najde nějaká oblast (specifický typ dat, konkrétní složka výstupu, jazykový jev...), ve které vyniká nad ostatními. Je zde tedy potenciál, jak zkombinováním více metod dosáhnout výsledku lepšího, než jaký má každá z nich samostatně. Pravděpodobnost úspěchu je přirozeně tím vyšší, čím jsou metody rozdílnější, neboť u podobných metod natrénovaných navíc na stejných datech se dá očekávat (a toto očekávání bývá také většinou splněno), že budou mít problémy na stejných místech.

Zaměřme se nyní konkrétně na kombinované morfologické značkování, tedy na úlohu, jak za použití libovolných prostředků dosáhnout toho, abychom z morfologické nabídky pro každé slovo co nejlépe vybrali jedinou v daném kontextu správnou značku (případy skutečné homonymie zanedbejme, neboť většina metod pro ně nemá podporu).

Již s několika statistickými taggery, kde výstupem každého z nich je pro každé slovo právě jedna značka, se dají provádět experimenty spočívající převážně v jednodušším či složitějším výběru (hlasování) mezi nezávisle dosaženými dílčími výsledky, viz např. [9], [17]. Ovšem ten pravý kombinační potenciál se otevírá teprve při zahrnutí metody pracující na zcela jiném principu, v tomto případě pravidly řízené morfologické disambiguace. Jak už jsme několikrát uvedli, tato metoda v obecném případě nevybírá právě jednu značku (ačkoli v konkrétních případech může!), nýbrž pouze omezuje obdrženou morfologickou nabídku na základě lingvistické přípustnosti jednotlivých kombinací značek v dané větě. Přitom je jedno, zda se jedná o plnou nabídku přímo z morfologické analýzy, nebo o nabídku jakkoli předem modifikovanou (tedy zřejmě redukovanou). Speciálním případem je situace, kdy každé slovo již má přiřazenu právě jednu značku (taggerem, anotátorem) a pravidla se použijí pro dodatečnou kontrolu správnosti tohoto přiřazení (buď onu jedinou značku ponechají, nebo nezbude žádná).

Tato kapitola shrnuje nápady a experimenty, které byly provedeny s využitím tří různých statistických taggerů (popsaných v úvodní

kapitole) a disambiguačních pravidel. Některé z metod byly vyvinuty a vyzkoušeny již dříve a my jsme je pouze zopakovali s referenčními verzemi nástrojů a na referenčních datech, nejúspěšnější metody jsou však naše vlastní a zcela nové.

Jako referenční byla použita data z PDT 2.0 [7], veškeré experimenty byly vyhodnoceny na d-test datech (201 651 tokenů) a nejúspěšnější z nich taktéž na e-test (219 765 tokenů). Trénovací data pro statistické taggery (1 539 241 tokenů) i testovací data byla označkována morfologickou analýzou ve verzi CZ060406a (tedy z dubna 2006), disambiguační pravidla byla použita ve verzi ze září 2006.

V disambiguačním systému se v té době nacházelo 2234 pravidel rozdělených do tří skupin – *root*, *heuristika1* a *heuristika2*. *Root* jsou pravidla lingvisticky bezpečná, heuristické skupiny je mohou doplňovat (nepoužívají se samostatně, byt' technicky to možné je) a jsou poněkud „odvážnější" (mají tedy větší disambiguační výkon s větším rizikem chyby). Zařazování pravidel do těchto skupin probíhá na základě intuice jejich tvůrců – plně automatická evaluace chybovosti jednotlivých pravidel je prakticky neproveditelná, neboť většina pravidel se uplatňuje pouze ve spolupráci s dalšími. K experimentům zde dokumentovaným jsme nepoužívali jiné podmnožiny pravidel než *root* a *root* + *heuristika1* (dále jen *disheu1*).

Úspěšnost morfologické analýzy, pravidel a jednotlivých statistických taggerů na d-test datech je následující:

	precision	recall	F-measure
Morfologie	25.72 %	99.40 %	40.87 %
Pravidla *root*	58.76 %	98.90 %	73.72 %
Pravidla *disheu1*	67.36 %	98.24 %	**79.92 %**

Tagger	accuracy
Feature-based (dále jen *a*)	94.27 %
HMM (dále jen *b*)	95.13 %
Morče (dále jen *m*)	**95.43 %**

Značný náskok taggeru *m* oproti ostatním uvedeným je důvodem, proč jsme veškeré experimenty zahrnující pouze jeden statistický tagger prováděli primárně s ním. Významnější experimenty jsme však pro úplnost zkoušeli i s ostatními taggery a skutečně se nikdy nestalo, že by výsledek téhož experimentu s taggerem *a* či *b* byl lepší než výsledek s taggerem *m*.

3.1 Sériová kombinace pravidla – tagger

První, co nás napadne, když dostaneme k dispozici zmíněné nástroje, je zredukovat nejprve nabídku z morfologické analýzy pravidly a poté spustit libovolný statistický tagger. Starší experimenty (viz např. [6] popisující sériovou kombinaci pravidel s taggerem *b*) byly prováděny s nepříliš velkou množinou pravidel ručně přepsanou do C++. V současné době, kdy existuje kompilátor jazyka LanGR [16], lze automaticky experimentovat s libovolnou podmnožinou pravidel v systému.

Experiment jsme zopakovali pro všechny tři statistické taggery v kombinaci se sadou pravidel *root* i *disheu1*. Důležitá je otázka, zda taggery pro tento účel přetrénovávat, tedy zda i morfologickou nabídku v trénovacích datech zredukovat týmiž pravidly a natrénovat na nich speciální verze taggerů, či nikoli. Přetrénování bohužel musí provést sami autoři taggerů, my tuto možnost nemáme, čímž se úkol poněkud komplikuje.

Taggeru *b* se tato otázka netýká, protože při trénování využívá pouze ruční anotace, velikost morfologické nabídky nemá na jeho výkon vliv. U taggeru *m* jsme v raných fází kombinování zkoušeli ve spolupráci s autorem taggeru všechny experimenty s přetrénováním i bez něj. Výsledky s přetrénováním byly vždy stejné nebo horší, proto jsme je poté již vynechali. U taggeru *a* je nejpravděpodobnější, že by mu přetrénování mohlo pomoci, jde však o akt příliš náročný (zejména časově) a vzhledem k celkovému zaostávání tohoto taggeru za ostatními se nedají očekávat žádné průlomové výsledky, proto jsme se neodvážili autora taggeru s tímto požadavkem obtěžovat. Z uvedených důvodů jsou tedy všechny výsledky uvedeny ve variantách bez přetrénování.

(Původně jsme se domnívali, že tuto otázku bude treba řešit u všech kombinovaných metod, později se však ukázalo, že se vlastně týká pouze sériové kombinace jednoho taggeru s pravidly. Ve všech ostatních kombinovaných metodách se používají minimálně dva taggery za sebou, a tudíž by trénovací data pro druhý tagger musela být upravena s užitím prvního taggeru (který byl trenován na stejných datech). To je samozřejmě technicky možné, ale vzhledem k tomu, že taggery na svých vlastních trénovacích datech fungují jinak („lépe“) než na datech dosud neviděných, neodrážela by takto redukovaná nabídka realitu. Z toho sice formálně nikterak nevyplývá, že by výsledek s přetrénováním nemohl být lepší, nicméně ve spojení s důvody uvedenými v předchozím odstavci je to vskutku velmi nepravděpodobné.)

Výsledky sériové kombinace (pro přehlednost jsou zopakovány i výsledky samotných taggerů):

	–	root	disheu1
a	94.27 %	92.51 %	92.55 %
b	95.13 %	95.48 %	95.30 %
m	95.43 %	**95.64 %**	95.44 %

Feature-based taggeru pravidla vzhledem k chybějícímu přetrénování značně ublížila, zbylým dvěma naopak pomohla, přičemž celkové pořadí taggerů zůstalo zachováno. Není nikterak překvapivé, že méně úspěšnému (a také jednodušeji fungujícímu) taggeru *b* pomohla pravidla více než sofistikovanějšímu taggeru *m* (redukce chyby 7.19 % vs. 4.60 %). Přidání heuristiky se u těchto dvou taggerů ukázalo jako nevhodné, úspěšnější jsou varianty využívající pouze bezpečnou sadu pravidel (root).

3.2 Sériová kombinace tagger – pravidla

Jak už jsme uvedli v úvodu této kapitoly, pravidla lze použít nejen pro redukci víceznačné morfologické nabídky, nýbrž i pro zpětnou kontrolu jednoznačného označkování provedeného někým jiným (tedy buď ručně anotátorem, nebo automaticky taggerem).

Při běžné disambiguaci víceznačného vstupu pravidla fungují tak, že v případě nalezení nepřípustné kombinace tagů se příslušné pravidlům nevyhovující tagy odmažou, bez ohledu na to, zda ještě jiné tagy zbývají. Samozřejmě se tedy může stát i to, že různá pravidla smažou u jednoho tokenu postupně všechny značky. Tato situace není běžná a může znamenat buď kolizi pravidel (nejčastěji chybu jednoho z nich), nebo problém v textu či jeho zpracování (gramatická chyba, překlep, nedostatečný recall morfologické analýzy (zejména u cizích slov) apod.), ne nutně přímo na místě smazané značky. Chyby pravidel se dají tímto způsobem snadno najít a opravit, nicméně správný vstup nám nezaručí nikdo, proto je třeba zavést pro tyto případy nějaké „záchytné“ chování systému. Implicitní nastavení je takové, že pokud dojde ke smazání všech značek u jednoho tokenu, tyto značky se před vydáním věty na výstup zase vrátí, ovšem modifikované speciální úpravou, podle níž lze poznat, co se stalo. Výstup je tedy korektní (žádný token nezůstane bez značek) a zároveň umožňuje ladění, protože jsme o žádnou informaci nepřišli.

