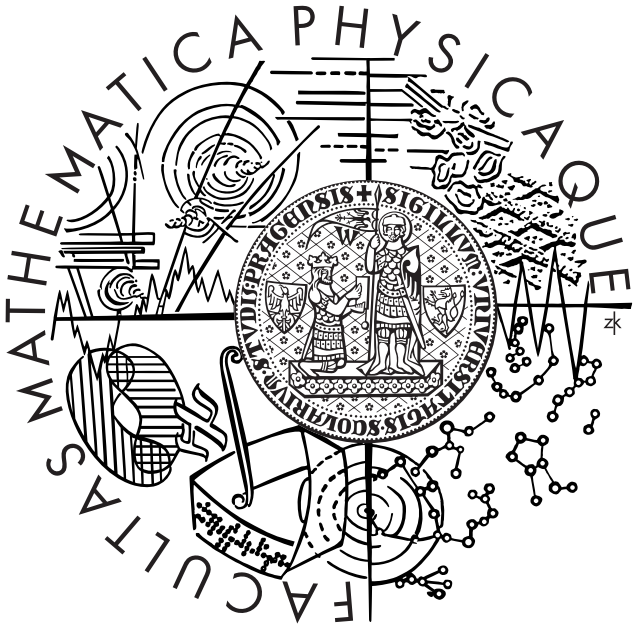
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Eduard Bejček

Automatické přiřazování významu – „Sense-tagging"

Ústav formální a aplikované lingvistiky

Vedoucí: RNDr. Markéta Lopatková, Ph.D.

Studijní program: Informatika
Studijní obor: Matematická lingvistika

Rád bych v prvé řadě poděkoval vedoucí RNDr. Markétě Lopatkové, Ph.D. za podrobné rady, pečlivé korektury a laskavou odbornou pomoc.

Dále osazenstvu IRC kanálu #jikos, kde mi poskytli mnohé cenné rady a ochotně pomohli vyřešit (převážně typografické) problémy.

Veliký dík patří mamince, tatínkovi a bratrovi, kteří v sobě nacházeli po celou dobu dostatek pochopení a obětavě mi poskytovali prostor, čímž mi umožnili práci dokončit.

Neméně chci poděkovat své přítelkyni, Haně Škaloudové, že stála trpělivě celou dobu při mně a podporovala mě.

Tuto diplomovou práci chci věnovat své babičce, Květě Komárkové, k jejím osmdesátým narozeninám.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 21. dubna 2006 Eduard Bejček

Obsah

Obsah **iii**

1 Úvod **1**
1.1 Seznámení s WSD, její význam a využití . . . 1
1.2 Zadání diplomové práce . . . 2
1.3 Vlastní práce . . . 3
1.3.1 Řešená úloha . . . 3
1.3.2 Členění práce . . . 3

2 Data a nástroje **5**
2.1 PDT . . . 5
2.1.1 Horizontální členění . . . 6
w-rovina . . . 7
m-rovina . . . 7
a-rovina . . . 7
t-rovina . . . 7
2.1.2 Vertikální členění . . . 9
2.2 Valence a PDT-VALLEX . . . 10
2.2.1 Valence . . . 10
2.2.2 Valenční slovník PDT-VALLEX . . . 10
2.3 WordNet . . . 11
2.3.1 Princetonský WordNet . . . 12
2.3.2 Euro WordNet . . . 13
2.4 btred, TrEd . . . 13
2.5 C4.5 . . . 14
2.6 Perl, Bash . . . 15
2.7 VisDic . . . 15

3 Příprava dat pro přiřazování synsetů **16**
3.1 Popis data . . . 16
3.1.1 Problém 1: Český Euro WordNet . . . 16
3.1.2 Problém 2: Data z PDT 1.0 . . . 17
1) Formát PDT 1.0 vs. PDT 2.0: . . . 18
2) t-rovina vs. a-rovina: . . . 26
3) Smísení dat PDT a ČWN . . . 28
3.2 Zpracování a namapování dat . . . 31
3.2.1 Získání vybraných synsetů (`get_senses_from_ewn-pdt.pl`) . . . 31
3.2.2 Vložení zvolených synsetů do PDT 2.0 (`map_senses_to_pdt.pl`) . . . 32
3.2.3 Přizpůsobení PML (`link_fake_data`) . . . 34

4 Automatické přiřazování synsetů **35**
4.1 Specifikace úlohy . . . 35
4.2 Charakteristiky dat . . . 36
4.2.1 Mezianotátorská shoda . . . 36
4.2.2 Statistický pohled na trénovací data . . . 37
4.2.3 Výpočet baseline . . . 41
4.2.4 Statistiky na testovacích i trénovacích datech . . . 42
4.3 Vlastní experiment . . . 44
4.3.1 Trocha teorie . . . 44
4.3.2 Konkrétní postup . . . 45
4.3.3 Zlepšování výsledků . . . 47
4.4 Evaluace a diskuse výsledku . . . 51
4.4.1 Testy na evaluačních datech . . . 51
4.4.2 Diskuse . . . 51
4.5 Krátká zmínka o rysech t-roviny . . . 52

5 Automatické přiřazování rámců **54**
5.1 Známé výsledky . . . 54
5.2 Specifikace úlohy . . . 54
5.3 Statistiky a baseline: . . . 55
5.4 Převzaté rysy . . . 55
5.5 Vlastní experiment, výsledky . . . 56

6 Závěry... **58**
6.1 Srovnání slovníků . . . 58
6.2 Budoucnost . . . 59

Literatura **61**

Seznam obrázků **63**

Seznam tabulek **64**

A Programátorská dokumentace **1**
A.1 Struktura CD 1
A.2 Jak udělat... 3

B Programátorská dokumentace – část slévání **4**
B.1 `get_senses_from_ewn-pdt.pl` 4
B.2 `map_senses_to_pdt.pl` 6
B.3 `link_fake_data` 7

C Programátorská dokumentace – část trénování **8**
C.1 `rysy_tred` 8
C.2 `get_features.btred` 10
C.3 `build_names.pl` 11
C.4 `classify.pl` 12
C.5 `results_process.pl` 13
C.6 `run` 14
C.7 `vypust_rysy.pl` 14

D Seznam používaných rysů **16**
D.1 ČWN, a-rovina 16
D.2 ČWN, t-rovina 20
D.3 PDT-VALLEX 20

Název práce: Automatické přiřazování významu – „Sense-tagging"
Autor: Eduard Bejček
Katedra: Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: RNDr. Markéta Lopatková, Ph.D.
e-mail vedoucí: Marketa.Lopatkova@mff.cuni.cz

Abstrakt: Tato práce se zabývá desambiguací významu substantiv a adjektiv. Významy rozlišuje dvojím způsobem, na základě synsetů z Českého WordNetu a podle valenčních rámců zachycených v PDT-VALLEXU. Tím vznikají dvě oddělené úlohy, spočívající v přiřazování synsetů/rámců víceznačným lemmatům. Řešíme je metodou strojového učení, konkrétně využíváme rozhodovací stromy. K disposici máme korpus PDT s bohatou anotací na více jazykových rovinách obsahující i valenční rámce a dále část PDT s ruční anotací pomocí ČWN. V první části práce se zabýváme spojením obou korpusů, neboť jsou v různých formátech. Ve druhé části s použitím syntaktických závislostí věty a morfologických informací přiřazujeme synsety z ČWN. Dosahujeme úspěšnosti 91,4 % (proti 90,0 % baseline) pro substantiva a 93,9 % (proti 93,2 %) pro adjektiva. To představuje opravení 13,7 % (resp. 10,0 %) chyb vůči baseline. Krátce tuto metodu aplikujeme také na t-rovinu, kde je však patrný nedostatek dat. Ve třetí části přiřazujeme valenční rámce substantivům na základě informací z hloubkové roviny významu. Dosahujeme úspěšnosti 86,9 % proti 84,9 % baseline. Na závěr porovnáváme oba použité slovníky.

Klíčová slova: WSD, WordNet, PDT-VALLEX, rozhodovací stromy, substantiva, adjektiva

Title: Automatic Word Sense Disambiguation
Author: Eduard Bejček
Department: Institute of Formal and Applied Linguistics
Supervisor: RNDr. Markéta Lopatková, Ph.D.
Supervisor's e-mail address: Marketa.Lopatkova@mff.cuni.cz

Abstract: This thesis deals with word sense disambiguation of nouns and adjectives. We make use of the two approaches to distinguish word senses: the first method is derived from synsets from the Czech WordNet, the second one uses valency frames from the PDT-VALLEX dictionary. This means we have the two separate tasks, the goal of which is to assign appropriate senses (synsets/frames) to ambiguous lemmas. We are employing machine learning methods, the most notable of which are the decision trees. For this purpose we are using two types of data, namely the PDT corpus with rich annotations on several language layers, and also the part of the PDT manually annotated using the CWN. In the first part we are trying to merge both corpora, since they are stored in the different formats. In the second part we are using syntactic and morphologic features to assign CWN synsets. We reach 91,4 % accuracy (compared to 90,0 % baseline) for nouns and 93,9 % (compared to 93,2 %) for adjectives. In other words this means that we can correct 13,7 % (10,0 % respectively) of errors made by the baseline. This method is applied on the t-layer, where lack of data is obvious. In the third part we are assigning valency frames to nouns using information from deep syntactic layer. The accuracy gained by us has grown to 86,9 % compared to 84,9 % baseline. At the very end, we compare the both dictionaries obtained.

Keywords: WSD, WordNet, PDT-VALLEX, decision trees, nouns, adjectives

Kapitola 1

Úvod

1.1 Seznámení s WSD, její význam a využití

Zkratka WSD (Word Sense Disambiguation, neboli desambiguace významu slov, zjednoznačnění) označuje oblast lingvistiky zabývající se významy slov z pohledu textu. Na rozdíl třeba od lexikografů, kteří rozdělují slova na významy z důvodů uspořádání slovníkových položek, lidé zabývající se WSD se snaží přiřazovat správné významy jednotlivým slovům vstupního (pokud možno přirozeného) textu, aby zpřesnily smysl slov a potažmo věty.

Máme za to, že tyto dva přístupy se principiálně odlišují a je otázka, zda je uskutečnitelná jejich kombinace. Při tvorbě (řekněme) velkého slovníku češtiny budeme pravděpodobně dbát na to, abychom zachytili všechny nuance, kterých může slovo nabývat, i jemné rozdíly významu, nečekaná spojení apod. Budeme počítat s tím, že ve slovníku bude člověk hledat, protože nějaký okrajový význam nezná. Když to bude jemně roztříděno, tedy když tam bude zmínka třeba o všech dvaceti „podvýznamech“, budeme snad spokojeni. Pak ale k tomuto hotovému slovníku přijdeme a budeme se snažit podle něho rozlišovat významy ve skutečném textu. Může se náhle ukázat, že je nelze vůbec jednoznačně přiřadit. Tak ačkoli stále budeme vědět, že slovo má původních dvacet významů, pro účely desambiguace mnoho z nich sloučíme zpátky, až se dostaneme ke stavu, kdy významů bude málo, ale budeme téměř vždy schopni rozhodnout, o který z nich se v textu jedná.

Existuje řada přístupů: význam lze určovat přiřazením položky slovníku, nebo klasifikací slov do skupin vytvořených automaticky (na základě nejmenší vzdálenosti podle nějaké metriky v textu); postupovat při tom lze na základě ručně vytvořených a lingvisticky podložených pravidel, nebo statisticky; existuje také nepřeberné množství možností, jakou informaci využívat k určení významu: sledovat a určovat kontext celého článku (a předpokládat spolu s [4], že se nezmění), dívat se na lemmata jen v nejbližším okolí slova, sledovat větnou strukturu, nebo naopak morfologické informace – a nebo všechno dohromady. My budeme mít k disposici slovník s jednotlivými významy a přiřazovat je budeme pomocí metody strojového učení. Zaměříme se zejména na informace z morfologie.

WSD tedy obecně představuje úlohu, která rozhoduje o významech jednotlivých slov ze vstupního (potenciálně už předzpracovaného, označkovaného) textu. Její úspěšnost se dá hodnotit proti vzorku dat, která byla opatřena významy ručně. Ani to však není zaručeně správné, protože ani člověk si v oblasti významu není vždy jistý.

Význam[1] WSD je poměrně široký. Kromě toho, že by nás úspěšné počítačové rozpoznávání významu posunulo o kousek blíže ke skutečně inteligentním strojům, rozumějícím lidskému jazyku, má tato disciplína také mnoho časově bližších, dostupnějších a v praxi již teď lépe využitelných dopadů. Bez správného pochopení významu se například neobejdou žádné počítačové překlady mezi jazyky. (Jak bez znalosti významu slova „práce“ rozhodneme, zda ho překládat do angličtiny jako „work“, „business“, „labour“, nebo jako „job“?) Také velice užitečné by bylo pochopení významu slov, potažmo celého článku, pro pokročilé vyhledávání v textech. (Oč lepší by byl dotaz nějakému vyhledávači „svíčka; oblast: automobilismus“, než ze všech nalezených článků ručně vyřazovat ty o voskových svíčkách. Na to je potřeba prohledávané texty kategorisovat podle jejich obsahu, a to nejlépe automaticky.)

1.2 Zadání diplomové práce

Cílem naší práce by mělo být automatické přiřazování slovníkových položek jednotlivým výskytům slov, zejména substantiv a sloves. Jedná se o úlohu ze skupiny „word sense disambiguation“. Naším úkolem bylo navrhnout nástroj, který pro jednotlivé výskyty slov v textu určí, o kterou položku zadaného slovníku jde.

Měli jsme k disposici dva velice odlišné slovníky. Prvním slovníkem byl Český WordNet (se všemi pojmy, podklady a nástroji seznámíme později) založený na synsetech. Druhým byl valenční slovník PDT-VALLEX, kde kritériem významu je valenční rámec.

Dále jsme měli ručně anotovaný korpus PDT. Jeho jedna část má přiřazeny valenční rámce PDT-VALLEXu, druhá (částečně se protínající s první) pak synsety Českého WordNetu.

Úkolem bylo otestovat a porovnat oba přístupy k významu a implementovat samotné automatické značkování. K tomu jsme využili existující nástroje, zejména C4.5 zajišťující trénování rozhodovacích stromů.

[1] Zde se ukazuje, že samo slovo význam má více významů. Zde máme na mysli význam jako účel, použití, dopad, rozsah využití.

1.3 Vlastní práce

1.3.1 Řešená úloha

V této práci se zaměříme více na substantiva a adjektiva. Ukázalo se, že pro slovesa je úloha již řešena ([2]).

Naši práci zahájíme úlohou přiřazování synsetů, tedy částí související s WordNetem.[2] Vyřešíme problém namapování dvou různých formátů dat, abychom získali rysy pro trénování. Tento problém se ukáže jako poměrně rozsáhlý.

Zaměříme na substantiva a adjektiva. Zkusíme nalézt vhodnou sadu rysů, díky které bychom v úspěšnosti překonali poměrně vysoko nasazenou baseline. Budeme pracovat převážně na a-rovině PDT, některé testy jsme však provedeme i na t-rovině: a ukážeme na nich nedostatečné množství dat.

Mnohem méně pozornosti budeme věnovat druhé části úlohy, tedy přiřazování valenčních rámců. I v této části budeme postupovat obdobným způsobem a využijeme předchozích výsledků, ale také v odborném textu publikovaných rysů pro a-rovinu a pokusíme se je využít na t-rovině.