Pro kontrolu jednoznačného označkování použijeme pravidla úplně stejně jako při běžné disambiguaci, ovšem zajímají nás pouze tyto problémové případy – tedy smazání jediné nabídnuté značky pravidly. Zde totiž může být (krom možnosti chyby v pravidlech či v textu) ještě jeden důvod, proč se tak stalo, a to chybné označkování, což je právě to, co hledáme.

Abychom mohli tento postup použít pro automatické opravy, je třeba zodpovědět následující otázky:

1. Zda smazání značky v důsledku chybného značkování je výrazně častější než smazání v důsledku problému v textu.
2. Zda lze s dostatečnou úspěšností určit, která značka je chybná (nemusí to být nutně ta smazaná).
3. Zda lze s dostatečnou úspěšností přiřadit správnou značku místo nalezené chybné.

První otázku na chvíli odložme a zaměřme se na druhou. Nalezení skutečného místa chyby (ať už v textu, nebo ve značkování) je velký problém, za současných podmínek z větší části neřešitelný. Do jisté míry lze předvídat a ve větě vyhledávat předem dobře definované

gramatické chyby a překlepy, při obecné kolizi pravidel se však musíme (za předpokladu bezchybnosti pravidel) spokojit pouze s vágním „chyba (v textu či značkování) je s velkou pravděpodobností přímo na místě smazání“, případně s pravdivým, leč ne tolik užitečným tvrzením „v této větě je někde chyba“.

3.2.1 Možnost nahrazení konkrétní značky

První dvě otázky můžeme tedy spojit do jedné, která zní: „Jaká je pravděpodobnost, že smazání značky bylo způsobeno tím, že právě tato značka byla přiřazena chybně?“ Odpověď lze snadno odhadnout z porovnání s ruční anotací na testovacích datech, výsledky uvádíme pro tagger Morče:

	root	disheu1
smazaných značek	0.78 %	0.83 %
z toho skutečně chybných	59.72 %	59.66 %
správných	40.28 %	40.34 %

Zde vidíme, že převaha námi hledaných případů nad ostatními (tedy takovými, kdy smazání značky způsobila chyba u jiné značky nebo chyba v textu věty) není nikterak přesvědčivá. Navíc se vzhledem k nadprůměrné kvalitě textů v PDT jedná o optimistický odhad, při rozsáhlejší ruční analýze korpusu SYN2005 jsme seznali, že coby důvody pro smazání značky spíše převažují chyby v textu. Z tohoto důvodu nemá smysl se o automatické zpětné opravování značek pokoušet – pokud bychom prohlásili všechny pravidly zamítnuté značky za chybné a hledali za ně náhradu, téměř v polovině případů si tím naopak zcela jistě uškodíme (přičemž ve zbylých případech si sice můžeme pomoci, leč nikdo nám to nezaručí). Statistické taggery se totiž s chybnými větami (gramatické chyby, typografické chyby nebo překlepy) dokáží vyrovnat podstatně lépe než pravidla.

3.2.2 Možnost odmítnutí celé věty

Vrátíme se ještě ke druhé možnosti využití informace o smazání značky, kdy ji interpretujeme jako „někde ve větě je chyba“ (v textu nebo ve značkování). Toho se kdysi pokoušeli využít Pavel Květoň s Pavlem Krbcem v nepublikovaném experimentu, kdy pravidla postupně odmítala hypotézy statistického taggeru nastaveného tak, aby vydával N nejnadějnějších výstupů (n-best). Experiment však nebyl úspěšný, pravidla většinou buď přijala hned první vstup, nebo odmítla všechny.

Dle našeho názoru je odmítací potenciál pravidel obtížné využít k přímému vylepšení značkování, může však být užitečný při rozdělování textů na „hezké“ a „ošklivé“ (jak co se týče obsahu, tak značkování), a tedy pomoci při přípravě dat pro trénování metod bez učitele (unsupervised). Konkrétně lze například vytvořit subkorpus „hezkých“ vět – statistické taggery se na všech značkách shodnou a pravidla žádnou z nich nesmažou.

3.3 Třífázové značkování s určením slovního druhu

K metodě popsané v tomto oddílu nás vedla následující myšlenka: domníváme se, že pravidla nemohou v rámci sériové kombinace pravidla – tagger projevit veškeré své schopnosti. Jejich uplatnění totiž brzdí mnoho homonymií, které sama nemohou rozhodnout buď zatím, nebo vůbec. Důvodem je jednak přílišná opatrnost, jednak křížová závislost pravidel – často se stává, že pravidlo A potřebuje pro své uplatnění na daném vstupu mít tento vstup zjednoznačněn pravidlem B, ovšem pravidlo B čeká zase na výsledek aplikace pravidla A. Zdaleka nejobtížnější překážkou aplikace pravidel je slovnědruhová homonymie, kterou ovšem statistické taggery umějí rozhodovat s velmi vysokou úspěšností. Návrh kombinované metody je tedy následující:

1. Nechat tagger určit slovní druh, což provedeme tak, že ho necháme určit celou značku a vrátíme všechny značky z morfologické nabídky, které se s vybranou značkou shodují na druhé pozici, tedy v detailním určení slovního druhu (to implikuje i shodu na pozici slovního druhu, tagset je zde redundantní)
2. Výstup předchozího kroku předložit pravidlům, kterým se tím otevře podstatně více možností než při prořezávání kompletní morfologické nabídky (odstraní se mnoho křížových závislostí).
3. Výstup pravidel finálně zjednoznačnit opět taggerem. Tento tagger nemusí být nutně totožný s taggerem použitým v první fázi.

Úspěšnost jednotlivých taggerů v určení SUBPOS:

a	**99.31 %**
b	99.22 %
m	99.25 %

Výsledky prvního kroku (zvídavý čtenář může zapřemýšlet, proč se recall vždy liší od výsledku v předchozí tabulce[1]):

[1]) Důvodem je, že recall morfologie pro slovní druh je vyšší než pro celou značku – anotátor zřejmě v některých případech zvolil tag, který (aktuální verze) morfologie vůbec nenabízí, ovšem s některým z nabízených tagů se shoduje ve slovním druhu.

	precision	recall	F-measure
a	30.05 %	98.92 %	46.10 %
b	30.10 %	98.83 %	46.15 %
m	30.10 %	98.87 %	46.15 %

Výsledky druhého kroku:

	precision	recall	F-measure
a+root	64.81 %	98.68 %	78.24 %
a+disheu1	70.53 %	98.36 %	82.15 %
b+root	65.07 %	98.59 %	78.40 %
b+disheu1	70.81 %	98.27 %	82.31 %
m+root	65.07 %	98.62 %	78.41 %
m+disheu1	70.81 %	98.30 %	**82.32 %**

Celkové výsledky (řádky specifikují prostředky použité v prvním a druhém kroku, sloupce taggery použité ve třetím kroku):

	a	b	m
a+root	92.81 %	95.68 %	**95.78 %**
a+disheu1	93.08 %	95.69 %	95.77 %
b+root	92.76 %	95.63 %	95.72 %
b+disheu1	93.02 %	95.64 %	95.71 %
m+root	92.79 %	95.63 %	95.75 %
m+disheu1	93.05 %	95.64 %	95.73 %

Výsledky jsou velmi příznivé, lepší než u prosté sériové kombinace. Nepřekvapí, že tagger *a* je bez přetrénování naprosto nevhodný pro použití v poslední fázi, nicméně nepředpokládáme, že by mu přetrénování pomohlo k překonání zbylých dvou taggerů (na to je výkonnostní propast mezi ním a jimi příliš velká), proto nás nemusí mrzet, že jsme to nezkusili. Nejlepší výsledek zaznamenala kombinace taggeru *a* v první fázi (což je v souladu s pozorováním, že tento tagger je nejúspěšnější v určování SUBPOSu) následovaného sadou pravidel *root* a taggerem *m*, nicméně rozdíly v úspěšnosti všech kombinací končících taggerem *m* jsou vskutku nepatrné. Pokud bychom chtěli použít pouze jeden tagger pro obě fáze procesu (což můžeme chtít např. v důvodů implementačních nebo licenčních), bude to samozřejmě tagger *m*.

Zajímavé je, že v tomto experimentu se, asi jako v jediném, nijak výrazně neprojevil rozdíl mezi užitím jednotlivých skupin pravidel. Nejen z tohoto důvodu jsme provedli kontrolu, zda mají pravidla vůbec nějaký efekt, tedy zda zlepšení netkví jenom v rozdělení úlohy určení značky na dvě fáze (SUBPOS a zbytek).

Kontrola účinnosti pravidel – výsledky pouze při kombinaci prvního (řádek) a třetího (sloupec) kroku:

	a	b	m
a	92.96 %	95.18 %	**95.42 %**
b	92.90 %	95.13 %	95.37 %
m	92.92 %	95.15 %	95.40 %

Jak vidno, naše obavy byly plané, bez zařazení pravidel nejsou taggery v této kombinaci schopny překonat nejlepší z nich (tedy *m*) použitý samostatně, dokonce vůbec ani dosáhnout jeho výsledku.

Uvedenou metodu jsme svého času pokládali na nejlepší a také jsme ji (ve verzi *m – root – m*) použili pro první oficiální označkování korpusu SYN2005.

3.4 Třífázové značkování se sjednocením taggerů

V tomto oddílu je popsána v současné době nejúspěšnější metoda značkování češtiny. Od předchozích metod se zásadně liší tím, že návrh od začátku počítá s použitím více než jednoho statistického taggeru. Její princip opět nejlépe přiblížíme popisem jednotlivých kroků:

1. Data nechat označkovat nezávisle N taggery.
2. Z výsledků taggerů udělat sjednocení (pro každý token tedy dostaneme 1 až N tagů, podle toho, jak moc se taggery shodly nebo lišily).
3. Takto získanou nabídku nechat prořezat pravidly.
4. Provést závěrečné zjednoznačnění jedním taggerem.