Na závěr přineseme shrnutí experimentů, srovnání obou přístupů rozlišujících význam (tedy podle valenčního rámce na straně jedné a podle synonymie, hyponymie a hyperonymie na straně druhé) a zmíníme možnosti dalšího výzkumu, kterých jsme si při naší práci povšimli, ale nemohli se jim dostatečně věnovat.

1.3.2 Členění práce

Seznámili jsme se s obecnou formulací WSD a také se zadáním této diplomové práce. Na následujících řádkách si prohlédneme, z čeho sestává.

V **druhé** kapitole se seznámíme nejprve se všemi daty (korpusem a slovníky), která budeme používat, podíváme se, na jakých základech jsou vystavěny a co přinášejí. Ve zbytku vyjmenujeme nástroje, které jsme během práce používali, a se zajímavějšími se blíže seznámíme.

Třetí kapitola začíná prozkoumávat data určená k automatickému přiřazování synsetů. Vzhledem k problémům, které s sebou přinesla změna formátu, se celá kapitola věnuje namapování dvou podstatných složek dat na sebe, slití v jeden celek. Po přesném popsání problému, při němž se nevyhneme ukázkám obou dvou formátů, se seznámíme s jeho řešením.

Hlavní experiment, na a-rovině, přinášíme v kapitole **čtvrté**. Začneme rozborem dat, která máme k disposici, a pomocí několika číselných a grafických charakteristik si popíšeme jejich povahu a vlastnosti. Seznámíme se s lidskou mezianotátorskou shodou a spočítáme

[2] Úlohou WSD pomocí WordNetu se zabýval například Senseval 3: `http://www.senseval.org`.

jednoduchou baseline jako základ pro naše pokusy. V části 4.3.2 přinášíme vlastní experiment a teorii týkající se zvolené metody – rozhodovacích stromů. Kapitolu zakončíme obdobným pokusem na t-rovině.

Pátou kapitolu věnujeme druhému důležitému experimentu, přiřazování valenčních rámců. Věnovali jsme jí méně pozornosti a nezabírá také mnoho stran.

V **závěru** shrneme celou práci, zvolené postupy i dosažené výsledky. Zamyslíme se nad rozdíly mezi dvěma používanými slovníky a zdůrazníme výhody a nevýhody každého. Na úplný závěr předestřeme vyhlídky do budoucna a možnosti dalšího postupu.

V celém textu využíváme tohoto menšího fontu k zařazení méně důležitých poznámek k textu, vysvětlení okrajových pojmů, k diskusím některých rozhodnutí, zamyšlením apod.

Kapitola 2

Data a nástroje

Na začátku bychom rádi čtenáře seznámili s prostředím, ve kterém tato práce vznikala. Tedy s nástroji, které jsme k ní používali, a s daty, nad kterými jsme prováděli veškeré experimenty.

Všechny programy jsme vyvíjeli v prostředí Linuxu ve skriptovacích jazycích btred (kap. 2.4), Perl a GNU Bash (s využitím standardních linuxových balíků, kap. 2.6). Skripty[1] jsou na přiloženém CD. Pro účely strojového učení rozhodovacích stromů jsme používali program C4.5 (kap. 2.5). K prohlížení slovníku WordNet jsme používali nástroj VisDic (kap. 2.7) vyvinutý na Masarykově univerzitě v Brně.

Základem celé práce bylo poměrně čerstvě vydané PDT 2.0 Beta a česká verse WordNetu, proto začněme s nimi.

2.1 PDT

Stručně řečeno, PDT je korpus češtiny anotovaný na několika rovinách representace. PDT (Prague Dependency Treebank, [7]) je posledních deset let vyvíjen na Ústavu formální a aplikované lingvistiky (ÚFAL) Matematicko-fyzikální fakulty University Karlovy. Vychází z funkčně generativního popisu (FGD, Functional Generative Description), který navazuje na strukturalistické tradice Pražského lingvistického kroužku. Vznikl jako snaha o ověření, rozšiřování a zjemňování teoretického popisu v rámci FGD a jako praktické využití této teorie. Již v dnešní podobě je PDT bohatým zdrojem dat hojně anotovaným v nejrůznějších oblastech lingvistiky. Jako takový může sloužit a slouží jako podklad pro mnoho dalších lingvistických studií (ověřování, vyhledávání) a na jeho základě je možné vyvíjet nástroje pro

[1] Upozornění: Všechny skripty byly psány pro naši potřebu. Při dodržení všech vstupních podmínek popsaných v této práci (zejména přílohy B až C) pracují správně. Jiná data a jiné použití nebylo předpokládáno a chování skriptů není v takových případech testováno.

zpracování přirozeného jazyka (NLP, Natural Language Processing). Stát se jedním z dílčích kroků si klade za cíl i tato práce.

Nejprve si řekněme několik vět o FGD. Jde o popis jazyka a vůbec přístup k němu, který v šedesátých letech minulého století navrhl Petr Sgall [17]. V pozdějších letech byl jím i jeho spolupracovníky dále rozvíjen. Jeho dnešní podoba stojí na třech základních pilířích:

- **závislostní syntax**
- **tektogramatická rovina** (hloubková struktura věty, nesoucí význam; valence slov; koreference, ...)
- **aktuální členění** (formální popis informační struktury věty: rozdělení výpovědi na to, *o čem* se mluví, a to, *co se o tom* říká)

V polovině devadesátých let se začal projekt, který si vytkl cíl aplikovat FGD na skutečná data. K tomuto účelu se použila část (2 miliony tokenů) Českého národního korpusu (ČNK[2]) a rozběhla se rozsáhlá anotace, která trvala 10 let a vedla k morfologickému označkování všech 2 milionů tokenů, k analytické representaci 1,5 milionu a na podstatné tektogramatické rovině bylo zpracováno 800 000 tokenů. Anotace si také vyžádala určité ústupky a technická řešení, která v čisté teorii FGD nebyla přítomna. V roce 2001 byl zveřejněn PDT 1.0 (ještě bez tektogramatické representace) a v roce 2005 pak PDT 2.0 Beta, jenž jsme používali v naší práci (dále o něm budeme mluvit jen jako o PDT 2.0, či prostě PDT, nebude-li potřeba ho odlišovat od předchozí verse).

2.1.1 Horizontální členění

Korpus PDT je značkován na třech rovinách:

m-rovina odpovídá morfematické rovině FGD;

a-rovina je pomocná analytická rovina vytvořená navíc a zachycuje povrchovou syntax věty;

t-rovina odpovídá tektogramatické rovině FGD.

My se tohoto značení přidržíme – a také pro ostatní termíny v této práci použité platí, že se vztahují k tomu, jak jsou realisované v PDT. Kromě těchto tří rovin musela být zavedena ještě čtvrtá:

w-rovina je rovinou slovní; byla zavedena do PDT, protože teorie nepočítala s chybami textu.

[2]Homepage: `http://ucnk.ff.cuni.cz/`

Teď podrobněji k rovinám:

w-rovina

Jak jsme naznačili, tato rovina obsahuje přesně zachovaný původní text, bez všech oprav. Je pouze tokenisovaný na slova a na interpunkci (aby na ně bylo možno odkazovat) a jsou označené hranice vět a odstavců.

Tvorba korpusu probíhala od spodních rovin k vyšším. V současném stavu nejsou všechna data anotována na všech rovinách (vyšší roviny tedy pro část dat chybí). Důvodem není jen časová náročnost, ale také fakt, že při určité práci s daty vyšší roviny je nutné mít *více* dat roviny nižší. Plná anotace vyšší roviny by tedy stejně nemohla být vždy využita. Vždy platí, že konkrétní věta anotovaná na určité rovině je anotována i na všech rovinách nižších. Nejen proto mají w-rovinu přirozeně všechna data.

m-rovina

Tato rovina obsahuje morfologickou informaci. Každý token má přiřazen svůj (opravený) slovní tvar (a případnou informaci o opravě), svůj lemmatisovaný tvar. Hlavně však je označkován svým morfologickým tagem (viz [6, 8]), obsahujícím tvaroslovné informace jako slovní druh, pád, čas, rod, apod. Celá rovina byla ručně zkontrolována. Také m-rovina ještě přísluší všem datům.

a-rovina

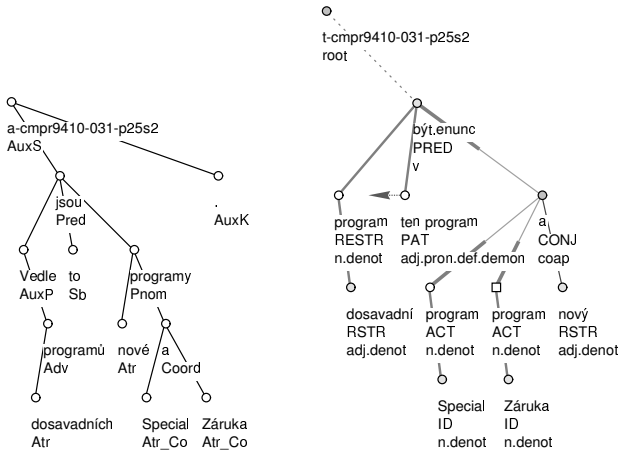
Rovina analytická, nebo též syntaktická, je rovinou technickou, slouží pouze k překlenutí propasti mezi morfologickou a tektogramatickou rovinou. Protože je svým způsobem „na půli cesty“, využila se při tvorbě PDT.

Tato rovina má (na rozdíl od předchozích dvou lineárních) strukturu stromovou. Obsahuje anotační informace o větných členech a jejich závislostech. Zachycuje větnou syntax. Na obrázku 2.1 vidíme příklad, jak se zobrazují některé informace a-roviny v programu TrEd (kapitola 2.4).

Na úrovní a-roviny je anotováno zhruba $^3/_4$ všech vět, tedy 1,5 milionu tokenů.

t-rovina

T-rovina byla nejnáročnější na zpracování (40 % všech vět, 800 tisíc tokenů), ale právě v ní tkví také největší přínos PDT. Jejím základem je také stromová struktura, ale neodpovídá té z a-roviny: representuje *hloubkovou* syntax věty. Obsahuje funktory, gramatémy, aktuální

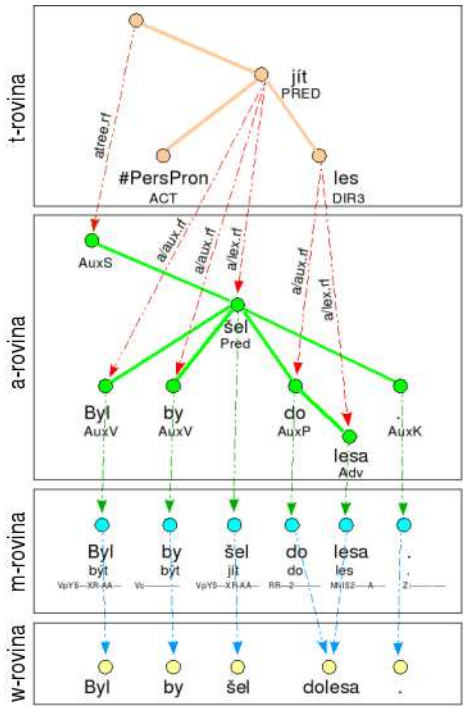


Obrázek 2.1: Zachycení věty „Vedle dosavadních programů jsou to nové programy Special a Záruka.“ na a-rovině a t-rovině PDT

členění (topic/focus), slovesné rámce. Mimo to obsahuje ale také informace o koreferenčních vztazích, což narušuje stromové vlastnosti. Ukázka je na obrázku 2.1.

Mezi všemi rovinami je zachována informace o tom, který uzel odpovídá kterému na jiné rovině. (To i přesto, že t-rovina může mít jiný počet uzlů, neboť všechna „významově nepodstatná“ slova jsou odstraňována, případně změněna na atributy slov autosémantických, dochází ke slučování a naopak i ke vzniku nových generovaných uzlů zastupujících na povrchu vypuštěná vyjádření apod.)

Velmi názorně a stručně je vztah a provázanost mezi rovinami vidět na obrázku 2.2.



Obrázek 2.2: Propojení rovin PDT (převzato z [7])

2.1.2 Vertikální členění

Napříč rovinám representace, zmíněným v předchozí kapitole, je PDT pro další použití při strojovém učení a podobných úlohách rozděleno (můžeme si to představit kolmo k rovinám, tedy vertikálně) na tři skupiny: na trénovací, testovací a evaluační data. V těchto množinách je rozložení m-, a- a t-roviny stejné jako v datech celkových. Množiny jsou rozděleny následovně: osm dílů do trénovacích dat, jeden díl do testovacích a jeden díl do evaluačních.

Podle tohoto dvojího členění jsou data rozdělena již na CD obsahujícím PDT. Jsou tam adresáře `mw/`, `amw/` a `tamw/` podle jednotlivých rovin, kam až dostoupila anotace. A v každém z nich pak adresáře `train-1/` až `train-8/`, `dtest/` a `etest/`.

Na závěr uvedeme webové stránky, které o uvedeném pojednávají podrobněji: `http://ufal.mff.cuni.cz/pdt2.0/index-cz.html`. Na CD je zveřejněna jen ukázka dat.

K manipulaci s PDT jsme používali nástroje TrEd (prohlížení) a btred (vyhledávání), více v kapitole 2.4.

2.2 Valence a PDT-VALLEX

Slovník PDT-VALLEX je valenční slovník, je tedy namístě vysvětlit si tento pojem.

2.2.1 Valence

Valenční teorie je součástí FGD. Popisuje syntaktické okolí slovesa,[3] povinná i volitelná doplnění, jejich formu (pád a případně předložku) a jejich „význam“ – funktor. Každému slovesu lze přiřadit jednu nebo několik variant této valenční informace – valenčních rámců. Například můžeme „položit něco někam“, ale budeme-li pokládat otázku, můžeme ji položit nanejvýše někomu, ale ne někam. Takto se tedy dělí sloveso podle své valence. Navíc ona věc/otázka, kterou pokládáme, je v obou případech obligatorní.

Lépe to vysvětlíme již konkrétně na příkladu z valenčního slovníku.

Obrázek 2.3: Sloveso „krýt“, „krýt se“ a substantivum „krytí“ v PDT-VALLEXu

2.2.2 Valenční slovník PDT-VALLEX

Jak přesně vypadá struktura PDT-VALLEXu a co vše se u valence zaznamenává si vysvětlíme na příkladu slovesa „krýt“, obrázek 2.3. Položka slovníku sestává z hlavního lemmatu („krýt“) a z popisu čtyř valenčních rámců. Přibližně platí, že valenční rámec odpovídá významu: že pokud se dva rámce od sebe liší syntakticky, budou se lišit i sémanticky. Proto se dá

[3] Valence je vlastností autosémantický slov, detailně je popsána primárně u sloves. Nověji je zpracovávána též pro substantiva a adjektiva, zejména pro ta, která mají slovesná rozvití.

o přiřazování rámců mluvit jako o přiřazování významu. Ze stejného důvodu je každý rámec opatřen příkladem, který pomáhá lépe pochopit význam („krýt bankovky“ proti „brnění ho krylo“). Obsahem valenčního rámce jsou valenční doplnění. Každé doplnění je specifikované funktorem (ADDR), předložkou (před), pádem (7) a informací o obligatornosti (?).