Máme-li k dispozici tři taggery, naskýtá se nám více variant této metody – v úvodním sjednocení lze použít buď všechny tři, nebo jejich libovolnou dvojici, v závěru se může použít libovolný z nich. Výsledky všech těchto možností v jednotlivých krocích jsou následující:

1. a 2. krok (sjednocení taggerů):

	precision	recall	F-measure
$a \cup b$	92.18 %	96.90 %	94.48 %
$a \cup m$	92.30 %	97.04 %	94.61 %
$b \cup m$	93.19 %	97.05 %	**95.08 %**
$a \cup b \cup m$	90.81 %	97.66 %	94.11 %

3. krok (sjednocení + pravidla):

	precision	recall	F-measure
$(a \cup b) + root$	93.56 %	96.74 %	95.12 %
$(a \cup b) + disheu1$	93.99 %	96.63 %	95.29 %
$(a \cup m) + root$	93.71 %	96.86 %	95.26 %
$(a \cup m) + disheu1$	94.15 %	96.77 %	95.44 %
$(b \cup m) + root$	94.11 %	96.90 %	**95.48 %**
$(b \cup m) + disheu1$	94.46 %	96.81 %	95.62 %
$(a \cup b \cup m) + root$	92.67 %	97.46 %	95.00 %
$(a \cup b \cup m) + disheu1$	93.32 %	97.32 %	95.28 %

4. krok (závěrečné zjednoznačnění taggerem uvedeným ve sloupci):

	a	b	m
$(a \cup b) + root$	95.43 %	95.49 %	95.96 %
$(a \cup b) + disheu1$	95.54 %	95.58 %	95.96 %
$(a \cup m) + root$	95.56 %	96.03 %	95.73 %
$(a \cup m) + disheu1$	95.68 %	**96.05 %**	95.82 %
$(b \cup m) + root$	95.81 %	95.58 %	95.77 %
$(b \cup m) + disheu1$	95.89 %	95.71 %	95.86 %
$(a \cup b \cup m) + root$	95.52 %	95.66 %	95.84 %
$(a \cup b \cup m) + disheu1$	95.69 %	95.80 %	95.95 %

Nejzajímavější výsledek vydala (pro nás poněkud překvapivě) varianta, kdy se v úvodním sjednocení použijí taggery *a* a *m* a závěrečný krok obstará tagger *b*. Rozdíl mezi sadou pravidel *root* a *disheu1* zde nehraje příliš velkou roli, ve většině ostatních variant je však dosti výrazný, a to ve prospěch *disheu1*.

Že zvítězí varianta využívající všechny tři taggery, a nikoli pouze dva, nijak nepřekvapuje, původně jsme však více věřili sjednocení všech tří taggerů v prvním kroku. Pravidla zřejmě zapůsobí tím více, čím menší výběr tagů je jim dán k dispozici. Velmi zajímavý je v tomto kontextu přínos heuristických pravidel, neboť ta byla původně určena k řešení právě opačného úkolu, totiž k odvážnější redukci značek u příliš velké nabídky.

Nenacházíme žádné zdůvodnění, proč zvítězila zrovna varianta $(a \cup m) + pravidla + b$ a ne $(a \cup b) + pravidla + m$. Svou roli zde jistě hraje to, že tagger *b* z definice není citlivý na změnu velikosti morfologické nabídky, přesto bychom však čekali, že varianta $(a \cup b) + pravidla + m$ bude mít menší odstup.

Zajímavá je i nejúspěšnější varianta z těch, u kterých vstupují do hry právě dva taggery, je to $(b \cup m) + disheu1 + m$ (s odstupem následovaná kombinací $(a \cup m) + disheu1 + m$). Tento výsledek je poměrně logický a koresponduje s pořadím úspěšnosti samotných taggerů.

Stejně jako u předchozí metody jsme zkusili vynechat pravidlový krok, abychom ověřili jeho účinnost.

	a	b	m
$(a \cup b)$	94.94 %	95.13 %	**95.87 %**
$(a \cup m)$	95.05 %	**95.87 %**	95.46 %
$(b \cup m)$	95.56 %	95.13 %	95.48 %
$(a \cup b \cup m)$	94.85 %	95.14 %	95.47 %

Odstup téměř dvou desetin procenta jasně ukazuje, že pravidla nemálo přispívají k celkovému výsledku nejlepšího experimentu. Výsledek získaný při tomto ověřování (a to shodně u dvou variant: $(a \cup m) + b$ a $(a \cup b) + m$) ovšem velmi potěší, neboť to je (pro češtinu) poprvé, co se podařilo jen za pomoci kombinace statistických taggerů takto výrazně (o více než čtyři desetiny procenta) překonat úspěšnost nejlepšího z nich. Pravidlový disambiguační systém je velmi komplikovaný a bohužel i řádově pomalejší než statistické taggery, proto tato „odlehčená" metoda jistě najde uplatnění při výpočetně náročném zpracování velkých dat, např. webových textů.

Nejúspěšnější variantu zde popsané metody ($(a \cup m) + disheu1 + b$) jsme s úspěchem použili pro nové oficiální přeznačkování ČNK [1] SYN2000 a SYN2005, jakož i pro nový korpus SYN2006PUB. Oproti metodě použité pro původní označkování korpusu SYN2000 (tagger *a*) došlo ke zvýšení úspěšnosti o 1.54 % (výsledky na e-test, viz následující oddíl), lze tedy důvodně předpokládat, že počet chyb ve značkování korpusu SYN2000 se snížil o více než 1.5 milionu.

3.5 Shrnutí

Neoevaluuje-li mě Julie, oevaluuji se sám!

V této kapitole jsme jednak ověřili úspěšnost dříve navržených kombinačních experimentů na současných verzích nástrojů a aktuálních datech, jednak navrhli a provedli experimenty vlastní, které se ukázaly jako výrazně úspěšnější. Za celkový výsledek můžeme prohlásit nalezení několika nových metod, z nichž si můžeme vybrat v závislosti na tom, jaké nástroje máme v danou chvíli k dispozici (a chceme použít – již byly zmíněny rychlostní důvody, pro které někdy nemusí být použití pravidel vhodné).

Následuje shrnutí těchto nejúspěšnějších metod spolu s výsledky nejen na PDT 2.0 d-test datech, nýbrž i na e-test.

Máme k dispozici	Nejúspěšnější metoda	d-test	e-test
jeden tagger	m	95.43 %	95.12 %
dva taggery	$-$	–	–
tři taggery	$(a \cup m) + b$ nebo $(a \cup b) + m$	95.87 %	95.52 %
1 tagger + pravidla	$SUBPOSm + root + m$	95.75 %	95.44 %
2 taggery + pravidla	$(b \cup m) + disheu1 + m$	95.86 %	95.49 %
3 taggery + pravidla	$(a \cup m) + disheu1 + b$	**96.05 %**	**95.68 %**

Morče coby samostatně nejúspěšnější tagger figuruje i ve všech kombinacích s nejlepšími výsledky. Přínos taggerů *a* a *b* je téměř srovnatelný. Pro různé úlohy jsou vhodné různě odvážné sady pravidel.

V následující tabulce jsou uvedeny redukce chyby v porovnání Morčete, nejlepší kombinace bez pravidel a nejlepší kombinace s pravidly.

Metoda	Morče	Sjednocení bez pravidel
Sjednocení bez pravidel	8.20 %	—
Sjednocení s pravidly	11.48 %	3.57 %

Jak zlepšení dané sjednocením taggerů, tak další vylepšení tohoto sjednocení přidáním pravidel je podle všech obvykle užívaných testů statisticky signifikantní.

3.5.1 Analýza chyb

V následující tabulce naleznete srovnání chybovosti (v procentech) na jednotlivých pozicích tagu u vybraných metod (tři samostatné taggery, nejúspěšnější metoda bez pravidel (s1) a nejúspěšnější metoda s pravidly (s2)). V rozpisu tagu jsou vynechány nevyužité pozice – rezervy.

	a	b	m	s1	s2
1 (POS)	0.61	0.70	0.66	0.57	0.57
2 (SUBPOS)	0.69	0.78	0.75	0.64	0.64
3 (GENDER)	1.82	1.49	1.66	1.39	1.37
4 (NUMBER)	1.56	1.30	1.38	1.18	1.15
5 (CASE)	4.03	3.53	3.08	2.85	2.62
6 (POSSGENDER)	0.02	0.03	0.03	0.02	0.02
7 (POSSNUMBER)	0.01	0.01	0.01	0.01	0.01
8 (PERSON)	0.06	0.07	0.08	0.06	0.05
9 (TENSE)	0.05	0.08	0.07	0.05	0.04
10 (GRADE)	0.29	0.28	0.30	0.26	0.27
11 (NEGATION)	0.29	0.31	0.33	0.28	0.28
12 (VOICE)	0.05	0.08	0.06	0.05	0.04
15 (VAR)	0.31	0.31	0.31	0.28	0.29

Z tabulky je vidět, že sjednocení taggerů vylepšilo (nebo alespoň nezhoršilo) výsledky dosažitelné samostatnými taggery na všech pozicích, zajímavých je zejména prvních pět pozic tagu (odrážejících základní morfologické kategorie slovní druh, rod, číslo a pád). Ostatní pozice nebývají využívány tak často, a proto je i chybovost na nich dosti nízká u všech metod. Přidání pravidel pak vylepšilo výsledky kombinace taggerů zejména u pádu, kde jsou nadále patrné největší rezervy.