PDT-VALLEX začal vznikat téměř zároveň s PDT: jednak je valence součástí FGD a tudíž se v PDT měla objevit, jednak informace o obligatornosti členů valence pomáhají s doplňováním uzlů, které byly na povrchu vypuštěné, do t-roviny [5]. Na počátku se za základ PDT-VALLEXU vzalo několik set prvních sloves VALLEXU[4] a dále k němu přidával všechna lemmata, na která se v datech narazilo. Není tady kladen žádný důraz na úplnost slovníkového hesla, zanášely se pouze ty rámce, které byly potřeba. Do slovníku se také zaváděla substantiva (pouze deverbativní) a některá adjektiva. Celý slovník byl na závěr ručně zkontrolován. Slovník čítá na 10 200 sloves, substantiv a adjektiv. Poté se přistoupilo k definitivnímu označkování t-roviny PDT pomocí tohoto valenčního slovníku.

2.3 WordNet

Čtenář, který se s WordNetem již dříve seznámil, může bez obav následující kapitolu přeskočit. Přestože takových bude jistě většina, sluší se WordNet plnohodnotně uvést a čtenáře s ním seznámit, neboť se o něj opírá značná část naší práce.

WordNet (budeme zkracovat WN), ač je to také slovník, má úplně jiné uspořádání než slovníky, které bereme běžně do ruky v tištěné podobě. WordNet je určitý druh ontologie.

Ontologie[5] je filosofická disciplína zabývající se jsoucnem. V počítačové vědě se používá (poněkud nadneseně) pro dále vysvětlený datový model. Popisuje objekty (z nějaké oblasti) a zachycuje vztahy mezi nimi. Hlavními stavebními prvky ontologií jsou:

prvky	nejnižší, základní objekty ontologie;
třídy	skupiny objektů;
vlastnosti	znaky, rysy a atributy, které mohou mít objekty a které mohou sdílet;
vztahy	způsoby, jakými jsou jednotlivé objekty navzájem provázány.

Hlavní využití nacházejí ontologie v oblasti umělé inteligence, sémantických webů a při snaze počítačově representovat (části) světa.

[4] Nezaměňovat, jedná se o dva různé slovníky, které vznikaly paralelně. Nepatrně starší VALLEX vznikal s cílem úplného pokrytí vybraných frekventovaných sloves. (Poslední veřejná verse 1.0 čítá 1400 sloves, ale nadále se masivně se rozrůstá.) Tím ale VALLEX nebyl dostatečný pro anotaci PDT, protože v něm mnoho sloves chybělo.

[5] Zdroj: `http://en.wikipedia.org/wiki/Ontology_(computer_science)`

Jednotlivá hesla WordNetu nejsou řazena lineárně (a uspořádána typicky podle abecedy), ale jsou rozprostřena do sítě, na níž není definováno lineární uspořádání. Některé informace (*vlastnosti*) o daném slově jsou uvedeny přímo u něj (jak jsme na to zvyklí z tištěných slovníků všech druhů), ale jiné, mnohdy zajímavější (a v případě WN dokonce ty hlavní, kvůli nimž vznikl a na nichž je založen), vyčteme pouze z uspořádání celé sítě, z okolí slova, které nás zajímá (*vztahy*). Slovníkem se prochází na základě sémantiky namísto abecední podobnosti.

Když v roce 1978 začal profesor psychologie na universitě v Princetonu George A. Miller[6] budovat základy WN, byl to pokus převést do praxe představy psycholingvistů[7] o fungování asociací a spojů v lidském mozku, o struktuře, kterou si nějak uchováváme a která nám umožňuje rychle vyhledávat již poznané věci (resp. slova) a jim podobná, související, přesněji či obecněji určující atd.

WordNet je založen hlavně na vztahu synonymie, hyponymie a hyperonymie. Slova, nazývaná literály (v terminologii ontologie *prvky*), stejného významu tvoří synonymní[8] množiny, tzv. synsety (*třídy* v terminologii ontologie). Tyto synsety jsou uspořádány do stromové struktury pomocí hypo-hyperonymických vazeb (matka je hyperonymum, dcera je hyponymum).

Z pohledu matematického je WN orientovaný, graf jehož vrcholy tvoří synsety a hrany tvoří vazby mezi nimi (krom zmíněné hypo-hyperonymie také antonymie, mero_part, holo_member aj., souhrnně jsou označovány jako ILR, internal language relations). Na úplném vrcholu hierarchického uspořádání stojí top-level koncepty (jako například: žijící, artefakt, funkce, budova, přírodní, ...) pod nimiž visí jejich hyponyma a hyponyma hyponym atd. WN si tedy můžeme představit jako les[9], zaměříme-li se výhradně na hyponymické hrany. Je důležité ještě jednou zdůraznit, že vrchol representuje nějaký **význam**, který je zastoupen často mnoha **slovy**.

2.3.1 Princetonský WordNet

První WordNet byl sestaven na Princetonské universitě, New Jersey, kde byla také vymyšlena jeho koncepce [3]. Je pochopitelně v angličtině a jsou do něj zanesena substantiva, adjektiva, verba a adverbia. V poslední versi WordNet 2.1 je přes 100 000 synsetů.

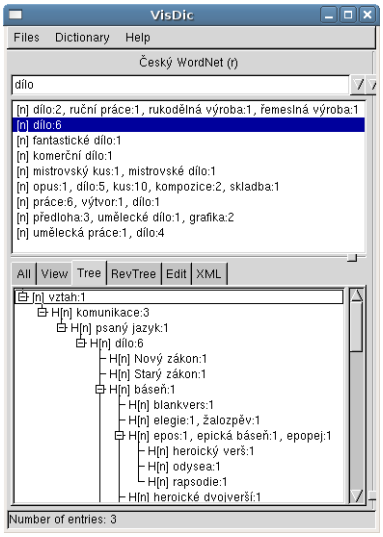
V pozdějších letech se vyvinulo mnoho odnoží, které se od Princetonského více či méně liší, ale které se k němu hlásí jako k hlavní inspiraci.

[6] Zdroj: `http://www.fathom.com/feature/1140/index.html`

[7] Zdroj: `http://wordnet.princeton.edu/`

[8] Není to absolutní synonymie, ale slova v jednom synsetu jsou vždy v nějakém kontextu vzájemně nahraditelná.

[9] Přesněji řečeno, není to už ani les, neboť stromová struktura sice byla záměrem, ale na několika místech je porušena vícenásobnou dědičností. Jsou to však spíše výjimky, takže na celkové představě o WN se tím prakticky nic nemění.



Obrázek 2.4: Slovo „dílo“ v Českém WordNetu (zobrazeno ve VisDicu [9])

2.3.2 Euro WordNet

O významu myšlenky WordNetu svědčí mimo jiné i to, že bylo založeno několik dalších projektů podobného charakteru.[10] Jejich cílem bylo přenést (neboť o překladu se dost dobře mluvit nedá) WN do jiných jazyků. Při té příležitosti byly do WN i do vznikajících jazykových mutací přidány odkazy na takzvaný Inter-Lingual-Index, všude zmiňovaný jen jako ILI.

Jedním z těchto projektů, Euro WordNet[11] (dále EWN), vytvářel a standardisoval WordNet v sedmi evropských jazycích. Kromě estonštiny, francouzštiny, holandštiny, italštiny, němčiny a španělštiny se zapojila také čeština. Když byl projekt EWN v roce 1999 uzavřen, byla tedy na světě první verse Českého WordNetu (ČWN). Ta se ještě dále vyvíjela a upravovala, například v projektu BalkaNet (vývoj WordNetů pro jazyky na Balkáně; zkušenosti s ČWN zde byly cenným přínosem (například v [14])). Český WordNet jsme vzali za zdroj distinkcí významu slov a stal se základem pro náš experiment.

2.4 btred, TrEd

Pro práci s PDT (kap. 2.1) bylo v Ústavu formální a aplikované lingvistiky na MFF UK vyvinuto více nástrojů. Pro naše účely jsme si zvolili btred (skriptovací nástroj postavený nad Perlem) a jeho grafickou nadstavbu TrEd [12].

[10] Bližší informace na stránkách `www.globalwordnet.org`.
[11] Homepage: `http://www.illc.uva.nl/EuroWordNet/`

TrEd slouží k prohlížení a editaci stromových struktur, v našem případě t-roviny a a-roviny PDT. (Byl také používán jako anotační nástroj pří vývoji PDT.) Zobrazuje jednotlivé věty z načteného souboru, a to ve formě čitelné věty a jako strom. Je možno si navolit, jaké atributy uzlu se budou do obrázku vpisovat. Ukázku (pouze výsledného stromu, nikoli celého nástroje) jsme viděli na obrázku 2.1.

TrEd jsme používali hlavně k ověření dílčích otázek a řešení problémů. Mnohem více jsme pracovali s nástroji na vyhledávání v treebanku. Na adrese `http://ufal.mff.cuni.cz/pdt2.0/doc/tools/tred/bn-tutorial.html` je přehledný tutoriál pro užívání btredu. Program btred je ideální k dávkovému procházení velkého množství stromů PDT a vyhledávání informací. Zadá se mu, co má v souborech procházet (jen stromy/jednotlivé uzly/...), a předává se mu makro v Perlu, kterým se zpracuje každý z nalezených stromů/uzlů/... My toho využívali pro průchod PDT a vyhledání vhodných rysů pro strojové učení. Z maker (soubory `PML_A.mak`) jsme používali funkce `GetEParents()`, `GetEChildren()`, `DiveAuxCP()` a `GetANodeByID()` pro „správný“ průchod stromem.

Oba dva programy jsou šiřitelné pod licencí GPL. Na CD nejsou, ale je možno je stáhnout na adrese `http://ufal.mff.cuni.cz/~pajas/tred/tred-dep-unix.tar.gz`.

2.5 C4.5

C4.5 je nástroj zajišťující komfortně strojové učení (o něm více až v kapitole 4.3.1). Cílem práce nebylo implementovat metodu strojového učení, proto jsme použili otestovaný nástroj. Program C4.5 je předchůdcem (a z jiného pohledu také volně šiřitelnou versí) používaného program C5.0, nebo také see5 (`http://www.rulequest.com/see5-info.html`, kde je také srozumitelný tutoriál). Protože hlavní funkčnost C5.0 implementuje již C4.5, stačila nám tato již nepodporovaná verse. Její autor, profesor Ross Quinlan, neposkytuje práva k šíření (proto není k disposici na CD), ale umožňuje stažení z jeho osobních stránek `http://www.rulequest.com/Personal/`. Manuálová stránka pro C4.5 je zde `http://cerium.raunvis.hi.is/~tpr/courseware/ml/source/c4_5/c4_5.html`.

Přestože strojové učení představíme až později, alespoň dvěma větami nastíníme, jak program funguje a co dokáže. Nástroj C4.5 dokáže z dostatečně velkého množství případů (charakterisovaných nějakými rysy a cílovou hodnotou) postavit rozhodovací strom, který cílovou hodnotu přiřazuje sám. Tuto schopnost případně otestuje a změří na zvláštní části dat.

2.6 Perl, Bash

Tyto dva notoricky známé nástroje asi netřeba zvláště představovat. Více o Perlu lze zjistit na `http://www.perl.org`, o Bashi na `http://www.gnu.org/software/bash/`.

Perl (verse 5.8.7) jsme zvolili pro snadné skriptování a mocné řetězcové operace díky regulárním výrazům. Chtěli bychom upozornit na velice dobrý tutoriál na webu, kterému vděčíme za zvládnutí základů jazyka: `http://docs.rinet.ru/P7/`.

Bash (verse 3.00.16) jsme si vybrali proto, jak je přirozeně a pohotově začleněn do systému Linux a jednoduše manipuluje s filesystémem. Spolu s ním jsme používali také některé standardní utility například z balíku textutils, coreutils (`wc`, `cat`, `tail`, `head`; `cp`, `ln`, `ls`, `basename`), gzip (`gzip`, `gunzip`, `zgrep`, `zcat`), diffutils (`diff`), `tar`, `grep` a mnoho dalších (viz jejich manuálové stránky). To byl druhý důvod výběru Bashe.

2.7 VisDic

S programem VisDic[12] jsme se již mimoděk seznámili na obrázku 2.4, kde zobrazoval výsek ČWN. To je také jeho hlavní využití. VisDic (sloužící k visualisaci slovníku) dokáže načíst více slovníků (nejen typu WordNetu, [9]) a zobrazovat je paralelně, každý v jiném režimu (pokud si uživatel přeje). Budeme-li mluvit o WordNetu, tak užitečné režimy jsou:

- seznam všech synsetů;
- zobrazení jednoho synsetu přehledně (tj. barevně odlišeno ID synsetu, synonymní literály, hyperonyma, hyponyma, glosa, příklad apod.);
- zobrazení jednoho synsetu v syrových datech, konkrétně v XML;
- zobrazení vybraného synsetu v hierarchické struktuře: jsou vidět všichni předchůdci až ke kořeni; co se týče hyponym, je možné jejich seznamy rozbalovat a opět sbalovat;
- zobrazení vybraného synsetu v obrácené hierarchii: prakticky totéž, ale otočeno; nad synsetem jsou hyponyma a pod ním lze zobrazit hyperonyma;
- poslední režim je editační: editovat lze všechny položky a navíc přidávat a mazat celé synsety.

VisDic jsme používali spíše okrajově, k vyhledání konkrétních jevů a ověření určitých hypothes.

[12] Homepage: `http://nlp.fi.muni.cz/projekty/visdic/`, kde je také zdarma ke stažení.

Kapitola 3

Příprava dat pro přiřazování synsetů

Jak již bylo řečeno, tato část naší práce měla za úkol automaticky přiřazovat k textu v PDT synsety Euro WordNetu jakožto nositele významu slov.

3.1 Popis data

K disposici jsme měli výstupní data z několikaletého (a nutno zdůraznit: stále ještě probíhajícího) projektu, jehož cílem je anotovat Pražský závislostní korpus pomocí synsetů z EWN. Je na místě hned na počátku zmínit dvě velké nesnáze. Jedna je daná realisací českého EWN, druhá souvisí s formátem dat.

3.1.1 Problém 1: Český Euro WordNet

Český Euro WordNet (a WordNet vůbec, viz kapitola 2.3) je zajímavý a velice podnětný projekt, avšak jeho realisace naráží na řadu problémů daných nesnadným uchopením pojmu význam – často se ani rodilý mluvčí sám není schopen uspokojivě rozhodnout, kolik významů vlastně dané slovo má, jak je přesně vymezit a kam do hierarchie jej v důsledku zavěsit.

Tyto všeobecné problémy myšlenky WordNetu jsou nepříjemné už samy o sobě. K tomu je potřeba přičíst fakt, že Český WordNet nebyl na jaře 2002, ze kdy pocházejí naše data[1], ještě zcela dokončen.

Verze WordNetu, která byla použita k anotaci, byl český klon, který vzniká v rámci projektu Euro WordNet v Brně a vychází z překladu anglického WordNetu. Protože tento projekt stále probíhá a WordNet se mění, byla zvolena aktuální verze k datu začátku anotace (ono jaro 2002) a pozdější změny již nebyly nijak reflektovány (z důvodu konsistence). Použitý

[1] Budeme-li tedy o slovníku mluvit v přítomném čase, nemyslíme tím jeho současnou versi, nýbrž tuto, která sloužila k anotaci dat PDT.

slovník není tudíž vždy vyhovující, pravděpodobně v důsledku rozpracovanosti. Navíc jsou některé synsety poplatné překladu z anglického originálu.[2] Dále WordNet, který byl použit, ještě zpravidla u synsetu neobsahoval glosy či příklady, takže význam bylo lze uchopit pouze pomocí hyponym a hyperonym daného synsetu. To je pochopitelně velká překážka pro anotátory.

Jako příklady „zvláštností“ uveďme, že

- WordNet obsahuje jedinou interjekci (alespoň v ČWN je to tak označeno): „šššˇ; nevidíme důvod, proč do WordNetu zavádět citoslovce, ale navíc je zde šššˇ ve významu „dětské slovo pro lokomotivu“, čili zřetelně coby substantivum;
- synsety „valný:1“ a „valný:3“ se neliší ničím jiným než identifikačním číslem synsetu, neboť jsou v synsetu samy a nemají žádné vazby na ostatní synsety;
- spojení „životní prostor“ se vyskytuje dvakrát, vždy samo v synsetu: jako „životní prostor:1“ a (opět) „životní prostor:1“ (skutečně se stejným číslem, liší se jen ID synsetu); první má pod sebou (tedy jako svá hyponyma) „přízemí:2“, „stupňovité sezení:1“ a „tribuna:3“; druhý výskyt má pod sebou „oddychový čas:1“ a sám sebe (!);
- významy mnoha slov se nedají přesně vyčíst, neboť není k disposici dostatek informací (jak je ostatně patrné i z předchozího příkladu).

Celkově se zdá, že autoři na jednu stranu rozštěpili významy na drobnější dílky, než by bylo vhodné[3], na druhou stranu pak ale pro plné využití základního stavebního prvku, totiž synonymie vytvářející synsety, slučovali a termínem synonyma tak označovali věci poměrně různorodé (např. reportáž $\approx$[4] studie, časem $\approx$ občas, ozubené kolo $\approx$ přehazovačka apod.).

Tolik k problémům WordNetu, které ovlivnily spíše práci anotátorů, jejich shodu a kvalitu získaných dat. Druhá potíž se týká formátu dat, která jsme získali, a ovlivnila proto naši práci při automatickém přiřazování synsetů.

3.1.2 Problém 2: Data z PDT 1.0

Data z PDT 1.0, na kterých se započala anotace, byla ve formátu CSTS, který se dnes již nepoužívá. Při anotování nebyl důvod zaměřit se na data z nějaké roviny, takže pro část

[2] O tom a jiných problémech při tvorbě ČWN pojednává [14].

[3] Soudíme tak podle toho, že když se nyní opravují a kontrolují anotace, dochází i k úpravám slovníku: a často jsou slučovány mnohé synsety (ve WordNetu se např. rozhodli slovo „dobrý“ rozštěpit do osmi (!) synsetů; ty byly později při úpravách slovníku sloučeny v synset jediný), neboť není patrná jednoznačná distinkce mezi jednotlivými významy – viz absence glos a ostatních informací.

[4] $\approx$ ve smyslu synonymie

dnes anotovaných dat existuje t-rovina, pro většinu je však k disposici jen a-rovina. Všechny informace z ČWN potřebné pro anotaci byly vloženy přímo do anotovaných dat – a stejně tak vypadal i výstup anotace.

Proč tyto tři skutečnosti vadí?

1) Formát PDT 1.0 vs. PDT 2.0:

Během práce na anotaci se zcela zásadně změnila specifikace formátu, ve kterém se PDT uchovává. Anotace významu probíhala na datech PDT 1.0, který byl uložen ve formátu CSTS, kdežto všechny ostatní pro naši práci potřebné informace jsou obsaženy v datech PDT 2.0 Beta, uloženém ve formátu PML ([13]). Pro jednoduchost řekněme, že rozdíly jsou hlavně ve čtyřech věcech:

1. Zatímco PDT 1.0 bylo ještě uloženo ve formátu z rodiny SGML, současné PDT 2.0 Beta už je v XML. Tato změna jazyka byla využita k rozsáhlejším úpravám struktury formátu.

2. Zcela novým způsobem se segmentovaly věty do souborů, změnily se tudíž i jejich názvy.

3. Změnilo se kódování češtiny z ISO-8859-2 v PDT 1.0 na UTF-8 v nové versi.

4. V PDT 2.0 od sebe byly odděleny jednotlivé roviny representace vět, tj. jedna věta může být obsažena až ve čtyřech vzájemně provázaných souborech, pro slovní, morfologickou, analytickou a tektogramatickou rovinu (resp. v přesnější terminologii PDT pro w-, m-, a- a t-rovinu).

Poslední rozdíl nám naštěstí ve fázi zpracovávání a převodu dat nezpůsoboval potíže. Největší problémy byly s jiným dělením do souborů.

Dá se v podstatě říct, že CSTS a PML jsou dva zcela odlišné formáty. Společné mají pouze to, že representují tatáž data.

Příklad :

Prohlédněme si postupně v obou formátech kratičký úryvek textu:

Pracovníci Národní galerie nemohou vystupovat jako soudní znalci.

* Neexistuje tedy (. . .).

Uvidíme na něm, jak se vypadají jednotlivá slova, ale také jak se uvozuje celý dokument, jak se značí hranice vět a odstavců a kterak se značkují speciální mimotextové symboly jako třeba hvězdička. Nejprve tedy ve starším CSTS (z rodiny SGML).

soubor `ca10.am.gz`

```
<csts lang=cs>
      <h>
            <source>Českomoravský profit</source>
            ...
      </h>
      <doc file="s/inf/j/1994/cmpr9410" id="016">
            <a>
                 ...
            </a>
            <c>

              <p n=10>
                  <s id="cmpr9410:016-p10s3">
                      <f cap>Pracovníci
                        <l>pracovník<t>NNMP1-----A----
                        <MDl src="a">pracovník
                          <MDt src="a">NNMP1-----A----
                        <MDl src="b">pracovník
                          <MDt src="b">NNMP1-----A----
                        <A>Sb<r>1<g>4
                      <f cap>Národní
                        <l>národní<t>AAFS2----1A----
                        <MDl src="a">národní
                          <MDt src="a">AAFS2----1A----
                        <MDl src="b">národní
                          <MDt src="b">AAFS2----1A----
                        <A>Atr<r>2<g>3
                      <f>galerie
                        <l>galerie<t>NNFS2-----A----
                        <MDl src="a">galerie
                          <MDt src="a">NNFS2-----A----
                        <MDl src="b">galerie
                          <MDt src="b">NNFS2-----A----
                        <A>Atr<r>3<g>1
                      <f>nemohou
                        <l>moci_^(mít_možnost_[něco_dělat])
                          <t>VB-P---3P-NA--1
                        <MDl src="a">moci_^(mít_možnost_[něco_dělat])
```

```
    <MDt src="a">VB-P---3P-NA--1
  <MDl src="b">moci_^(mít_možnost_[něco_dělat])
    <MDt src="b">VB-P---3P-NA--1
  <A>Pred<r>4<g>0
<f>vystupovat
  <l>vystupovat_:T<t>Vf--------A----
  <MDl src="a">vystupovat_:T
    <MDt src="a">Vf--------A----
  <MDl src="b">vystupovat_:T
    <MDt src="b">Vf--------A----
  <A>Obj<r>5<g>4
<f>jako
  <l>jako<t>J,-------------
  <MDl src="a">jako
    <MDt src="a">J,-------------
  <MDl src="b">jako
    <MDt src="b">J,-------------
  <A>AuxY<r>6<g>8
<f>soudní
  <l>soudní<t>AAMP1----1A----
  <MDl src="a">soudní
    <MDt src="a">AAMP1----1A----
  <MDl src="b">soudní
    <MDt src="b">AAMP1----1A----
  <A>Atr<r>7<g>8
<f>znalci
  <l>znalec<t>NNMP1-----A----
  <MDl src="a">znalec
    <MDt src="a">NNMP1-----A----
  <MDl src="b">znalec
    <MDt src="b">NNMP1-----A----
  <A>Atv<r>8<g>1
  <D>
<d>.
  <l>.<t>Z:-------------
  <MDl src="a">.
    <MDt src="a">Z:-------------
  <MDl src="b">.
```

```
                        <MDt src="b">Z:-------------
                      <A>AuxK<r>9<g>0
              <p n=11>
                  <s id="cmpr9410:016-p11s1">
                      <i>b
                      <d>&ast;
                        <l>&ast;<t>Z:-------------
                        <MDl src="a">&ast;
                          <MDt src="a">Z:-------------
                        <MDl src="b">&ast;
                          <MDt src="b">Z:-------------
                        <A>AuxG<r>1<g>2
                      <f cap>Neexistuje
                        <l>existovat_:T<t>VB-S---3P-NA---
                        <MDl src="a">existovat_:T
                          <MDt src="a">VB-S---3P-NA---
                        <MDl src="b">existovat_:T
                          <MDt src="b">VB-S---3P-NA---
                        <A>Pred<r>2<g>0
                      <f>tedy
                        <l>tedy<t>Db-------------
                        <MDl src="a">tedy
                          <MDt src="a">Db-------------
                        <MDl src="b">tedy
                          <MDt src="b">Db-------------
                        <A>AuxY<r>3<g>2
                      ...
              <p n=12>
                  <s id="cmpr9410:016-p12s1">
                      <i>/b
                      ...

            </c>
      </doc>
      <doc file="s/inf/j/1994/cmpr9410" id="017">

      </doc>
</csts>
```

Stručně popíšeme některé základní elementy formátu CSTS:

`<csts>` je kořenový element formátu CSTS, pod ním visí vše ostatní;

`<h>` je hlavička souboru a obsahuje například jméno periodika, ze kterého text pochází, rok vydání a informace o anotaci;

`<doc>` uvozuje nový dokument. V jednom souboru, jak je patrné z příkladu, může být i více dokumentů (což je v PDT 2.0 změněno). Atribut `file` spolu s atributem `id` se později staly vodítky při mapování do nového uspořádání v PDT 2.0;

`<p>` obsahuje každý nový odstavec, paragraf;

`<s>` obsahuje jednotlivou větu;

`<f>` je element pro slovní formu (původní, nikoli lemmatisovanou);

`<l>` nese informaci o lemmatu nadřazeného elementu `<f>`;

`<t>` obsahuje všechny morfologické informace ve formě tzv. tagu, viz [6, 8];

`<A>` je elementem obsahujícím analytickou funkci slova ve větě;

`<r>` informuje o pořadí slova ve větě;

`<g>` odkazuje ke slovu nadřazenému (governing node), identifikovaném pomocí jeho pořadí, `<r>`;

`<d>` je vlastně variantou `<f>` pro veškeré tokeny, které nejsou slovy (tedy pro interpunkci).

Nyní se podíváme na tutéž část textu v novějším formátu PML (z rodiny XML), použitém v PDT 2.0. Tam jsou morfologická informace a analytická informace uloženy ve dvou souborech.

soubor `cmpr9410_016.m.gz`

```
<?xml version="1.0" encoding="utf-8"?>
<mdata xmlns="http://ufal.mff.cuni.cz/pdt/pml/">

    <head>
        <schema href="mdata_schema.xml" />
        <references>
            <reffile id="w" name="wdata" href="cmpr9410_016.w.gz" />
        </references>
```

```
</head>
<meta>
    <lang>cs</lang>
</meta>
<s id="m-cmpr9410-016-p2s1">
    ...
</s>
...
<s id="m-cmpr9410-016-p10s3">
    <m id="m-cmpr9410-016-p10s3w1">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w1</w.rf>
        <form>Pracovníci</form>
        <lemma>pracovník</lemma>
        <tag>NNMP1-----A----</tag>
    </m>
    <m id="m-cmpr9410-016-p10s3w2">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w2</w.rf>
        <form>Národní</form>
        <lemma>národní</lemma>
        <tag>AAFS2----1A----</tag>
    </m>
    <m id="m-cmpr9410-016-p10s3w3">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w3</w.rf>
        <form>galerie</form>
        <lemma>galerie</lemma>
        <tag>NNFS2-----A----</tag>
    </m>
    <m id="m-cmpr9410-016-p10s3w4">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w4</w.rf>
        <form>nemohou</form>
        <lemma>moci_^(mít_možnost_[něco_dělat])</lemma>
        <tag>VB-P---3P-NA--1</tag>
    </m>
    <m id="m-cmpr9410-016-p10s3w5">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w5</w.rf>
        <form>vystupovat</form>
        <lemma>vystupovat_:T</lemma>
        <tag>Vf--------A----</tag>
```

```
    </m>
    <m id="m-cmpr9410-016-p10s3w6">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w6</w.rf>
        <form>jako</form>
        <lemma>jako</lemma>
        <tag>J,-------------</tag>
    </m>
    <m id="m-cmpr9410-016-p10s3w7">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w7</w.rf>
        <form>soudní</form>
        <lemma>soudní</lemma>
        <tag>AAMP1----1A----</tag>
    </m>
    <m id="m-cmpr9410-016-p10s3w8">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w8</w.rf>
        <form>znalci</form>
        <lemma>znalec</lemma>
        <tag>NNMP1-----A----</tag>
    </m>
    <m id="m-cmpr9410-016-p10s3w9">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p10s3w9</w.rf>
        <form>.</form>
        <lemma>.</lemma>
        <tag>Z:-------------</tag>
    </m>
</s>
<s id="m-cmpr9410-016-p11s1">
    <m id="m-cmpr9410-016-p11s1w1">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p11s1w1</w.rf>
        <form>*</form>
        <lemma>*</lemma>
        <tag>Z:-------------</tag>
    </m>
    <m id="m-cmpr9410-016-p11s1w2">
        <src.rf>manual</src.rf>
        <w.rf>w#w-cmpr9410-016-p11s1w2</w.rf>
        <form>Neexistuje</form>
        <lemma>existovat_:T</lemma>
        <tag>VB-S---3P-NA---</tag>
```

```
            </m>
            <m id="m-cmpr9410-016-p11s1w3">
                <src.rf>manual</src.rf>
                <w.rf>w#w-cmpr9410-016-p11s1w3</w.rf>
                <form>tedy</form>
                <lemma>tedy</lemma>
                <tag>Db-------------</tag>
            </m>
        </s>
    </mdata>
```

Opět vysvětlíme důležitější elementy formátu PML, m-roviny:

`<mdata>` je kořenový element m-roviny formátu PML. Atribut se odkazuje na namespace PML;

`<head>` obsahuje odkazy na schema PML a na w-rovinu;

`<s>` vyjadřuje celou novou větu (téměř zachován z CSTS);

`<m>` obsahuje informace o jednom slově;

`<form>` je správný tvar slova ve větě (případně opravený);

`<lemma>` je jeho lemmatisovaný tvar;

`<tag>` obsahuje morfologický tag.

Zdá se, že nový zápis je přibližně stejně dlouhý. To ovšem byla dosud pouze m-rovina. Nebudeme čtenáře nudit podobně vypadajícím a podobně dlouhým výpisem a-roviny. Uvedeme pouze, že (kromě téměř shodně vypadající hlavičky) má odlišnou strukturu, neboť jednotlivé elementy jsou do sebe zanořovány způsobem odpovídajícím podřízenosti v analytickém stromě.[5] Jako příklad uvedeme pouze krátkou ukázku tří slov „Pracovníci$_{\mathsf{Sb}}$ ... galerie$_{\mathsf{Atr}}$ nemohou$_{\mathsf{Pred}}$...“. Nelze si nepovšimnout, že v souboru a-roviny se již nevyskytují lemmata, ale pouze odkazy k uzlům na nižší rovině, které lemmata obsahují (pro lepší orientaci v příkladu jsme proto připojili do ukázky ke slovům i jejich analytickou funkci). První uzel v ukázce je kořen stromu; na něm teprve visí predikát.