Následují konfuzní matice výsledků na páté pozici (pád) pro sjednocení taggerů bez pravidel a sjednocení taggerů s pravidly. Řádky uvádějí výsledek metody, sloupce referenční anotaci.

Sjednocení bez pravidel:

tg/an	-	1	2	3	4	5	6	7	X
-	82753	37	41	0	18	3	4	7	21
1	53	26027	286	11	939	21	8	5	81
2	9	205	29363	21	146	0	25	14	24
3	1	41	70	5265	54	0	50	23	1
4	50	1835	404	12	21302	1	155	44	15
5	0	8	0	3	2	36	0	1	0
6	3	18	54	15	128	0	17914	3	3
7	29	26	19	8	73	0	0	9010	3
X	115	312	90	7	44	21	14	5	4242

Sjednocení s pravidly:

tg/an	-	1	2	3	4	5	6	7	X
-	82747	39	43	2	18	3	2	7	23
1	50	26063	290	13	883	22	6	7	97
2	8	188	29397	23	128	0	18	16	29
3	0	37	71	5310	48	0	14	24	1
4	37	1561	406	13	21597	1	145	41	17
5	0	10	0	8	2	29	0	1	0
6	3	17	56	18	120	0	17917	3	4
7	31	22	20	8	62	0	0	9022	3
X	109	285	86	6	48	21	11	6	4278

Z matic je vidět, že největší problémy způsobuje homonymie nominativu s akuzativem, což je zároveň případ homonymie, která bez znalosti kontextu činí mnohdy problém i čtenáři (*Československo napadlo Německo.*). V nejbližší době dojde k nové anotaci problémových míst PDT více anotátory, což by nám mělo dát lepší představu o tom, jaký podíl z uvedené chybovosti způsobuje nedokonalost taggerů a kolik případů je strojově a případně i ručně nerozhodnutelných.

3.5.2 Možná rozšíření

V této kapitole není popsáno ani vyhodnoceno množství experimentů, které jsme prováděli průběžně ve snaze vylepšit jednotlivé metody. Jelikož se všechny ukázaly jako méně úspěšné než „normální“ varianty vylepšovaných metod, nepovažujeme podrobnější popis za nutný, uvedeme však alespoň jejich výčet:

- sériová kombinace s taggerem *m* přetrénovaným na pravidly zpracovaných datech
- pokusy s jinými množinami pravidel, než jsou standardní *root* a *disheu1* (buď zmenšenými, nebo naopak rozšířenými o *heuristiku2*), a to u sériové kombinace i obou trojfázových značkování
- iterativní sjednocení (z výsledků tří taggerů po sjednocení a pravidlech se opět udělá sjednocení, které se zpracuje pravidly a taggery)
- „opatrné“ sjednocení (kde se taggery neshodnou, vrátíme buď celou morfologickou nabídku, nebo všechny tagy mající stejný SUBPOS jako jeden z tagů zvolených)
- různé možnosti kombinace sériového předřazení pravidel a sjednocení (sjednocení obyčejných verzí taggerů s pravidly vylepšenými verzemi apod.)

Obecně se ukázalo, že nejlepších výsledků dosáhneme, pokud každý komponent použijeme v kombinaci právě jednou (tedy nikoli opakovaně), to platí zejména pro pravidla, jedinou výjimkou je třífázové značkování s vrácením slovního druhu, kde použijeme stejný tagger dvakrát, ovšem napoprvé jen částečně.

Podle nás hlavním důvodem celkově vysoké úspěšnosti kombinovaných metod je vedle dílčí úspěšnosti jednotlivých komponent také jejich principiální odlišnost. To platí nejen pro taggery vs. pravidla, ale i pro taggery mezi sebou – každý používá jiný algoritmus, byť jsou natrénovány na stejných datech.

Pokud bychom měli tu možnost, bylo by dobré vyzkoušet ještě následující kroky:

- v první fázi třífázového značkování s vrácením slovního druhu použít nějaký „parciální tagger“, který se specializuje pouze na slovní druh
- provést veškeré experimenty ve variantě s přetrénováním taggeru *a*, případně i *m*
- přidat do množiny, z níž se vybírají podmnožiny pro sjednocení, libovolný další tagger (dostatečně úspěšný a dostatečně odlišný od ostatních).

Kapitola 4

Rozšíření pravidel na syntax

V této kapitole podrobně popíšeme experiment s rozšířením působnosti disambiguačních pravidel do oblasti syntaxe – od motivace přes realizaci až po vyhodnocení a závěry.

Cílem experimentu bylo za pomoci přiměřeného rozšíření již existujících disambiguačních pravidel (jakož i programového vybavení) získat informace o povrchové syntaxi věty a ověřit, zda mohou napomoci (opět již existujícím) nástrojům pro parsing češtiny. Důraz byl kladen na to, aby byl „vytěžen“ syntaktický potenciál existujících pravidel, potažmo jazyka LanGR, snažili jsme se tedy vyhnout jednak psaní přílišného množství zcela nových pravidel zaměřených pouze na syntax, jednak zásadnímu přepracování či obcházení interpretu jazyka LanGR. Jinými slovy, cílem nebylo napsat nový pravidlový parser, nýbrž pouze zkusit zužitkovat to, co již máme (disambiguační pravidla).

Toto předem dané omezení bylo také důvodem, proč jsme experiment uzavřeli coby neúspěšný. Ukázal sice některé slibné cesty, kterými se ti, kdo budou v budoucnu chtít kombinovat pravidlové a statistické metody při parsingu, mohou zkusit ubírat, leč tyto cesty leží již mimo rámec našeho zadání (nelze je prověřit bez zásadního přepracování formalismu i systému pravidel). Přesto doufáme, že některé ze závěrů, které prezentujeme na konci kapitoly, mají natolik obecnou platnost, že budou případným pokračovatelům k užitku.

4.1 Představa

Základní myšlenkou je, že velká část disambiguačních pravidel jde při „ohledávání“ vstupu do značné hloubky, zjištěné informace o struktuře věty jsou však využity jen částečně (pro disambiguaci) a poté zapomenuty, ačkoli by mohly být k užitku i v dalších fázích automatického zpracování textu.

Jako příklad poslouží soubor disambiguačních pravidel pro řešení shody podmětu s přísudkem. Abychom mohli unifikaci potřebných morfologických kategorií u podmětu a přísudku provést, musíme nejprve oba zmíněné větné členy bezpečně identifikovat. A když už tuto informaci máme, byla by škoda ji nevyužít později při budování syntaktické struktury věty. Totéž se týká například přívlastku shodného, shody vztažného zájmena s antecedentem, shody v koordinaci – všechny tyto jevy (a mnoho dalších) disambiguační systém velmi spolehlivě rozpoznává na základě pečlivé a opatrné analýzy vstupní věty.

Dosavadní pravidla sice zdaleka nejsou schopna disambiguovat celý vstup, to však ani není jejich účelem. Jsou velmi opatrná, vydají tedy buď informaci velmi bezpečnou, nebo žádnou. Proto ani nelze pomýšlet na úplný parsing věty – přirozeným výstupem syntaktického rozšíření pravidel by měl být (v případě, že toužíme po stromové struktuře) „bezpečný les“, který lze buď doplnit o chybějící hrany záchytnými pravidly, nebo postoupit k dalšímu zpracování či využití jiným nástrojům. Výstup však může být i podstatně obecnější, tedy pouze ve formě jednotlivých informací o vztazích mezi dvojicemi (případně N-ticemi) větných členů, a to informací jak pozitivních („je zde vztah určitého typu“), tak negativních („mezi těmito větnými členy žádný přímý syntaktický vztah není“). Tyto informace pak mohou jiné nástroje využít mnoha různými způsoby (při samotném stavění závislostního či derivování složkového stromu, při redukování množství výsledků apod.).

4.1.1 Pozitivní a negativní pravidla

Rozpoznávání konkréních jevů a přiřazování správných tagů mají na starost tzv. pozitivní pravidla. Mnohá z nich k syntaktickému využití přímo vybízejí. Vedle nich se v systému vyskytují i pravidla negativní, která mažou tagy v dané konfiguraci nepřípustné (bez ohledu na to, jaké tagy mají být správné a zda vůbec v nabídce z morfologie jsou).

Drtivou většinu negativních pravidel nelze při rozšíření systému na syntax využít přímo, tedy pro bezprostřední vylepšení výsledku

pozitivních pravidel, a to ze dvou důvodů: jednak velká část negativních pravidel vůbec nemá k syntaxi vztah (např. vokalizace, slovosled příklonek), jednak zbylá negativní pravidla, která se syntaxí pracují, nejsou lingvisticky založena na tom, jak má výsledná syntaktická struktura věty vypadat, nýbrž „pouze“ na tom, jak vypadat nemá.

Z těchto negativních informací je značná část nadbytečná v každém případě, neboť vychází z chybných tagů, které byly následně umazány, proto libovolný parser takovouto nepřípustnou konstrukci vůbec nemůže nabídnout, zejména pokud došlo ke zjednoznačnění rozdílných slovních druhů.

Zbylou informaci, která není nadbytečná (např. o tom, že substantivum neshodující se se slovesem v příslušných kategoriích zřejmě nebude podmětem) by již některé jednodušší parsery využít mohly, v naší konkrétní situaci však k užitku nebude, neboť takovouto možnost pozitivní pravidla v žádném případě nemohla nabídnout (a my vybíráme pouze z jejich nabídky). Zůstává zde tedy potenciál využití pouze v případě kombinování s parserem, který je schopen takovéto „zákazy“ zužitkovat (a zároveň je sám o sobě dostatečně úspěšný na to, aby mělo smysl ho dále vylepšovat). Takovýto parser v době našeho experimentu k dispozici nebyl, proto se k této otevřené možnosti vrátíme až v závěru kapitoly.