[5] Zatímco v CSTS byly syntaktické závislosti zachyceny odkazem na nadřízené slovo, zde je stromová struktura věty representována stromovou strukturou XML.

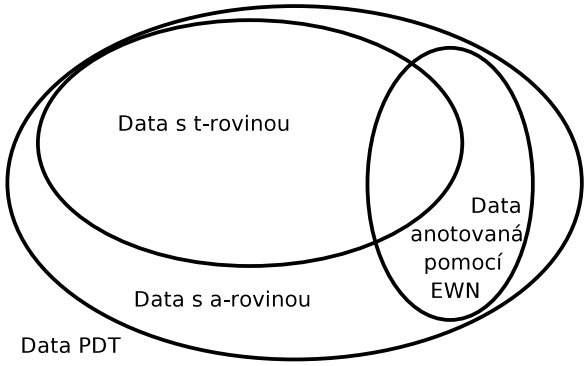
soubor `cmpr9410_016.a.gz`

```
<LM id="a-cmpr9410-016-p10s3">
  <s.rf>m#m-cmpr9410-016-p10s3</s.rf>
  <ord>0</ord>
  <children>

    <LM id="a-cmpr9410-016-p10s3w4">
      <m.rf>m#m-cmpr9410-016-p10s3w4</m.rf>
      <afun>Pred</afun>
      <ord>4</ord>
      <children>

        <LM id="a-cmpr9410-016-p10s3w1">
          <m.rf>m#m-cmpr9410-016-p10s3w1</m.rf>
          <afun>Sb</afun>
          <ord>1</ord>
          <children>

            <LM id="a-cmpr9410-016-p10s3w3">
             <m.rf>m#m-cmpr9410-016-p10s3w3</m.rf>
             <afun>Atr</afun>
             <ord>3</ord>
             <children>

               ...

             </children>
            </LM>

          </children>
        </LM>

      </children>
    </LM>

  </children>
</LM>
```

2) t-rovina vs. a-rovina:

Jak jsme uvedli v kapitole 2.1.1, nejsou všechny věty obsažené v PDT anotované na všech rovinách. Navíc nebyla (a ani nemohla v počátcích projektu být) při výběru dat k anotaci významu zohledněna hloubka, do které budou data anotována v PDT. Tak se stalo, že část získaných anotovaných dat má v PDT 2.0 svou representaci na m-rovině, a-rovině i t-rovině, pro větší část však dnes t-rovina chybí. Tento fakt jednak ovlivnil rozhodnutí o umístění informace o významu do dat (viz níže, začátek kapitoly 3.2.2), jednak v podstatě brání pokusům o automatické přiřazování synsetů s využitím atributů t-roviny, protože není k disposici dostatek dat (více viz níže, kapitola 4.5).

Podrobnosti přináší obrázek 3.1 a tabulka 3.1. Obrázek usnadňuje představu o tom, jak se k sobě mají data s anotací na a-rovině a na t-rovině a v jakém jsou vztahu k proběhlé anotaci významu pomocí ČWN. V tabulce jsou k nahlédnutí přesné počty tokenů (slov a interpunkce) na obou rovinách a podíl anotovaných tokenů v nich.



Obrázek 3.1: Prolínání rovin PDT a anotace významu

Rovina	Počet tokenů v PDT	Počet tok. v anot.[6]	Již anotováno
má a-rovinu, nemá t-r.	671 490[7]	374 512	56 %
má a-rovinu, má t-r.	833 357[8]	132 958	16 %
má a-r. (oba dva příp.)	1 504 847[9]	507 470	34 %

Tabulka 3.1: Rozložení anotovaných dat v a-rovině a t-rovině, procento anotace

To znamená, že je dokončeno téměř třikrát více dat, u kterých je k disposici pouze a-rovina, než těch, které mají i tektogramatickou anotaci. Kdyby bylo všech 507 470 tokenů vzato z dat majících t-rovinu, bylo by již nyní dokončeno 61 % dat s t-rovinou.

[6] Jde o počet všech tokenů v části, která byla anotována podle ČWN. Zdaleka ne všem těmto tokenům se dostalo přiřazeného synsetu (například proto, že ne všechna lemmata jsou autosémantická).

[7] Viz též `http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch03.html#a-data-full` – tabulka 3.2

[8] Viz též `http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch03.html#a-data-full` – tabulka 3.1

[9] Viz též `http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/en/html/ch03.html#a-data-full` – tabulka 3.5

3) Smísení dat PDT a ČWN

Z čistě pragmatických důvodů (totiž aby bylo možno použít již existující program pro anotaci – DA, autor J. Hana) byl již beztak nepřehledný formát dat (CSTS) ještě více znečistěn. Přímo do dat, obsahujících postupně všechna slova textu (každé na jednom řádku a obohacené o všechny morfologické informace), byla zavedena kompletní informace z ČWN, která byla užitečná pro rozhodování anotátorů. Na každém řádku tedy byla morfologická informace nahrazena všemi potenciálně možnými synsety, včetně těch s víceslovnými literály, plus jedenácti chybovými kategoriemi (viz níže). (V rekordním případě to znamená 96 synsetů i s popisem a 28 487 znaků na jednom řádku.) Nebylo tedy potřeba vyvíjet nový anotační nástroj a anotace byla pohodlná, cenou však byl obrovský nárůst dat s krajně nepřehledným formátem. Pojednává o tom [15].

Příklad:

```
<f>galerie
    <l->galerie--wsd-n-1-6120-16010-2545656/n/o
        <t->l: galerie-n,ochoz-n</t>
        <t->d: narrow recessed</t>
        <t-> balconylike area along an</t>
        <t-> upper floor on the</t>
        <t-> interior of a building</t>
        <t-> usu marked by a colonnade</t>
        <t->hl: balkon-n,lodžie-n</t>
        <t->hd: a platform projecting</t>
        <t-> from the wall of a</t>
        <t-> building and surrounded</t>
        <t-> by a balustrade or</t>
        <t-> railing or parapet</t>
    </l>
    <l->galerie--wsd-n-3-6121-16013-2546158/n/o
        <t->l: galerie-n,umělecká</t>
        <t-> galerie-n</t>
        <t->d: a room or series of rooms</t>
        <t-> where works of art are</t>
        <t-> exhibited</t>
        <t->hl: místnost-n,pokoj-n,světni</t>
        <t-> ce-n</t>
        <t->hd: an area within a building</t>
        <t-> enclosed by walls and</t>
        <t-> floor and ceiling; "the</t>
        <t-> rooms were very small but</t>
        <t-> they had a nice view"</t>
```

```
</l>
<l->*******galerie*******--wsd-n-1-1-1-1/n/o
    <t->* chybejici vyznam *</t>
</l>
<l->*******galerie*******--wsd-n-2-2-2-2/n/o
    <t->*chybna reflexivita*</t>
</l>
<l->*******galerie*******--wsd-n-3-3-3-3/n/o
    <t->*chybi pozitiv.vyz.*</t>
</l>
<l->*******galerie*******--wsd-n-4-4-4-4/n/o
    <t->*chybi negativ.vyz.*</t>
</l>
<l->*******galerie*******--wsd-n-5-5-5-5/n/o
    <t->*chybna lemmatizace*</t>
</l>
<l->*******galerie*******--wsd-n-6-6-6-6/n/o
    <t->*metaforick.pouziti*</t>
</l>
<l>*******galerie*******--wsd-n-7-7-7-7/n/o                    ⇐
    <t>* vlastni jmeno *</t>
</l>
<l->*******galerie*******--wsd-n-8-8-8-8/n/o
    <t->*nejasnyVyznamSlova*</t>
</l>
<l->*******galerie*******--wsd-n-9-9-9-9/n/o
    <t->*nejasneVyznamyLit.*</t>
</l>
<l->*******galerie*******--wsd-n-10-10-10-10/n/o
    <t->*chybiObecnyVyznam *</t>
</l>
<l->*******galerie*******--wsd-n-0-0-0-0/n/o
    <t->*******jiny*********</t>
</l>
<l->umělecká galerie--wsd-n-1-6121-16013-2546158/n/o
    <t->l: galerie-n,umělecká</t>
    <t-> galerie-n</t>
    <t->d: a room or series of rooms</t>
    <t-> where works of art are</t>
    <t-> exhibited</t>
```

```
            <t->hl: místnost-n,pokoj-n,světni</t>
            <t-> ce-n</t>
            <t->hd: an area within a building</t>
            <t-> enclosed by walls and</t>
            <t-> floor and ceiling; "the</t>
            <t-> rooms were very small but</t>
            <t-> they had a nice view"</t>
        </l>
    </f>
```

Seznámíme se s použitými elementy:

`<f>` je slovní forma, kterou je lemma vyjádřeno ve větě;

`<l>` jsou jednotlivé nabízené (multi)literály; obsahují vždy jak lemma, tak odkazy do ČWN;

`<t>` v elementech `<t>` jsou obsaženy potřebné informace z českého i anglického WN nutné pro anotátory, aby mohli kvalifikovaně rozhodnout:

- `l:` je výčet literálů v synsetu,
- `d:` je popis, definice synsetu (resp. odpovídajícího synsetu v anglickém WN),
- `hl:` je výčet literálů v hyperonymu a
- `hd:` obsahuje definici hyperonymního synsetu.

Můžeme si povšimnout, že v uvedeném příkladu (přibližně v polovině výpisu, označeno ⇐) chybí pomlčka za `l`, místo „`<l->*******galerie*******--wsd-n-7-7-7-7/n/o:`“ je tam pouze „`<l>*******galerie*******--wsd-n-7-7-7-7/n/o`“ – takto program DA označí ten synset, který anotátor vybral. Ostatní ponechává pro případné pozdější kontroly beze změny. Bohužel, z neznámého důvodu se ve formátu občas objevují chyby a nesrovnalosti, se kterými je vždy potřeba se vypořádat jednotlivě.

Už jsme se zmínili a z příkladu je také patrné, že anotátoři měli k disposici tzv. chybové kategorie. Ty sloužily k anotování nějakým způsobem výjimečných jevů. Podrobně je možné si je prohlédnout v předchozím příkladu. Jednalo se hlavně o označení slov, která chyběla v ČWN (č. 1, 2, 3, 4, 9, 10), výskytů, které byly v PDT špatně lemmatisované (č. 5), a výskyty, jejichž význam člověk není schopen určit (není z kontextu jasné, č. 8), nebo nemá smysl určovat (vlastní jméno, č. 7, a metaforické užití[10], č. 6).

Protože anotátoři byli dva, všechny soubory jsme obdrželi v dvojím provedení: jednou od anotátora A, podruhé od anotátora B. Nazývejme je `soubor_A` a `soubor_B`.

[10] Význam metafor pochopitelně v obecném smyslu je potřeba určovat a také to jde. Nikoli však prostředky, které byly zvoleny pro tuto anotaci.

3.2 Zpracování a namapování dat

V předchozí kapitole jsme se zabývali tím, jaká data a v jakém formátu jsme měli na začátku k disposici. Také jsme nastínili, jaké problémy data obsahovala. Abychom mohli s daty dále pracovat, tedy pokusit se naučit strojově přiřazovat správné významy, tedy správné synsety, chtěli jsme uspořádat data do přehledné formy. Jako přirozené se jevilo do stávajícího formátu PDT 2.0 přidat k mnoha v současnosti již obsaženým anotovaným jevům také nově získanou informaci o významu slov. Pochopitelně jsme za relevantní informaci o významu považovali pouze tu, kde se oba anotátoři shodli na synsetu. V opačném případě, tedy když každý z anotátorů volil jinak, bylo slovo vynecháno stejně tak, jako by vůbec nebylo anotováno. Až bude souhlasná informace o významu od obou anotátorů vložena do formátu PDT, bude již možné použít existující nástroje na práci s PDT (btred), vyhledat k autosémantickým slovům zvolené rysy a předložit je programu C4.5 k trénování. Po prostudování dat před námi tedy ležely tři hlavní úkoly:

- a. zpracovat anotovaná data – sloučit soubory obou anotátorů, odhalit shodu
- b. namapovat soubory PDT 1.0 na PDT 2.0 (zmiňované nové rozdělení do souborů)
- c. vložit informaci o významu do PDT 2.0

Úkol (a) a (b) jsme provedli prvním skriptem v Perlu (`get_senses_from_ewn-pdt.pl`, kapitola 3.2.1), úkol (c) druhým skriptem (`map_senses_to_pdt.pl`, kapitola 3.2.2). Jejich činnost si nyní ukážeme.

3.2.1 Získání vybraných synsetů (`get_senses_from_ewn-pdt.pl`)

Jeho činnost zmíníme jen stručně, plný popis spadá do programátorské dokumentace. Tento skript se stará o následující úlohy:

- vezme `soubor_A` a `soubor_B`, porovná je
- rozlišuje tři varianty: liší se, shodují se na synsetu, shodují se na chybě
- všechny chyby zahazuje (včetně metafor a vlastních jmen[11])

[11] Původně jsme uvažovali, zda tyto jediné dvě chybové kategorie nemá smysl zachovat (opět, jen tehdy, shodnou-li se na ní oba anotátoři) a umožnit pak rozpoznat vlastní jméno či metaforu a označit ji. Převládly však nakonec předpokládané nevýhody: vlastních jmen tvořených od jednoho slova může být značné množství a spolu nemusí mít nic společného (pan Nový, Nová Guinea a Nová obchodní banka), což by našemu systému spíše práci ztěžovalo a znemožnilo mu odhalit souvislosti, pokud by v této kategorii bylo mnoho nesourodých prvků. O metaforách od jednoho slova toto platí dvojnásob.

- zahazuje tedy
  - a) nedesambiguované,
  - b) neshodně desambiguované,
  - c) chybové kategorie
- výstup rovnou třídí do souborů s novým názvem (tak, jak jsou dokumenty rozděleny a pojmenovány v PDT 2.0)

Podrobnější informace jsou uvedeny v dokumentaci, v příloze B.1.

3.2.2 Vložení zvolených synsetů do PDT 2.0 (`map_senses_to_pdt.pl`)

Než přistoupíme k samotnému skriptu, zmiňme ještě jedno rozhodnutí, které jsme museli učinit. Bylo zapotřebí rozhodnout, kam do poměrně složité struktury PDT informaci o významu vložit. Připomeňme, že každá věta se může vyskytovat ve dvou až čtyřech souborech, podle toho, do jaké hloubky anotace došla: Každá věta má přirozeně svou rovinu slovní, všechny jsou dále anotované alespoň na m-rovině. Přibližně $^3/_4$ vět mají také svou a-rovinu, a polovina z nich též t-rovinu. Informace o významu slova se sice hodí nejlépe na t-rovinu, neboť ta má nejblíže k representaci významu. Bohužel, jak už jsme zmínili v kapitole 3.1.2 (v její druhé části), zdaleka ne ke všem datům, která máme anotována pomocí ČWN, existuje t-rovina. Na stejném místě jsme uvedli, že omezit se na průnik dat s t-rovinou a dat s anotací významu by nepoužitelně zúžilo rozsah dat. Navíc je tu ještě jeden aspekt, který by bylo potřeba uspokojivě vyřešit, než by se začaly významy vkládat na t-rovinu: ne každé slovo na povrchu má svůj vlastní uzel na t-rovině. Například spojení „koncem roku“ se representuje jediným uzlem, jeho t_lemma je „rok“, functor je „TWHEN“ a subfunctor je „end“[12]. A přitom jak lemma „konec“, tak „rok“ jsou v Českém WordNetu obsaženy ve více synsetech a u obou se tedy očekává desambiguační informace. Takže do t-roviny informace o významu slova jednoduše vložit nebylo lze. Ze zbývající a-roviny a m-roviny se nám jako vhodnější jevila morfologická, protože se význam slova (navíc určovaný pomocí slovníku, jímž ČWN rozhodně je) dá považovat za další z informací, které souvisí přímo se slovem jako takovým – a nikoli se strukturou věty kolem něho budovanou na a-rovině. V neposlední řadě se sluší podotknout, že umístění na m-rovinu bylo technologicky snazší, neboť tam jsou slova ještě řazena stejně jako ve větě (tedy ne podle struktury ve větné skladbě) a uzly obsahují lemma. Usnadnilo to tedy vyhledání správné posice, kam atribut významu do souboru vložit.

Struktura souboru m-roviny je poměrně jednoduchá, není nijak komplikovaně zanořovaná (jak jsme již viděli v kapitole 3.1.2). Uvnitř elementu `<s>`, representujícího větu, jsou jednotlivá slova označena `<m>`. Elementy uvnitř `<m>` jsou:

[12] Viz [10], str. 605.

`<src.rf>`: původ anotace (manuální);

`<w.rf>`: odkaz na rovinu slovní;

`<form>`: přesný tvar slova ve větě;

`<lemma>`: zlemmatisované slovo, jeho základní tvar a

`<tag>`: morfologická značka obsahující veškerou informaci o tvarosloví.

Přidali jsme tedy nový element `<ewn-sense>`, který obsahuje literál podle Českého WordNetu spolu se slovním druhem, číslem označující význam a s ID synsetu. Výsek souboru vypadá tedy po úpravě takto:

```
<m id="m-cmpr9410-041-p13s3w6">
    <src.rf>manual</src.rf>
    <w.rf>w#w-cmpr9410-041-p13s3w6</w.rf>
    <form>trhu</form>
    <lemma>trh</lemma>
    <ewn-sense>trh--n-3==608607</ewn-sense>
    <tag>NNIS2-----A----</tag>
</m>
```

A nyní přikročme k popisu samotného skriptu, který toto namapování a vložení provede, ovšem opět velice stručně a v bodech, podrobnější informace patří do programátorské dokumentace, v příloze B.2.

- vezme výstup z předchozího skriptu
- nalezne správná místa v PDT 2.0 na m-rovině
- umístí tag ve tvaru `<ewn-sense>kompletni znacka--n-1==342623</ewn-sense>`
- výstup uloží do souboru se stejným jménem a příponou `.e`

3.2.3 Přizpůsobení PML (`link_fake_data`)

Už zbývá jenom nově vytvořené soubory s příponou `.e.gz` začlenit do celého PDT. Nepřišlo nám vhodné zasahovat a pozměňovat přímo samotné PDT, proto jsme to obešli tím, že jsme vytvořili linky ze souborů `.e.gz` na soubory `.m.gz`, takže může být s novými soubory nakládáno jako se soubory m-roviny. Tím se dosáhlo zachování souborů bez našeho zásahu a zároveň potřebného stavu dat.

Takto však již data neodpovídají schematu formátu PML, který vychází z XML a definuje formát dat pro PDT 2.0. Pro práci s tredem a podobnými nástroji je PML zásadní, bylo tudíž ještě nutné upravit novou specifikaci formátu v souborech `resources/mdata_schema.xml` a `resources/adata_schema.xml`. Na příslušné místo byl přidán řádek pro element `<ewn-sense>` a pomocí atributu `required="0"` nevyžadujeme jeho přítomnost (což je nutné pro všechna neanotovaná slova a pro ta, kde se anotátoři neshodli). Příslušná část specifikace tedy vypadá následovně:

```
<member name="form" required="1"><cdata format="any"/></member>
<member name="lemma" required="1"><cdata format="any"/></member>
<member name="ewn-sense" required="0"><cdata format="any"/></member>

<member name="tag" required="1"><cdata format="any"/></member>
```

Kapitola 4

Automatické přiřazování synsetů

4.1 Specifikace úlohy

Úlohu jsme si specifikovali jako přiřazení synsetu z vlastního slovníku zadaným výskytům lemmat. Co to přesne znamená?

Vlastní slovník. Pro naši úlohu nepoužíváme celý ČWN, ale pouze jeho část, a to ty synsety, které byly někdy použity v trénovacích datech a na nichž se oba anotátoři shodli. Vzniklý slovník pro účely tohoto textu nazvěme PDT-ČWN. Přestože původní Český WordNet není závislý na datech a vznikl nezávisle na nich,[1] rozhodli jsme se pro tento postup vzhledem k argumentům v následujícím odstavci.

Zadané výskyty. Rozhodli jsme se, že vstupem našemu přiřazovacímu algoritmu nebude proud slov, ze kterých by se teprve vybíralo, která jsou víceznačná a co jim přiřadit. Místo toho jsme vstup očistili od všech nezajímavých slov (z pohledu přiřazování synsetů) a úkolem programu tedy bude přiřadit význam všem, které byly se shodou anotovány anotátory. Dovolíme si krátký rozbor tohoto rozhodnutí. V principu by sice bylo možné vybírat slova vlastním algoritmem založeným na tom, jaké je to lemma a zda jsme ho kdy viděli v trénovacích datech – ale to by nemělo moc velký smysl, protože bychom pak výsledky neměli s čím porovnat (ve všech případech, kdy by byl přiřazen synset slovu, které anotátoři třeba i anotovali, ale neshodli se). Pak ale nemůžeme říct vůbec nic o tom, jaký je správný význam. Respektive za správné přiřazení bychom museli považovat pouze příznak „nedesambiguováno“, přestože by nejspíš neexistovalo žádné pravidlo, jak poznat, kde se anotátoři shodli (= přiřazovat) a kde nikoli (= přeskakovat).

Když si dovolíme určitou abstrakci a připustíme, že jako arbitři bychom znali ten správný význam, mohlo

[1] Až v době psaní této práce se začal slovník mírně upravovat a začaly se zohledňovat různé nesnáze a nedostatky při anotaci.

by se snadno stát, že by byl strojově přiřazen a přesto vyhodnocen jako chybný: jen proto, že lidští anotátoři „selhali“.

Nyní je také vidět zmíněný důvod, proč nepoužíváme celý ČWN: výsledky by opět nebylo s čím porovnávat.

Toto rozhodnutí má vliv i na způsob měření úspěšnosti. Nemáme totiž žádný případ „přiřazeno X, když nemělo být přiřazováno nic“. Z definice obvyklých měr precision a recall plyne, že pro takto definovanou úlohu tato čísla splývají. Budeme tedy uvádět jen procentuální úspěšnost

$$\frac{\text{správně přiřazené}}{\text{všechny zpracovávané}} \cdot 100\,\%.$$

4.2 Charakteristiky dat

V této části se podíváme na strukturu dat, seznámíme čtenáře s mezianotátorskou shodou projektu, jehož výsledky používáme, spočítáme některé důležité charakteristiky a hlavně určíme baseline, která pak bude kritériem pro náš vlastní experiment. Poté se ještě vrátíme k některým charakteristikám a dovolíme si již pohled i do testovacích dat.

4.2.1 Mezianotátorská shoda

Náš experiment probíhal na týchž datech, o kterých pojednává článek [1] o dosažených výsledcích anotace PDT pomocí ČWN, o revizích výsledků a o některých opravách frekventovaných chyb.[2] Proto začneme statistikou z tohoto článku. Jak jsme uvedli v tabulce 3.1, anotací dosud prošlo 507 470 tokenů. Kolik z nich bylo anotováno, ukazuje tabulka 4.1. Nepatrně se liší od té v článku [1], neboť jeden soubor[3] musel být odstraněn (kvůli problémům diskutovaným v kapitole 3.1.2, část 3). V tabulce zmiňovaná „autosémantická slova“ pro nás představují substantiva, adjektiva, verba a adverbia (z anglických ekvivalentů také zkratka NAVD), ovšem adverbia se v ČWN nevyskytují.

Nás tedy bude zajímat oněch 148 547 slov, protože z nich jsme vytvořili konečnou podobu dat, na které natrénujeme a otestujeme rozhodovací stromy. Více než 69 % z nich jsou substantiva, 10 % adjektiva a zbylých 21 % jsou verba.

[2] Tyto opravy probíhaly současně s naší prací, proto jejich výsledky nebyly ještě zohledněny v datech našeho experimentu.

[3] Konkrétně soubor `c141`.

[4] Jedná se o víceznačnost lemmatu v ČWN. Je to tedy jiná víceznačnost, než nastává potom v našem experimentu se strojovým přiřazováním významu: protože v tomto druhém případě už půjde o slovník restringovaný jen na ty významy, které byly anotátory použity. Stručně řečeno: anotátor vybíral z většího počtu synsetů (vyšší počet víceznačných lemmat), stroj pak bude vybírat ze zúžené množiny PDT-ČWN.

	Abs. č.	% slov	% NAVD	% anot.
Počet tokenů	507 470			
Počet všech slov	430 921	100 %		
Počet autosémantických slov	300 343	69,7 %	100 %	
Počet anotovaných slov	148 547	34,5 %	49,5 %	100 %
Počet víceznačných[4]slov	116 530	27,0 %	38,8 %	78,4 %

Tabulka 4.1: Počty slov v anotovaných datech

Na těchto téměř 150 000 slov dosáhli dva anotátoři shody 61,7 %. Toto číslo výrazně kolísá pro různé slovní druhy. Pro anotovaná substantiva, adjektiva a verba vychází 66,0 %, 67,0 % a 44,8 % (v tomto pořadí). Počítáme pouze shodu na synsetu (tj. anotátor A vybral synset, nikoli chybu, a anotátor B vybral tentýž synset); kdybychom uvažovali i shodu na chybové kategorii (viz kapitola 3.1.2, část 3), dosáhneme 74,6 % (pro všechny tři slovní druhy dohromady).

4.2.2 Statistický pohled na trénovací data

K disposici jsme měli tři anotované slovní druhy: substantiva, adjektiva a verba. Rozhodli jsme se slovesům prozatím nevěnovat, neboť očekáváme, že ta jsou znatelně lépe zpracována v PDT-VALLEXU, kde už se také jejich přiřazování delší dobu testuje ([16] a [2]), navíc na nich byl výrazně nižší mezianotátorská shoda. Jde tedy v této práci o dva úkoly, jejichž úspěšnost budeme měřit odděleně:

- přiřazování významu (skrze synsety) všem substantivům, která jsou v PDT-ČWN (třebaže tam mají jen jediný význam) a
- přiřazování významu všem adjektivům z PDT-ČWN.

Data rozdělíme na trénovací a testovací a budeme postupovat následovně. Použijeme dělení tak, jak je provedeno v PDT 2.0 (kapitola 2.1.2), tedy zachováme stejné soubory ve stejných deseti skupinách. Dvě desetiny dat (testovací a evaluační) dáme prozatím stranou a budeme zkoumat pouze trénovací data (adresáře `train-1/` až `train-8/`). Nadále tedy budeme hovořit pouze o nich. Poté na jejich základě zvolíme způsob výpočtu baseline a tento výpočet provedeme na „development test“ datech, neboli na adresáři `dtest/`. Data pro evaluační testy (adresář `etest/`) necháme až do samého konce experimentování nedotčené. Podívejme se tedy na některé statistické údaje, které nám snad přiblíží povahu obou úloh (tedy úlohy se substantivy a úlohy s adjektivy). Nejprve trocha terminologie, kterou budeme používat:

lemma základní tvar slova; jak již bylo řečeno, mluvíme pouze o lemmatech, která jsou zastoupena ve vlastním slovníku PDT-ČWN a budeme jim přiřazovat význam;

výskyt instance lemmatu z PDT-ČWN; každému výskytu je přiřazován význam;

substantivum, adjektivum jde samozřejmě o slovní druh, ale „všechna substantiva" bude znamenat skutečně všechna v (trénovacích) datech se vyskytující; jedná se o širší skupinu než množina *výskytů*, jež je odvozena od *lemmatu* (tedy od lemmatu z PDT-ČWN);

význam možnost, která se pro lemma nabízí v PDT-ČWN, též synset; jeden z významů je pak danému výskytu lemmatu přiřazen;

jednoznačné použito vždy jen o lemmatu z PDT-ČWN; takové lemma, které má ve slovníku pouze jediný význam – není tedy co desambiguovat;

víceznačné opak „jednoznačného", takové lemma, které se v PDT-ČWN vyskytuje anotované více významy, je tedy z čeho vybírat, je co desambiguovat;

trénovací data část dat PDT, která jsou určena k trénování (viz kapitola 2.1.2), přibližně 8/10 dat: adresáře `train-1/` až `train-8/`; některé údaje uvádíme zvlášť pro trénovací data a zvlášť pro testovací, jiná mají smysl jen pro trénovací data;

testovací data asi 1/10 dat PDT (viz kapitola 2.1.2), určená k vývojovému testování (myšlen adresář `dtest/` – neboť `etest/` zatím nepoužíváme);

celá data trénovací data sjednocená s testovacími (stále bez evaluačních testovacích dat v adresáři `etest/`).

Substantiva se jeví jako zajímavější objekt zájmu (přinejmenším proto, že jich máme k disposici skoro sedmkrát více), proto se teď budeme chvíli zabývat hlavně jimi (také pro větší čitelnost následujícího textu). Obdobná čísla pro adjektiva lze dohledat v tabulce 4.2.

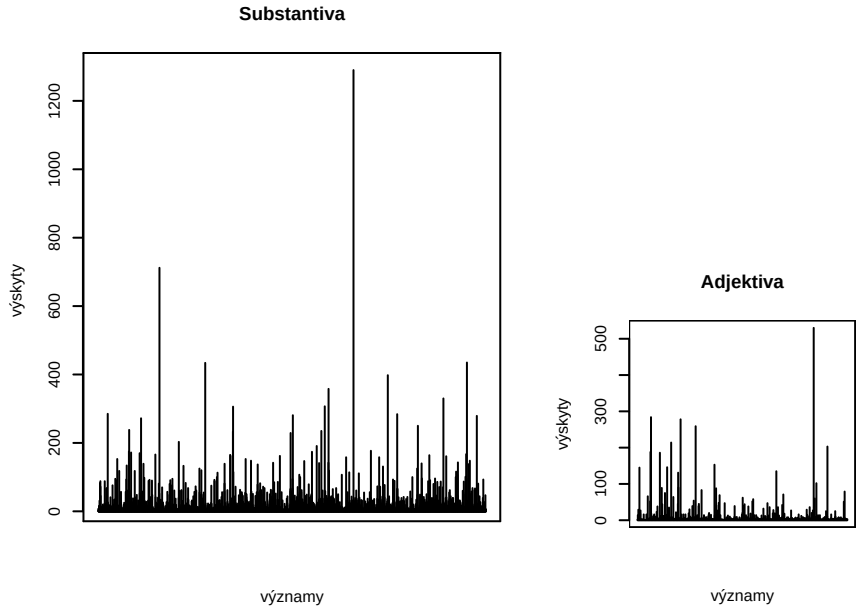
V trénovacích datech se vyskytuje 4 110 různých *lemmat* substantiv z PDT-ČWN. Pro tato lemmata tedy natrénujeme 4 110 rozhodovacích stromů, které by měly co nejlépe přiřazovat správný význam každému z nich.