4.2 Rozšíření pravidel

Prvním krokem v našem experimentu tedy bude, že všechna pozitivní pravidla (tj. taková, co vyhledávají konkrétní jevy a ponechávají u nich pouze ty morfologické značky, které jsou přípustné) mající vztah k syntaxi rozšíříme tak, aby vydávala navíc informaci o zjištěném vztahu.

Následuje zjednodušený příklad pravidla na shodu podmětu s přísudkem:

```
1  rule ShodaSubjPred1 {
2
3  RuleVariant v1 {
4
5  // konfigurační část
6
7  // začátek věty
8  ITEM SentenceStart;
9
10 // případná rozvití podmětu
11 SEQUENCE OF IsSafe SyntacticAdjective and Nominative;
12
13 // podmět v nominativu
14 subjekt = ITEM IsSafe Noun and Nominative;
15
16 // větné členy, které mohou stát mezi podmětem a přísudkem
17 SEQUENCE OF IsSafe (Adjective or Pronoun or Numeral or Infinitive
 or Adverb or Preposition or Particle or Interjection);
18
19 // přísudek
20 predikat = ITEM IsSafe (FiniteVerb and (not (lemma == "být")));
21
22 // konec konfigurační části
23 // výkonná část
24
25 // disambiguační akce
26 UNIFY subjekt WITH predikat IN [person,number,gender];
27
28 // syntaktická akce
29 CONNECT subjekt predikat "Desc: Subj, Dom: R";
30
31 };
32 };
```

V konfigurační části pravidla specifikujeme podmínky, za kterých se pravidlo uplatní, ve výkonné části probíhá jednak disambiguační akce, jednak akce syntaktická – disambiguační akce je původní, syntaktickou jsme v rámci rozšíření pravidla přidali. Morfologická disambiguace a syntaktické zpracování tedy probíhají zároveň.

Interpret jazyka LanGR neumožňuje za běhu uchovávat žádné jiné informace než znění věty a původní i současný (postupně disambiguovaný) stav morfologické nabídky, speciálně tedy neumožňuje průběžně budovat žádné datové struktury nesoucí informace neobsažené přímo ve zpracovávaných datech. Proto můžeme se zjištěnou syntaktickou informací udělat pouze to, že ji pošleme na výstup, kde ji převezme další program a zpracuje hromadně až po ukončení průchodu věty LanGRem.

Tento navazující program, který z jednotlivých syntaktických informací vybuduje souvislou strukturu, dostane k dispozici právě tyto informace a nic jiného (ani znění věty, ani její morfologické značkování). Účelem tohoto opatření je přesně oddělit fázi lingvistického zpracování věty (kterou má na starosti LanGR, neboť k tomu má velmi silné prostředky, jak formální, tak jazykové – slovníky apod.) a technickou fázi stavění syntaktické struktury. V té bychom totiž mohli větu lingvisticky analyzovat jen tehdy, pokud bychom některé z prostředků LanGRu duplikovali, což je zbytečné a hlavně nečisté. Na druhou stranu přináší tato oddělená funkcionalita jisté problémy, ke kterým se dostaneme v závěru této kapitoly, jakož i k možnostem jejich řešení.

Při zpracování věty *Ema(1) má(2) mísu(3).(4)* vyprodukuje výše uvedené pravidlo pro shodu podmětu s přísudkem (za předpokladu, že *má* je již zjednoznačněno na sloveso) následující informaci:

```
CONNECT 1 2 Desc: Subj, Dom: R
```

Čísla 1 a 2 udávají pořadí slov ve větě, která jsou v relaci (*Ema* a *má*), *Subj* je typ relace, *Dom: R* upřesňuje (je-li to třeba), který člen z uvedených je řídící (v tomto případě pravý, tedy 2 – *má*). Tento údaj nemusí nutně evokovat představu podřízenosti – v této fázi jde čistě o to, abychom věděli, které z dvojice slov (která už budeme v dalším zpracování identifikovat pouze pořadovými čísly, neuvidíme jejich tvar ani morfologické značky) je slovesem a které podmětem.

Z předchozího textu je patrné, že jsme se snažili udržet tuto část zpracování co nejobecnější, nezávislou na žádném konkrétním syntaktickém formalismu, vlastně jde pouze o popis jednotlivých jazykových jevů, které se ve větě vyskytují, bez ohledu na to, jak je později budeme chtít zformalizovat a zachytit. Jinými slovy vnitřní reprezentace výsledků pravidel nemá formu žádného typu stromu, nýbrž

pouze množiny dvojic (v původním návrhu N-tic) větných členů, které jsou spolu v nějakém syntaktickém vztahu.

Lze si velmi dobře představit, že z uvedených informací vybudujeme jak závislostní, tak složkový strom, případně i úplně jinou strukturu, stejně tak mohou pouze napomoci jinému nástroji takovou strukturu vybudovat.

V době, kdy syntaktické rozšíření vznikalo, obsahoval disambiguační systém kolem 2000 pravidel. Všechna jsme ručně prošli, analyzovali a 430 z nich rozšířili o poskytování syntaktické informace – některá z nich šla použít přímo, jiná bylo třeba mírně rozšířit. Zbylá pravidla byla negativní, nevztahovala se k syntaxi nebo poskytovala redundantní informaci (řešila speciální případy, na něž se později (po disambiguaci oním speciálním pravidlem) „chytilo“ pravidlo obecnější). Dále jsme dopsali cca 80 nových, čistě syntaktických „záchytných“ pravidel, o kterých bude řeč v oddílu 4.3.2.

4.3 Budování struktury

4.3.1 Volba formalismu

Druhou fázi zpracování, tedy skládání jednotlivých syntaktických informací do ucelené struktury popisující vstupní větu, jsme už museli pro účely průběžné evaluace přizpůsobit konkrétnímu formalismu. Zvolili jsme (ne nutně natrvalo) analytickou rovinu PDT [7], zejména proto, že je pro ni k dispozici velké množství ručně anotovaných dat.

V analytickém závislostním stromě, jak byl definován pro účely PDT, má každý uzel přiřazeno pořadové číslo „otce“ a analytickou funkci popisující typ relace. Kořen věty, obvykle sloveso, a závěrečná interpunkce visí na „umělém“ otci s pořadovým číslem 0. Správné přiřazení analytických funkcí není předmětem běžné evaluace parserů (měří se pouze úspěšnost přiřazení otců), proto jsme se na ně nesoustředili, používali jsme vlastní obdobu analytických funkcí s tím, že není problém jejich systém kdykoli v případě potřeby upravit, aby byl kompatibilní s PDT. Dále se tedy budeme zabývat pouze algoritmem přiřazení otce (a jeho úspěšností). Budeme-li hovořit o analytických funkcích, máme tím na mysli naše vlastní, jsou to identifikátory typu relace, jako např. *Subj* z příkladu v předchozím oddílu.

Ještě dodáme, že pro začátek jsme se rozhodli vydávat pro každou větu pouze jeden strom (les), neboť v té fázi vývoje, do které systém došel, by možnost vydávat více výsledků nebyla přínosem, nýbrž by pouze zdržovala ladění pravidel (sporná data ve většině případů znamenala chybu v pravidlech či stavebním algoritmu, a ne homonymní větu). Jelikož ke zjednoznačnění a výběru jediného stromu (lesa) dochází až v samém závěru zpracování, nebyl by problém toto chování kdykoli změnit.

4.3.2 Priorita, záchytná pravidla

V úvodním příkladu jsme pro zjednodušení vynechali prioritu, což je další údaj, který s sebou syntaktická informace vydaná pravidly nese. Priorita odráží, do jaké míry danému pravidlu, které informaci vydalo, „věříme“ – původní disambiguační pravidla mívají většinou nejvyšší prioritu, protože jsou velmi bezpečná, pomocná čistě syntaktická pravidla (obsahující zjednodušené popisy jevů, které ještě v disambiguačním systému zachyceny nejsou) mají prioritu střední a „záchytná“ pravidla typu „mým otcem je nejbližší sloveso“ mají prioritu nejnižší. Tato informace se zachová i v průběhu budování struktury, nakonec má tedy každá vybudovaná závislost přiřazenu i prioritu odpovídající prioritě pravidla, na základě něhož vznikla.

Důvod pro uvedený postup (tedy přidávání méně bezpečných pravidel do systému a jejich odstupňování pomocí priority) je ten, že jakákoli informace o větném členu je lepší než žádná (mnohdy nejen pro samotný větný člen, ale i pro úspěšné zpracování zbytku věty, viz následující oddíl), ovšem máme-li informací více, vždy samozřejmě upřednostníme tu důvěryhodnější (je tedy třeba je nějak rozlišit), a i u finální struktury se pak, podle typu úlohy, můžeme rozhodnout, jak velká část výsledku nás zajímá (buď můžeme chtít souvislé stromy, ovšem s velkým rizikem chyby, nebo můžeme použít pouze ty nejbezpečnější informace, které však máme pouze u malého množství syntaktických vztahů, případně něco mezi).

Protože v průběhu zpracování věty LanGRem se syntaktická informace neuchovává, nemáme jak ověřit, zda o každém větném členu už něco víme, tedy zda (a kde) je potřeba aplikovat záchytná pravidla. Proto je aplikujeme všude, kde je to možné (tato nebezpečná pravidla v žádném případě nedisambiguují, tedy nemění vstup, pouze se vyjadřují k syntaxi), s tím, že získané informace pak při budování struktury podle potřeby můžeme, ale nemusíme využít.

4.4 Transformace a skládání složitějších vztahů

Jelikož syntaktická informace dodávaná LanGRem má svým tvarem k popisu závislosti velmi blízko, stačí na první pohled pouze dát pro každý uzel dohromady všechny možnosti otce a vybrat toho s nejvyšší prioritou (to celé po nezbytných, leč zcela automatických úpravách informací o jevech, u nichž se se zvoleným formalismem neshodneme v tom, který člen je řídící a který závislý).

V případě nejistoty, např. máme-li dvě možnosti výběru otce se stejnou prioritou, můžeme zkusit některou z nich vyloučit tím, že ověříme, zda ve stromě (či jeho jednotlivých komponentách) nevznikají cykly – to můžeme samozřejmě ověřit v každém případě, nejen když se potřebujeme rozhodnout.

Tímto způsobem bychom ale mohli postavit korektní stromy pouze u velmi jednoduchých vět. U větší části „reálného“ vstupu je třeba provádět rozličné lingvisticky motivované transformace a budovat cílové závislosti z více dílčích vztahů.

4.4.1 Koordinace

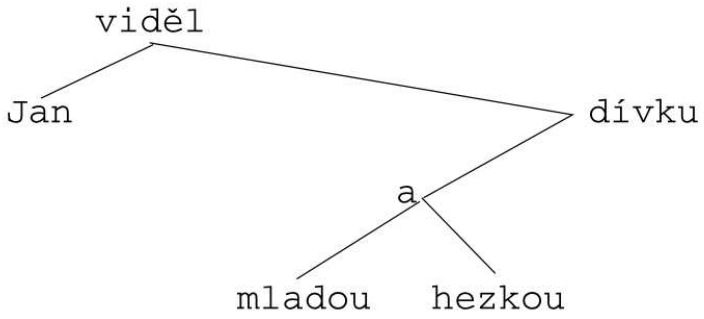
Typickým příkladem jevu, který je třeba speciálně ošetřit, jsou koordinace. Větný člen, který je součástí koordinace, může dostat z LanGRu více různých (správných) údajů o tom, kam patří – jde-li například o koordinované přívlastky shodné, můžeme pro jeden z nich (či oba) obdržet jak informaci o příslušnosti ke koordinaci, tak o shodě se jménem, které rozvíjejí.

Pro větu *Jan(1) viděl(2) mladou(3) a(4) hezkou(5) dívku(6).(7)* může výstup z LanGRu vypadat například takto:

```
CONNECT 1 2 Desc: Subj, Dom: R
CONNECT 0 2 Desc: AuxS, Dom: L
CONNECT 3 4 Desc: Coord, Dom: R
CONNECT 3 6 Desc: Attr, Dom: R
CONNECT 4 5 Desc: Coord, Dom: L
CONNECT 5 6 Desc: Attr, Dom: R
CONNECT 2 6 Desc: Obj, Dom: L
CONNECT 0 7 Desc: AuxG, Dom: L
```

Pro přívlastky *mladou* a *hezkou* máme po dvou informacích, z nichž ani jednu nesmíme příliš brzy zavrhnout, i kdyby měla nižší prioritu než druhá (tím se i technická stránka zpracování poněkud komplikuje). V tomto případě (rozpoznáme podle analytických funkcí) má přednost koordinace, přívlastky tedy dostanou jako otce spojku *a* (5) a tato spojka převezme jejich další potenciální otce, v tomto případě

jen jednoho, *dívku* (6). Vznikne korektní, jednoznačná a bezesporná struktura.

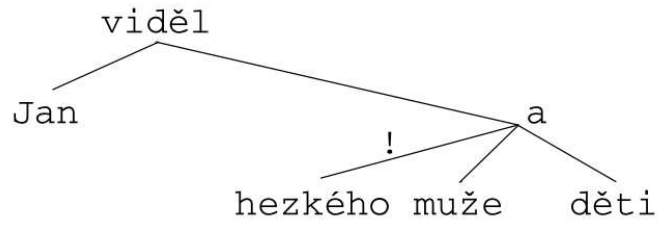


Budování koordinačních struktur má bohužel i své neduhy. V případě věty *Jan viděl hezké ženy a děti.* netuší ani čtenář, zda se hodnocení vzhledu vztahuje jen k ženám, nebo i k dětem, proto to samozřejmě nemůže tušit ani náš systém – vydává-li jen jedno řešení, musí si vybrat, a jelikož tak činí pouze na základě analytických funkcí (jiné informace nemá k dispozici), bude rozhodnutí v podobných situacích (přívlastek rozvíjející potenciálně buď celou koordinaci, nebo jen jeden její člen) vždy stejné – takové, jaké předem zvolíme a implementujeme. Další možností je vydávat v těchto případech oba výsledky.

A zde je právě kámen úrazu – ať už zvolíme řešení jakékoli, bude takové i v případě, kdy věta ve skutečnosti homonymní není, neboť jedna z variant je negramatická, ale program stavějící strukturu nemá možnost to ověřit. Příkladem mohou být věty *Jan(1) viděl(2) hezkého(3) muže(4) a(5) děti(6).(7)* a (již uvedená) *Jan(1) viděl(2) hezké(3) ženy(4) a(5) děti(6).(7)*. LanGR u obou vět zjistí (krom jiného), že substantiva na 4. a 6. pozici jsou v koordinaci a že přívlastek *hezké(ho)* rozvíjí první z nich (shoduje se s ním). Předá tedy programu budujícímu strukturu v obou případech stejnou informaci:

```
...
CONNECT 4 5 Desc: Coord, Dom: R
CONNECT 3 4 Desc: Attr, Dom: R
CONNECT 6 5 Desc: Coord, Dom: R
```

Pokud bychom se rozhodli, že koordinace vždy přebírá přívlastky svého prvního členu, vydáme v případě věty *Jan viděl hezkého muže a děti.* chybný výsledek (*hezkého* rozvíjí celou koordinaci).

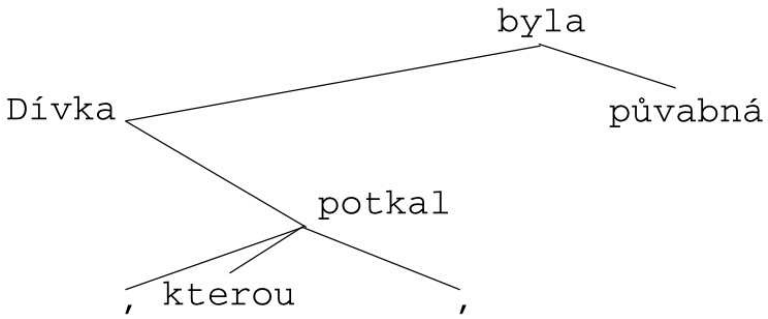


Pokud bychom připustili obě varianty (přebírání i nepřebírání), stále bude polovina výsledků pro větu *Jan viděl hezkého muže a děti.* chybná. Přitom jde o jednoduchou neshodu, kterou by s prostředky LanGRu bylo možné snadno ověřit, my však už nemáme možnost se do něj vrátit.

Zde jsme narazili na meze našeho návrhu – strukturu nelze uspokojivě vybudovat, pokud jsme text analyzovali pouze na jeden průchod a nemáme možnost upřesňovat výsledky průběžně. V tomto konkrétním případě by asi bylo možné problém vyřešit dobře mířeným „preventivním" negativním pravidlem, ale dostaneme se i ke složitějším problémům, které už takto vyřešit nepůjdou.

4.4.2 Antecedent vztažného zájmena

Dalším příkladem, nikoli na nedostatečnost našeho systému, nýbrž na skládání složitějších vztahů, je zapojování vztažných vět do struktury souvětí. Pro začátek uvedeme velmi triviální příklad, který bude nejlépe demonstrovat řešenou úlohu: *Dívka(1),(2) kterou(3) potkal(4),(5) byla(6) půvabná(7)*[1]*).(8)*



V analytickém stromě visí podstrom vztažné věty *, kterou potkal,* na slově *dívka*, přičemž kořenem podstromu je sloveso. Otcem slova *potkal* tedy má být *Dívka*. Mezi *potkal* a *Dívka* však žádný formálně

[1]) Škodolibí čtenáři mohou nahradit slůvkem *ohyzdná*.

vyjádřený syntaktický vztah (např. shoda) není – že má výsledný strom vypadat zrovna takto, je pouze věc definice analytické roviny. Z pravidel však dostaneme informace o ostatních vztazích, ze kterých už kýžený vztah složíme, konkrétně o tom, že *Dívka* je antecedentem zájmena *který*, a o tom, že *který* je objektem slovesa *potkal*.

```
...
CONNECT 1 3 Desc: Rel, Dom: L
CONNECT 3 4 Desc: Obj, Dom: R
```

Druhý zmíněný vztah užijeme i sám o sobě (otcem zájmena *který* má zůstat sloveso *potkal* i ve finálním stromě), první vztah však musíme zkombinovat s druhým, abychom dosáhli správného zavěšení vztažného podstromu. To se nám opět podaří pomocí ověření analytických funkcí – zadefinujeme, že pokud má slovo dva potenciální otce, z toho jednoho ve vztahu *Rel* (z čehož už poznáme, že dané slovo je vztažné zájmeno a tento otec je jeho antecedentem), potom druhý (a případně další) potenciální otec dostane přednost a vztah *Rel* s příslušným otcem převezme, stane se tedy sám dítětem antecedentu vztažného zájmena. Celá struktura souvětí se tedy vybuduje korektně dle obrázku a definice uvedené výše.

Popsaný jev mimo jiné zdůvodňuje existenci méně bezpečných pravidel. I když díky sofistikovaným pravidlům Tomáše Jelínka odhalíme antecedent i ve velmi zapeklitých případech, nebude nám to nic platné, pokud k tomu nebudeme mít informace o otcovství vztažného zájmena ze strany slovesa. Tehdy správnou strukturu vybudovat nemůžeme, nemůžeme se o to dokonce ani pokusit. Proto má smysl příslušné sloveso pomocí hrubšího pravidla (s náležitě sníženou prioritou) alespoň odhadnout, když už ho nemáme možnost určit s jistotou.