Dále se podívejme na *výskyty* lemmat. Pro natrénování rozhodovacích stromů máme k disposici 53 128 výskytů substantiv (je to 6,8krát více než pro adjektiva; lemmat substantiv je 6,5krát více než adjektiv). Přitom *všech substantiv* je v naší části 121 127 a anotováno mělo být přibližně 81 000 substantiv.

Poměr mezi všemi substantivy a těmi, které měly záznam ve WN, a tudíž byly anotovány, celkem přesně odpovídá 67 % z [1] měřeným na všech datech. Poměr mezi anotovanými a těmi, které se dostaly do našeho

experimentu, zase kopíruje 66% mezianotátorskou shodu na substantivech. Tedy trénovací data odpovídají rozložení celkovému a jsou v tomto ohledu representativní.

Každé lemma jsme při trénování spatřili v průměru téměř třináctkrát.

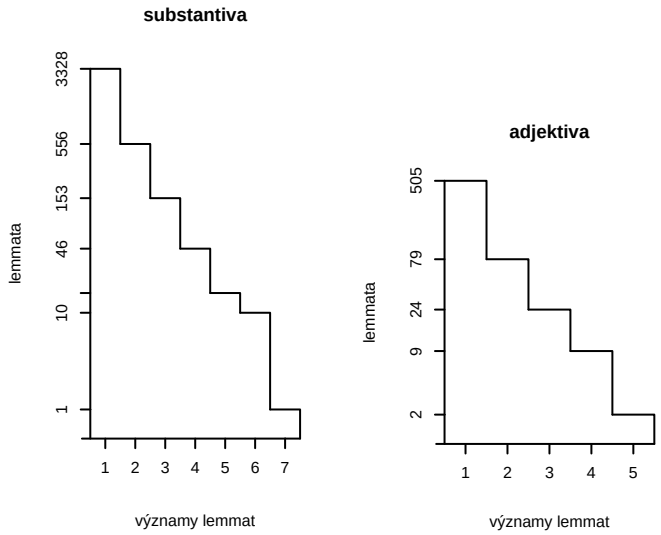


Obrázek 4.1: Počet výskytů jednotlivých významů

Nakonec prozkoumejme počty *významů*. Spočítáme míru víceznačnosti anotovaných lemmat[5]. Průměrně na jedno substantivum připadá 1,27 významu, z nichž je třeba vybírat. Každý význam je v průměru (při trénování) spatřen asi desetkrát, nicméně počty výskytů jednotlivých významů se velice liší, jak ukazuje obrázek 4.1. Počet použitých významů kolísá od jednoho do sedmi, rozložení je na obrázku 4.2. Lemmat s jediným významem je 3 328 (z 4 110). Zmíněných 1,27 významu na lemma není mnoho. Pokud ale zúžíme pohled jen na víceznačná lemmata, zvýší se průměrný počet významů na 2,43. Každý z nich je v trénovacích datech spatřen skoro třináctkrát.

Všechna uvedená čísla shrnuje tabulka 4.2, ve které jsou uvedeny i údaje pro adjektiva.

[5]Přesněji: oanotovaných lemmat. Nebudeme počítat, jaká byla míra víceznačnosti při anotování (tj. průměrný počet synsetů v ČWN připadající na jedno lemma), ale jaká je míra víceznačnosti v naší úloze automatického přiřazování (tj. průměrný počet použitých synsetů z PDT-ČWN na jedno lemma).



Obrázek 4.2: Zastoupení lemmat s daným počtem významů (osa y logaritmická)

Substantiva	Adjektiva	Popis
4 110	619	počet lemmat
53 128	7 861	počet výskytů
121 127	12 012	počet všech substantiv/adjektiv
12,93	12,70	průměrný počet výskytů na lemma
3 328	505	počet jednoznačných lemmat
1,27	1,26	průměrný počet významů na lemma (viz též obr. 4.2 pro absolutní hodnoty)
2,43	2,42	průměrný počet významů na víceznačné lemma
10,16	10,07	průměrný počet výskytů jednoho významu (viz též obr. 4.1 pro absolutní hodnoty)
12,79	12,85	průměrný počet výskytů jednoho významu víceznačného lemmatu

Tabulka 4.2: Frekvenční zastoupení v trénovacích datech

4.2.3 Výpočet baseline

Baseline, vůči které budeme moci porovnávat dosažené výsledky, získáme prostým přiřazením synsetu, který byl nejčastější v trénovacích datech, ke všem výskytům v testovacích datech. Výsledky budeme mít dva a jejich význam si dovolíme vysvětlit až v následující kapitole 4.2.4. Baseline je v tabulce 4.3 a pro nás je zatím důležitý pouze první řádek. Absolutní čísla vypadají přibližně následovně: 700 substantiv ze 7 000 bylo určeno špatně (tedy mělo jiný význam než nejčastější) a v adjektivech chybovalo 60 z 1 000.

nová lemmata	substantiva	adjektiva
vyřazená – zanedbaná	89,866 %	94,741 %
jako chyby	87,362 %	92,060 %

Tabulka 4.3: Baseline pro přiřazování synsetů, pokud „nespatřená“ lemmata vyřazujeme, resp. započítáváme jako chyby

Čtenář si jistě povšiml, že i takto primitivním způsobem „desambiguace“ bylo dosaženo překvapivě vysoké úspěšnosti. O čem to svědčí? V zásadě o tom, že na těchto datech je desambiguace snadná, ale její zlepšení bude hodně náročné a malé. Je jasné, že anotátoři ve většině případů vybírali tentýž synset pro dané lemma. Důvody mohou být různé. Například:

- **slovník:** nedostatečně specifikované distinkce ve slovníku a tudíž nutnost uchylovat se k nějakému obecnějšímu významu;
- **anotátoři:** je snazší vybrat obecný význam, než muset rozhodovat o jemných nuancích, i když je slovník v pořádku;
- **významy:** dále se tu naskýtá obvyklá otázka, jak volit hranici mezi významy – extrémem na jedné straně je jediný význam (tudíž žádná desambiguace a žádné úspěchy v ní), na druhé pak takové rozdrobení, že ani autor slovníku u konkrétního výskytu lemmatu není schopen určit, o který z jeho významů se jedná (a tedy častá volba nějakého obecného významu a z toho vyplývající desambiguace nepříliš se lišící od baseline);
- **data:** musíme mít stále na mysli, že data pocházejí z ČNK a dokonce všechna z novinových článků; tudíž nejsou dostatečně representativní: i kdyby například slovo „koruna“ mělo rovnoměrné zastoupení jako měna i jako ozdoba na hlavu, je téměř jisté, že v novinách bude drtivě převládat jako platidlo;
- **jazyk:** také je možné, že nedošlo k žádnému pochybení: že totiž i když má slovo mnoho významů, používá se téměř výhradně jen v jediném;

- **vágnost:** když autor tvoří nějaký text, nerozmýšlí vždy, v jakém významu které slovo používá (a už vůbec nerozmýšlí s WordNetem v ruce): často je mu lhostejné, kterého z několika podobných významů slovo nabude – a nebo je určitá víceznačnost dokonce jeho (uměleckým) záměrem; co ale potom může být přiřazeno jako význam než opět nějaký universální.[6]

K tomu je ještě důležité uvést jedno pravidlo, kterého se měli anotátoři držet: totiž měli maximálně zachovávat konsistenci výběru synsetu. Takže pokud si nebyli jisti, co vybrat, měli rozhodně dodržet tento výběr ve všech podobných případech. Přesto se (po konsultaci s PhDr. Petrou Möllerovou, zabývající se kontrolou anotací) přikláníme k domněnce, že skutečně toto nerovnoměrné rozložení významu v datech nastává. Je jisté, že jiným (ne tolik jednostranným) výběrem dat do korpusu by se rozložení pozměnilo a nejfrekventovanější význam by nebyl tolik častý. Nejsme však s to rozhodnout, jak významný by tento posun byl a zda by se rozložení mohlo aspoň přibližně vyrovnat.

4.2.4 Statistiky na testovacích i trénovacích datech

Nyní si již můžeme dovolit poodkrýt roušku zakrývající testovací data, nahlédnout na některé očekávané jevy a kvantifikovat je. Tak předně je potřeba zmínit, že je pochopitelné, že testovací data přinesla některé případy v trénovacích datech zatím neviděné. A to jak nová lemmata, tak nové významy původních lemmat. Začneme tím, že tento fakt (který jsme očekávali, protože z podstaty rozložení jevů v jazyce je pravděpodobné, že některé méně časté jevy padnou výhradně do oblasti testovacích dat) přiblížíme číselně. Poté se rozhodneme, jak s ním naložit.

V celých datech (tím v této kapitole myslíme `train-1/` až `train-8/` s nově přidaným `dtest/`) se vyskytuje celkem 4 270 různých anotovaných lemmat. Oproti 4 110 z trénovacích dat je to o 160 více: o ta lemmata, která v trénovacích datech vůbec nebyla[7]. Otestovat, jak dobře se přiřazování synsetů naučíme, budeme moci na 7 359 výskytech (o trochu méně než 1/7 trénovací množiny[8]). Z nich ale 205 výskytů náleží těm 160 lemmatům, která nebudou po natrénování známa.

Podívejme se ještě na významy. Míru víceznačnosti (tedy počet významů na lemma) na testovacích datech nemá dobrý smysl měřit (když se v testovacích datech nějaký význam nevyskytne, ani v nejmenším to nesnižuje složitost úlohy). Zajímavější je počet lemmat

[6] Mluvíme ve všech případech o situaci, kdy se (náhodou?) anotátoři shodnou – častěji (ale ne vždy) se asi stane, že nezvolí stejně a výskyt se do našich dat nedostane.

[7] To platí pochopitelně i opačně, mnoho lemmat se v trénovacích datech program naučí určovat, ale v testovacích se neobjeví, takže tato schopnost zůstane neotestována a vyjde vniveč.

[8] Kdyby byla anotovaná lemmata rozložena rovnoměrně, měla by to být jen 1/8 – osm dílů PDT jsou trénovací data, jeden jsou testovací, viz kapitola 2.1.2.

s jediným významem v celých datech: jednoznačných lemmat je 3 435 (z celkových 4 270). Asi je důležité si uvědomit, že oproti jednoznačným lemmatům z trénovacích dat se počet změnil dvojím způsobem: jednak přibyla nová jednoznačná lemmata, jednak ubyla ta, kterým přibyl v testovacích datech nový význam. Druhá skupina čítá přesně 52 lemmat. Celkově – jednoznačným i víceznačným lemmatům z trénovacích dat – přibude v testovacích datech 95 významů – tedy cílových synsetů, které by měly být přiřazeny, ale nemohou být, protože při trénování nebyly spatřeny a nelze je predikovat (připomínáme rozhodnutí o zúžení ČWN na PDT-ČWN).

Všechna zmíněná čísla jsou přehledně (a hlavně spolu s obdobnými čísly pro adjektiva) uspořádána v tabulce 4.4.

Substantiva	Adjektiva	Popis
Lemmata a výskyty		
4 270	654	počet lemmat celkem
1 698 / 7 359	292 / 1 272	počet lemmat / výskytů lemmat v testovacích datech
160 / 205	35 / 36	počet lemmat / výskytů lemmat z trénovacích dat neznámých
Nové významy v testovacích datech (pro lemmata z trénovacích dat)		
95 / 91	9 / 9	počet přibyvších významů / lemmat s přibyvším významem
3 435	533	počet jednoznačných lemmat napříč celými daty
pro lemmata v trénovacích datech jednoznačná		
53 / 52	7 / 7	počet přibyvších významů / lemmat s přibyvším významem
Celá data, train, dtest i etest		
153 032	58 816	počet všech substantiv / adjektiv
102 694	15 329	počet všech subst. / adjektiv k manuální anotaci

Tabulka 4.4: Frekvenční zastoupení v testovacích a v celých datech

Jaký je správný postoj k oněm 160 „neviděným“ lemmatům? Jak jsme zmínili v kapitole 4.1, nerozhodujeme, která slova desambiguovat a která nikoli, proto bez spatření těchto lemmat při trénování nemáme mechanismus, jak jim cokoli přiřadit.

Ve stručnosti rozeberme, jaké bychom měli možnosti. Za předpokladu, že bychom o každém slovu nejprve rozhodovali, zda je určené k desambiguaci, a teprve pak mu přiřadili význam, mohli bychom přiřazovat významy i těmto neznámým lemmatům. Otázkou je, co bychom jim přiřazovali. Při trénovaní se neobjevila ani jednou, takže bychom museli zvolit nějakou víceméně náhodnou metodu: přiřadit vždy první synset,

přiřadit náhodný synset apod. Ale z čeho vybírat náhodný synset? Muselo by to být z celého ČWN, ne jen z našeho pracovního PDT-ČWN, vytvořeného jen ze spatřených synsetů. Řekněme, že bychom tedy takto změnili rozhodnutí a pracovali s celým ČWN. Rozmyslíme nyní, jak by se změnily výsledky. Zvětšili bychom tedy objem dat, na kterých bychom testovali. A pro tuto novou množinu bychom již nepoužívali rozhodovací stromy, ale vlastní implementaci některého zmíněného triviálního algoritmu. A výslednou úspěšnost bychom porovnali s novou baseline. Ta by měla také testovací data rozšířená – a jaký algoritmus bychom použili tam? Nejspíš tentýž, protože nic „jednoduššího" asi neexistuje. Výsledkem by tedy bylo snížení rozdílu mezi baseline a výsledkem našeho experimentu. A náhradou bychom nezískali nic.

Při měření úspěšnosti lze volit mezi dvěma přístupy (a tím se vracíme k tabulce 4.3): Buď nová lemmata úplně zanedbáme, vypustíme je z testovacích dat, jako by tam vůbec nebyla (první řádek tabulky), a nebo každý výskyt některého z těchto 160 lemmat budeme považovat za chybně přiřazený (protože žádný) rámec (druhý řádek tabulky). Je vidět, že obě čísla spolu budou úzce souviset: máme-li podíl správně přiřazených ku všem, tak při změně výpočtu baseline od „zanedbávání" k „započítávání" jednoduše zvětšíme jmenovatel o 160. Stejným způsobem upravíme i jmenovatel úspěšnosti našeho pokusu. Výsledkem bude „ohlazení rozdílu" mezi těmito dvěma porovnávanými čísly. Navíc: zanedbáme-li neviděná lemmata, je tu možnost (sice jen teoretická) dosáhnout 100 %. Při druhém způsobu počítání se nám sníží horní hranice, kterou nelze přesáhnout. Budeme tedy po většinu času neviděná lemmata při výpočtech ignorovat.

S nově přibyvšími významy je to snazší: protože lemmata, ke kterým patří, z trénovacích dat známe a budeme je tedy desambiguovat, jednoduše se nám nepodaří přiřadit správný synset a bude to započítáno jako chyba. Což je to, co bychom intuitivně očekávali.

4.3 Vlastní experiment

Tato kapitola se bude zabývat naším postupem (zvolenou technikou, použitými nástroji, výběrem rysů) vedoucím k automatické desambiguaci a v následující uvedeme dosažené výsledky v porovnání s baseline.