4.4.3 Ostatní

Krom výše podrobně popsaných transformací jsme implementovali dalších asi třicet – většina se týká různých typů koordinací a složitějších přísudků. Transformace se provádějí v cyklu, dokud je co provádět. Teprve poté dojde k závěrečnému zjednoznačnění, tj. že pokud některému z větných členů stále zůstává více než jeden otec, vybereme toho s nejvyšší prioritou. Je-li takových více, můžeme se rozhodnout, jak s tím naložíme – vybrat náhodného otce, hlásit chybu, vydat více výsledků apod.; každopádně změna chování systému na tomto místě je opravdu jednoduchá, proto není problém současnou „strategii jediného stromu (lesa)“ kdykoli nahradit jinou. Průběžně i na závěr kontrolujeme, zda ve stromě nevznikají cykly.

4.5 Evaluace

Ačkoli ze skeptického tónu, jímž je prostoupen předchozí text, si čtenář možná udělal předběžný závěr, že náš systém je schopen správně zpracovat právě a jen věty uvedené zde jako příklady, není tomu tak. Výsledky jsou sice slabší, nikoli však přímo ostudné.

Abychom mohli zhodnotit úspěšnost, musíme nejprve upřesnit, co od systému chceme. Jinak bude výsledek vypadat, pokud chceme jen výstup nejbezpečnějších pravidel, a jinak, pokud požadujeme celý strom, jehož souvislosti ovšem musíme napomoci tím, že uzly, o kterých skutečně nic nevíme, zavěsíme nějakým triviálním způsobem.

Pro kolektivní vyhodnocení všech typů výstupů však naštěstí stačí vydat pouze jeden výsledek, při jehož vytváření uplatníme všechny typy záchytných akcí. U každé závislosti zůstáva totiž uvedena její priorita, proto můžeme při vyhodnocování jednotlivých strategií snadno odfiltrovat výsledky pod přípustnou hranicí bezpečnosti.

Testování jsme prováděli na datech PDT 1.0 d-test, z nichž první polovinu jsme použili pro průběžné ladění, druhou polovinu pro závěrečné přeměření. Celková úspěšnost při vybudování souvislého stromu je jen pomocným ukazatelem (neboť souvislý strom není primárním cílem našeho programu), zajímalo nás zejména, jak rozsáhlé je uplatnění pravidel různých priorit (tedy kolik % dat se podařilo zpracovat pravidly určité priority) a úspěšnost přiřazení otce na těchto datech.

První řádky obou tabulek uvádějí celkovou úspěšnost, v dalších řádcích je uveden podíl na datech (pokrytí) a úspěšnost na těchto datech pro jednotlivé úrovně bezpečnosti pravidel. Výsledky jsou porovnány s výsledky dvou tehdy nejlepších statistických parserů [11] [3] na týchž datech.

Úspěšnost na vývojových datech:

	Pokrytí	Pravidla	Charniak	Collins
Všechna data	100 %	66.38 %	85.18 %	83.31 %
Velmi bezpečná	36.04 %	**94.15 %**	94.64 %	94.08 %
Bezpečná	23.47 %	80.81 %	87.29 %	85.60 %
Nebezpečná	21.96 %	50.26 %	79.94 %	77.67 %
Žádná	18.51 %	13.12 %	70.32 %	66.13 %

Úspěšnost na testovacích datech:

	Pokrytí	Pravidla	Charniak	Collins
Všechna data	100 %	65.37 %	84.24 %	82.29 %
Velmi bezpečná	35.64 %	**93.35 %**	93.65 %	93.07 %
Bezpečná	22.69 %	80.42 %	86.72 %	85.26 %
Nebezpečná	21.45 %	50.63 %	79.86 %	77.73 %
Žádná	20.20 %	14.79 %	69.39 %	64.78 %

Jak jsme již dříve uvedli, *velmi bezpečná* pravidla jsou ta, která vycházejí přímo z disambiguačních pravidel, *bezpečná* byla psána přímo pro syntax a nejsou tak detailně propracovaná, *nebezpečná* jsou záchrany typu „mým otcem je nejbližší sloveso“, v případě neuplatnění žádných pravidel jsme otcem uzlu učinili jeho pravého souseda (s levým sousedem byla úspěšnost téměř stejná, jen nepatrně nižší).

Je vidno, že výsledky velmi bezpečných pravidel jsou srovnatelné s výsledky statistických parserů (bohužel ne lepší). Ostatní skupiny pravidel již za statistickými parsery velmi viditelně zaostávají, což se ovšem dalo očekávat. Jejich výsledky nás tolik nezajímají, neboť od začátku bylo cílem nikoli stavět úplný parser, nýbrž získat (a to právě pomocí velmi bezpečných pravidel) alespoň nějaké informace o syntaxi věty, ovšem pokud možno lepší, než získá jakákoli jiná metoda.

Takže ta statistika fakt šetří lidem práci!

Miroslav Spousta

Základní vady výsledku velmi bezpečných pravidel jsou dvě, vzájemně provázané: malé pokrytí a nízká úspěšnost. Předpokládáme, že zvýšit úspěšnost 93 % resp. 94 % by šlo už opravdu velmi ztuha (z principu: homonymie konstrukcí, občasná nekonzistence ruční anotace apod.), proto jsme naše snahy upínali ke zvýšení pokrytí. Snadno totiž nahlédneme, že úspěšnost 93 %, nezajímavá na 35 % dat, kde „to umí každý“, by už mohla být velmi zajímavá, pokud bychom jí dosáhli třeba na 60 % dat.

Bohužel veškeré naše snažení v tomto směru zcela selhalo – i v samotné disambiguaci je hlavním a kritickým problémem nízké uplatnění pravidel (podrobnosti v kapitole o kombinovaných metodách značkování), proto není překvapením, že se projevilo i zde. Nemá-li disambiguační pravidlo, ať už jakkoli kvalitní a přesné, možnost se

uplatnit, zejména kvůli příliš homonymnímu kontextu a cyklické závislosti, nemůže se uplatnit ani jeho syntaktická část. Tento problém nelze vyřešit uvnitř syntaktického rozšíření, neboť tkví jinde – přímo v systému disambiguačních pravidel.

V rámci kombinovaných metod značkování se nám podařilo nalézt způsob, jak mohou taggery pravidlům předzpracovat vstup tak, aby se mohla uplatnit co nejvíce, naskýtá se tedy samozřejmě otázka, zda by stejná metoda nemohla pomoci i zde. Domníváme se, že ne, a krom toho jsme to samozřejmě zkoušeli, nikoli ovšem s hybridní metodou, kterou jsme v té době ještě neměli, nýbrž pouze s jedním taggerem. Konkrétně jsme tedy syntaktické zpracování pustili na datech morfologicky disambiguovaných taggerem Morče, výsledky však byly téměř stejné (nepatrně horší) jako na datech přímo z morfologie, pokrytí je tedy i v tomto případě velmi nízké (krom toho, že se jistě projevily i další slabiny tohoto postupu, zejména chyby taggeru).

4.6 Shrnutí

V předchozím textu bylo již uvedeno několik důvodů, proč náš postup selhal. Zde je shrneme a navrhneme, jak by se celý úkol případně mohl řešit jinak a lépe.

4.6.1 Nevyhovující návrh

Náš návrh systému, který vycházel z úmyslu co nejvíce využít a co nejméně měnit dostupná pravidla a programové nástroje, je příliš omezující, neumožňuje prolínání lingvistického zpracování textu a stavění syntaktické struktury. Toto omezení nejen způsobuje chyby, které byly popsány v oddílu o transformacích, ale také znemožňuje rozpoznat vztahy, jejichž členy jsou od sebe „příliš daleko“. V případě věty *Pes, který štěká, nekouše.* není žádné pravidlo tak odvážné, aby hledalo shodu přes jistý separátor klauzí (čárku před vztažným zájmenem), proto neodhalíme syntaktický vztah mezi *pes* a *nekouše*. Kdybychom měli možnost vztažnou větu po jejím úplném analyzování a zavěšení na antecedent pro účely dalšího zpracování vypustit a analyzovat pouze torzo původního souvětí (*Pes nekouše.*), napomůže to nejen syntaxi, ale i původní disambiguaci (úspěšně se provede unifikace podmětu a přísudku v příslušných morfologických kategoriích).

Tuto funkcionalitu by bylo možné nasimulovat cyklickým pouštěním LanGRu a programu pro stavění stromů (ten by zároveň obstaral i zmíněné vypouštění, LanGR by tedy dostával na vstupu postupně redukované věty). Zatím však nemělo valný smysl něco takového zkoušet, protože vzhledem k nízkému uplatnění pravidel je velmi nízký i počet úplně a bezpečně analyzovaných vypustitelných konstrukcí, nehledě na teoretické problémy s úlohou spojené (haplologie čárek apod.).

4.6.2 Nevyhovující systém pravidel

I u samotné morfologické disambiguace se již ukázalo, že systém dosud napsaných pravidel není valně přínosný při zpracování nepředzpracovaného (tedy plně morfologicky označkovaného) textu. Je-li naším cílem úplná disambiguace, jsou výsledky statistických taggerů na celé morfologické nabídce jen o málo horší než na datech předzpracovaných pravidly (pravidla tedy, i přes značné zvýšení precision, nezpracují nic „zajímavého“, pouze to, co by tagger ve velké většině případů dokázal i bez nich). Stejná situace je i v syntaxi: pravidla (jsou-li používána pozitivně, tedy k budování struktury) správně

odhalí mnoho vztahů, ale tytéž vztahy a se stejnou úspěšností odhalí i kvalitní statistický parser.

S tímto problémem se asi můžeme vyrovnat pouze zásadním přeformulováním úlohy. Ověřili jsme, že v samostatném budování struktury jsou pravidla slabá a nepřinášejí nic nového, mohla by však být užitečná při ověřování a upravování výsledků získaných jinými metodami (zakazování či potvrzování jednotlivých hran apod.). Je však bohužel jen málo parserů, které si nechají do své práce takto zasahovat, a jak jsme již uvedli na začátku kapitoly, v době, kdy syntaktické rozšíření vznikalo, jsme žádný parser s touto možností (a zároveň dostatečně úspěšný) k dispozici neměli.

Ve světle později dosažených výsledků na poli kombinovaných metod značkování se však domníváme, že nejnadějnější by bylo pokusit se implementovat analogii k postupu použitému v tamním „vítězném“ modelu, tedy redukovat (zejména negativními) pravidly množinu výsledků více parserů.

Kapitola 5

Závěr

Hlavním výsledkem popsaným v této práci jsou nové metody kombinace nástrojů pro morfologickou disambiguaci. Krom „vítězné“ metody převyšující svou úspěšností o více než 0.5 % nejúspěšnější samostatný tagger (redukce chyby 11.48 %) bylo představeno i několik dalších kombinovaných metod, ne již tolik úspěšných, leč stále výrazně úspěšnějších, než jsou jejich jednotlivé komponenty, jakož i jiné metody doposud známé, a zároveň majících nižší výpočetní nároky než metoda vítězná.

Zásluha autorky práce na celkovém výsledku je samozřejmě jen částečná, pro úspěch celku je zásadní funkčnost a úspěšnost jednotlivých komponent, které jsou každá dílem jiného autora (v případě pravidlové disambiguace kolektivu autorů). Na druhou stranu metody, které se nakonec v kombinování osvědčily nejvíce, nejsou zcela přímočaré a cesta k závěrečnému výsledku vyžadovala mnoho nápadů a experimentů.

Zbylé dva výsledky této práce nejsou nikterak průlomové, ovšem s hlavním tématem taktéž souvisejí a jsou výlučným dílem autorčiným.

V prvním případě jde o vytvoření nového datového zdroje s využitím zdroje již existujícího (vytvoření valenčního slovníku deverbativních adjektiv z valenčního slovníku sloves), který je primárně určen pro užití v projektu pravidlové disambiguace, je však natolik univerzální, že jej může využít celá lingvistická komunita. Vedlejším efektem tvorby slovníku je podrobné rozdělení sloves do vzorů pro odvozování deverbativních adjektiv, v této šíři dosud nikým neprovedené. I tohoto výsledku lze dále využít, jak v teoretickém výzkumu, tak v praxi, např. v morfologických slovnících.

Posledním tématem, kterému se v této práci věnujeme, je experiment s rozšířením disambiguačních pravidel do oblasti syntaxe. Experiment nebyl úspěšný (nepodařilo se s jeho pomocí zlepšit dosavadní nejlepší výsledek v parsingu češtiny), může však být dobrým výchozím bodem pro toho, kdo by se o něco podobného, tedy (částečný) parsing na základě pravidel, chtěl pokusit v budoucnu.

English summary

Combining Statistical and Rule-Based Approaches to Morphological Tagging of Czech Texts (Formal description of Czech sentences and problems concerning its implementation)

The thesis consists of three parts, which are all related to a rule-based morphological disambiguation project [22].

The first part (Chapter 2) describes conversion of a surface valency lexicon of Czech verbs to a surface valency lexicon of adjectives that can be derived from these verbs and that use their (possibly modified) valency frames. After preparing the necessary data by hand, the conversion can be fully automatic and every change of the source lexicon can be automatically reflected in the destination lexicon. We have successfully converted the verb valency lexicon "Brief" with about 15,000 verbs to a valency lexicon of about 27,000 deverbal adjectives. In the chapter, there are also described some interesting peculiarities in the process of creating passive adjectives and their valency frames.

The second part (Chapter 3) contains the main result of the thesis. Several hybrid disambiguation methods are described which combine the strength of hand-written disambiguation rules and three different statistical taggers (HMM, Maximum-Entropy and Averaged Perceptron). The results of the hybrid system are better than any other method tried for Czech tagging so far.

The third part (Chapter 4) describes an extension to a rule-based morphological disambiguation system, which makes the disambiguation rules usable for shallow parsing. The system is not very successful, but some ideas discussed in the thesis can be possibly used in the future to improve the performance of statistical parsers.

Literatura

[1] Český národní korpus
<http://ucnk.ff.cuni.cz>

[2] Michael Collins: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In: Proceedings of EMNLP. (2002)

[3] Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, Christoph Tillmann: A Statistical Parser of Czech. In: Proceedings of the 37th Meeting of the ACL, 505–512. University of Maryland, College Park, Maryland. (1999)

[4] Drahomíra „johanka" Doležalová, Tomáš Jelínek: Získávání informací o povrchové valenci sloves a adjektiv z ČNK. In: Proceedings of Grammar and Corpora, Praha 2005, v tisku.

[5] Jan Hajič: Disambiguation of Rich Inflection (Computational Morphology of Czech). Vol. 1. Karolinum Charles University Press. Prague. (2004)

[6] Jan Hajič, Pavel Krbec, Pavel Květoň, Karel Oliva and Vladimír Petkevič: Serial Combination of Rules and Statistics: A Case Study in Czech Tagging. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics. CNRS – Institut de Recherche en Informatique de Toulouse and Université des Sciences Sociales, Toulouse, pp. 260–267. (2001)

[7] Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová: Prague Dependency Treebank v2.0. CD-ROM. Linguistic Data Consortium, Cat. LDC2006T01. Philadelphia, PA, USA. ISBN 1-58563-370-4. (2006)
<http://ufal.ms.mff.cuni.cz/pdt2.0>

[8] Bohuslav Havránek, Alois Jelička: Česká mluvnice. SPN Praha. (1986)

[9] Barbora Hladká: Czech Language Tagging. PhD thesis, UK MFF. (2000)

[10] Aleš Horák: Verb Valency and Semantic Classification of Verbs. In: Proceedings of Text, Speech and Dialogue Brno, pp. 61–66. (1998)

[11] Eugene Charniak: A Maximum-Entropy-Inspired Parser. In: Proceedings of NAACL. Seattle, Washington. (2000)

[12] Frederick Jelinek: Statistical Methods for Speech Recognition. The MIT Press. Cambridge, Massachussetts. (1998)

[13] Fred Karlsson and Atro Voutilainen and Juha Heikkilä and Arto Anttila (eds.): Constraint Grammar: a language-independent system for parsing unrestricted text. Natural Language Processing. Vol. 4, Mouton de Gruyter, Berlin and New York. (1995)

[14] Marie Kopřivová: Valence českých adjektiv. NLN, Praha. (2006)

[15] Pavel Krbec: Language Modeling for Speech Recognition of Czech. PhD thesis, UK MFF. (2005)

[16] Pavel Květoň: Rule-based Morphological Disambiguation. PhD thesis, UK MFF. (2006)

[17] Petr Němec, Kiril Ribarov: Making the Good Taggers Even Better: Application of Artificial Neural Networks in Morphological Tagging of Czech. In: Proceedings of Language&Technology'05, pp. 85–89. (2005)

[18] Miroslav Nepil: Relational Rule Induction for Natural Language Disambiguation. PhD thesis, FI MU Brno. (2003)

[19] Karel Oliva: Linguistics-based PoS-tagging of Czech: Disambiguation of *se* as a Test Case. In: Proceedings of the 4th Conference on Formal Description of Slavic Languages held in Potsdam, pp. 299–314. (2001)

[20] Karel Oliva: On Retaining Ambiguity in Disambiguated Corpora. In: Traitement Automatique des Langues vol. 42 No. 2, Hermes Science Publications, Paris. (2001)

[21] Karel Oliva: Korpusová lingvistika pro lingvistické korpusy (Corpus Linguistics for Linguistic Corpora). In: Proceedings of the Workshop SLOVKO'03, Bratislava 2003, v tisku.

[22] Karel Oliva, Milena Hnátková, Pavel Květoň, Vladimír Petkevič: The Linguistic Basis of a Rule-Based Tagger of Czech. In: Proceedings of the Text, Speech and Dialogue Brno, pp. 3–8. LNAI 1902, Springer–Verlag. Berlin Heidelberg. (2000)

[23] Karel Oliva, Vladimír Petkevič: On the Need of *Linguistic* Linguistic Interpretation of Corpora. In: Book of Abstracts of the Meeting of Societas Linguistica Europea 2001, Leuven, August 2001. (2001)

[24] Karel Pala, Pavel Ševeček: Valence českých sloves. In: Sborník prací FFUB, Brno. (1997)

[25] Pavel Šmerk: K dorfologické desabiguaci češtiny. Teze disertační práce, FI MU Brno. (2007)

[26] Prague Dependecy Treebank v1.0
<http://ufal.mff.cuni.cz/pdt/Corpora/PDT_1.0>

[27] Drahomíra „johanka“ Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, Pavel Květoň: The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In: Proceedings of Balto-Slavonic Natural Language Processing Workshop, ACL 2007, Praha. (2007)

[28] Radek Sedláček: Morphemic Analyser for Czech. PhD thesis, FI MU Brno. (2004)

[29] Pasi Tapanainen and Atro Voutilainen. Tagging accurately: don't guess if you know. In: Proceedings of the 4th conference on Applied Natural Language Processing, pp. 47–52. Stuttgart. (1994)

[30] Jan Votrubec: Volba vhodné sady rysů pro morfologické značkování češtiny. Diplomová práce, UK MFF. (2005)