4.3.1 Trocha teorie

Zvolili jsme jednu z metod strojového učení, a to metodu rozhodovacích stromů. Stručně nastíníme, v čem spočívá. Program zajišťující strojové učení dostane na vstupu velké množství n-rozměrných vektorů. Prvních $n-1$ složek obsahuje *rysy*, poslední složku budeme nazývat *cíl*, tedy to, co se program má naučit a posléze přiřazovat. Rysy by měly být voleny tak, aby jich byl dostatek a byly relevantní pro určení poslední složky, tedy cíle, jímž je v naší úloze

synset.[9] Program prochází jednotlivé vektory, hledá závislosti a staví *rozhodovací stromy*, které co nejlépe vystihují závislosti mezi různými rysy a cílem.

Rozhodovací strom je takový strom, kde jsou vnitřním uzlům přiřazeny rysy, hranám jejich hodnoty a listům přiřazeny cíle. Při rozhodování se prochází stromem od kořene a pro další postup se vždy vybere ta hrana, která odpovídá hodnotě rysu v daném vrcholu zkoumaném. Příchod do listu znamená nalezení hledané hodnoty – cíle.

Proces stavění rozhodovacího stromu má dvě fáze: samotnou stavbu a následné prořezávání. Při stavbě se postupuje tak, aby rysy přinášející největší informační zisk (tj. rozdělující množinu případů do přibližně stejně velkých skupin) byly co nejblíže kořeni stromu.

Když se tímto způsobem celý strom postaví, začne se s prořezáváním stromu do výsledné podoby. To znamená, že pro vybrané uzly se odřízne celý jejich podstrom: tím se z uzlu stane list a jako hodnota se mu přiřadí ta, která byla nejčastější pro trénovací případy spadající do jeho podstromu. Uzly se vybírají tak, aby se po jejich odříznutí úspěšnost na vybraném vzorku dat (z trénovacích) nezhoršila. Důvodem pro prořezávání je eliminace chyb v datech a zabránění přeučení, když je dat nedostatek. Výsledné trénovací stromy jsou poměrně robustní vůči chybám a šumům v trénovacích datech. [11]

Aby mohl experiment proběhnout, je potřeba dvou věcí: programu, který se o trénování postará, a vybrání vhodných rysů, které přinesou nejlepší výsledky. Pro trénování jsme využili již existujícího programu C4.5 (viz kapitola 2.5), který implementuje strojové učení právě na základě rozhodovacích stromů. Ten navíc vyžaduje přesně vymezit množinu hodnot (byť třeba neomezenou), jichž může nabývat každá z n složek vektoru.

4.3.2 Konkrétní postup

Podívejme se zblízka, co přesně vyžaduje C4.5 na svém vstupu a jakým způsobem tato data opatříme. Pro jednu úlohu je potřeba dodat soubory `uloha.data`, `uloha.test` a `uloha.names`. První dva obsahují na řádcích jednotlivé vektory (hodnoty oddělené čárkami), třetí obsahuje názvy jednotlivých rysů (složek vektoru) a vymezení jejich hodnot (přičemž neakceptuje rysy s jedinou možnou hodnotou). Pomocí souboru `uloha.data` si C4.5 vybuduje rozhodovací stromy a následně je použije k přiřazení hodnot vektorům ze souboru `uloha.test`. Tyto hodnoty obratem porovná s poslední složkou vektoru v témže souboru (kterou pochopitelně do tohoto okamžiku nepoužívala) a vyhodnotí úspěšnost (jako error rate, tady procento chyb).

Zmíněná `uloha` pro nás bude vždy jedno jediné lemma, neboť je nutné pro různá lemmata s různými významy a různým chováním natrénovat také různé rozhodovací stromy. Náš záměr je tedy vytvořit tolik souborů *`lemma`*`.data` a *`lemma`*`.test`, kolik se v trénovacích datech vyskytuje lemmat, vybrat $n-1$ rysů, které považujeme za potenciálně užitečné, naplnit

[9] Dovolíme si místo samoúčelného zavádění nových termínu označovat některé součásti metody strojového učení rovnou pojmy z naší úlohy desambiguace. Podřídíme zkrátka popis obecné metody jejímu dalšímu použití v tomto textu. Věříme, že to čtení usnadní, aniž by to zároveň zamlžilo popisovanou techniku.

je daty z PDT a přidat příslušný synset. Konkrétně z trénovací části vyrobit *lemma*`.data` a z dtestu *lemma*`.test` (etest leží až do konce experimentů stranou a neprovádí se na něm žádné pokusy: toto již nebudeme zdůrazňovat a termínem *testovací* budeme označovat data pro vývojové testování). Dále s pomocí hodnot v souboru *lemma*`.data` vyrobit jejich seznam a tím soubor *lemma*`.names`. Potom již budeme moci spustit C4.5 postupně na všechna lemmata z trénovacích dat, změřit úspěšnost na testovacích datech – a proces opakovat s novou sadou rysů.

Zde padla dvě rozhodnutí:

1) Rysy budeme vybírat ručně (a testovat jejich užitečnost). Jiný postup je natrénování vedlejší úlohy strojového učení, která vybírá z obrovské škály rysů ty, které dosahují nejlepších výsledků v cílové desambiguační úloze. Tím, dá se říci, supluje práci, kterou jsme se rozhodli dělat (alespoň prozatím) ručně.

2) Rysy zvolíme společné pro všechna lemmata (zvlášť jen pro slovní druhy, ačkoli ty dvě množiny se nebudou lišit příliš). Není principiálně vyloučené nalézt vhodnou množinu rysů pro každé lemma, je to ovšem práce mnohem náročnější a hlavně by měla být založena na větším množství dat, než máme k disposici.

Ponořme se v popisu ještě hlouběji a projděme pět fází, ze kterých sestává každý jednotlivý experiment. Budeme však struční, protože podrobné informace přináší příloha C (C.2 až C.6).

V **první** fázi projdeme všechna zvolená data PDT (typicky všechna z trénovací a z testovací množiny), nalezneme všechna anotovaná slova (tedy slova s přiřazeným synsetem ve značce `<ewn-sense>`) a postupně pro každé toto slovo naplníme vektor požadovanými rysy. Seznam rysů zvlášť pro každý slovní druh se specifikuje v konfiguračním souboru `rysy_tred`, tedy odděleně od programu. Výsledkem je jediný soubor *PoS*`.data` s vektory pro všechna lemmata odpovídající specifikovanému slovnímu druhu.

Zmínili jsme již v kapitole 3.1.2, že jen malá část naších dat má anotaci na t-rovině. Přestože jsme začali pokusy právě s rysy z t-roviny (kap. 4.5), museli jsme ji opustit, protože dat bylo vskutku nedostatečně. Mluvíme-li tedy o získání rysů z PDT, jedná se o rysy z a-roviny a z m-roviny. Uzly, ze kterých rysy vybíráme, je desambiguovaný uzel sám a jeho syntaktické okolí (tedy získané na a-rovině). Jak se ukazuje z [16], přímé, lineární okolí uzlu ve větě nepomáhá tolik jako okolí ve stromě, ale hlavně téměř vůbec nepomáhá spolu se syntaktickým.

Druhý krok je projít tento soubor vektorů *PoS*`.data` a pro každou složku vektoru, tedy rys, sestavit seznam všech hodnot, kterých může nabývat. Z těchto seznamů vyrábíme soubor `universal.names` ve formátu, jaký požaduje C4.5. Zdůrazňujeme, že soubor vzniká pouze na základě trénovacích dat.

Tady bychom rádi upozornili, že jsme si dovolili zůstat při tomto universálním souboru a nevyrábět takový soubor pro každé lemma zvlášť. Ze všech souborů *lemma*.names vedou symbolické linky na tento universální. Dochází tím k nadgenerování, což ovšem nevadí; důležité je, že se tam každá hodnota rysu objeví, neboť obsahuje sjednocení všech dílčích souborů. Ušetří se tím mnoho diskového prostoru i procesorového času.

Třetí fází je roztřídění souboru *PoS*.data podle lemmat do jednotlivých souborů *lemma*.data. V názvu souboru rušíme případnou diakritiku, správný tvar lemmatu je v souboru na prvním řádku jako komentář. Protože syntaxe souboru rysy_tred umožňuje nechat vygenerovat více rysů a ne všechny pak použít k natrénování (užitečné hlavně pro pomocné hodnoty při testování výsledků, např. ID uzlu, díky němuž je možné zobrazit syntaktický strom v TrEdu apod.), třetí fáze je tím okamžikem, kde dojde k vypuštění těchto rysů.

Nutno zmínit, že pro testovací data se chováme trochu jinak. Než vložíme hodnotu rysu do výsledného souboru, zkontrolujeme, zda se nachází v universal.names. Pokud ne, nahradíme jeho hodnotu ? (otazník), což C4.5 umožňuje (za jiným, ale v principu podobným účelem: označují se tak hodnoty, které neznáme – což je svým způsobem tento případ).

Následně vytvoříme všechny symbolické linky na universal.names.

Čtvrtá v pořadí je samotná C4.5 spuštěná postupně na všechna lemmata, která se vyskytla i v trénovacích i v testovacích datech. Z výstupu je pro nás důležitý počet případů a počet chyb.

V **páté** fázi projdeme výsledky C4.5 a spočítáme výslednou úspěšnost. Zároveň při stejném průchodu počítáme na aktuálním vzorku dat baseline.

4.3.3 Zlepšování výsledků

Začali jsme jednoduchou a malou množinou rysů[10], abychom získali první výsledky. Ty byly dosti neuspokojivé, neboť jen o málo (asi 0,3 procentního bodu) převyšovaly baseline. Určili jsme dva důvody:

- První byl docela zjevný: baseline je příliš vysoká. Když prostým přiřazením nejčastějšího dosáhneme tak vysokých výsledků, nedá se očekávat, že se to ještě o mnoho vylepší.
- Bližším ohledáním výsledků z C4.5 jsme zjistili, že se rozhodovací stromy v naprosté většině případů redukují na jediný uzel. Program C4.5 totiž úspěšně[11] implementuje algoritmus prořezávání. V našem případě to ovšem znamená, že strom prořezal až na

[10] Pouze základní morfologické rysy a pouze pro dané lemma, tedy bez využití syntaktických vztahů ve větě. Kompletní seznam používaných rysů je v příloze D.

[11] Úspěšně znamená, že jednak výsledné stromy jsou menší, což je jistě u mohutných stromů výhoda, a jednak ačkoli testy na trénovacích datech se tím lehce zhorší, výsledky na testovacích datech dopadají lépe, což potvrzuje, že odříznuté větve byly matoucí a neměly ve stromě co pohledávat.

jeden jediný uzel, list. Hodnotou toho listu pak je jediný přiřazovaný synset – ten nejpravděpodobnější. Takže v této fázi se naše výsledky málo odlišovaly od baseline, která dělá totéž pro všechna lemmata.

Usoudili jsme, že když s prvním důvodem mnoho nepořídíme, budeme muset zařídit, aby C4.5 dostala tolik rysů a tak relevantních, aby postavila stromy s „pevnými větvemi“, které by pak nepodlehly prořezávání. Zaměřili jsme se na zmnohonásobení počtu rysů pro lemma. Úspěšnost se zvyšovala, ale zdaleka ne vždy, některé rysy byly zřejmě zavádějící a výsledná čísla jimi byla dokonce „stržena“ zpět na horší posici.

Nebudeme zde rysy rozebírat podrobně (seznam je v příloze), ale popíšeme pouze typy, se kterými jsme experimentovali. Rysy byly prakticky totožné pro substantiva i adjektiva, i pokusy provádíme naráz a již to nebudeme zmiňovat. V základní sadě jsme měli některé významnější morfologické příznaky (slovní druh, pád, číslo, rod, ...) a analytickou funkci (tj. Sb, Atr, Adv apod.) pro desambiguované slovo, pro jeho efektivního[12] otce, efektivního syna a pro levé i pravé bratry (tj. efektivní syny efektivních otců). Naměřená úspěšnost (90,8 % pro substantiva a 95,0 % pro adjektiva) překračovala baseline o necelý procentní bod, resp. o necelé dvě desetiny p. b. (Výsledky popisovaných experimentů přináší tabulka 4.5.)

Zkusili jsme pro okolní uzly (otec, syn, bratr) rozšířit nabídku rysů na kompletní tagset (česky sada tagů), tj. všech 15 složek z vektoru morfologické analýzy. Dosud jsme programu C4.5 nabízeli jen několik vybraných. Pro substantiva došlo k poklesu úspěšnosti, pro adjektiva k mírnému nárůstu. Kompletní tagset jsme proto z dalších testů opět vyřadili (a vrátili se k němu nakonec, abychom konstatovali, že adjektivům dokáže ještě stále o trochu pomoci).

V dalším experimentu jsme (mimo jiné) nalezli řídící sloveso, tedy první uzel v řadě efektivních otců, který je označen jako sloveso, a do rysů jsme přidali informace o tomto uzlu (tedy některé morfologické kategorie a analytickou funkci). Výsledky pro substantiva to zhoršilo, adjektiva si polepšila.

Následně jsme přidali ještě rysy pro řídící predikát (neboť předchozím experimentem jsme si ověřili, že nalezené řídící sloveso ani zdaleka nebývá predikátem věty) a pro adjektiva řídící substantivum. To překvapivě adjektivům uškodilo. Nicméně pro substantiva jsme získali zatím nejlepší výsledek.

V posledním pokusu bylo tedy použito 48 rysů pro substantiva a 54 pro adjektiva. Rozhodli jsme se otestovat přínos každého jednotlivého rysu zvlášť. Spustili jsme sadu 102 testů, z nichž v každém chyběl právě jediný rys. Cílem bylo porovnat výsledky, zjistit, které rysy jsou nejpřínosnější (bez nich úspěšnost rapidně poklesne), které zbytečné nebo alespoň

[12] Tedy pro uzel nadřízený uzlu desambiguovanému ve **stromě na a-rovině** pro danou větu. Efektivní otec (pokud se liší od přímého otce) lépe odpovídá lingvistické interpretaci stromu. Typickým případem, kdy se liší, je koordinace. Přímý otec člena koordinace je koordinační spojka, efektivní otec je otec spojky („Petr a Pavel jedí.“ – efektivní otec Petra je jíst, jak by tomu bylo (pro oba typy závislosti) i ve větě „Petr jí a Pavel jí.“). Rozdílů je ovšem více. Funkce k nalezení efektivního otce a syna je součástí btredu.

	Substantiva	↕	Adjektiva	↕
Baseline	89,866 %	–	94,741 %	–
Základní sada	90,816 %	↑	94,984 %	↑
+ kompletní tagset	90,663 %	↓	95,065 %	↑
+ řídící sloveso	90,579 %	↓	95,388 %	↑
+ řídící sloveso, subst., predikát	90,928 %	↑	95,146 %	↓
1. vypuštění	91,124 %	↑	95,631 %	↑
2. vypuštění	91,320 %	↑	95,712 %	↑
3. vypuštění	[(a)] **91,348 %**	↑	94,741 %	↓
2. vypuštění + tagset synů	—	–	[(b)] **95,793 %**	↑
Se započítáním neviděných lemmat				
Baseline	87,362 %		92,060 %	
Nejlepší výsledky [(a),(b)]	88,803 %		93,082 %	

Tabulka 4.5: Postupné zlepšování úspěšnosti

zastupitelné (výsledky jsou stejné s nimi i bez nich) a možná i které škodí (úspěšnost se po jejich vypuštění zvedne).

Jedná se samozřejmě pouze o metodu heuristickou, rysy v rozhodovacím stromě pracují společně. Představme si, že vyřadím rys A a jeho roli zastoupí B a C. Úspěšnost vzroste. Pak vypustím B a „jeho“ případy se nějak rozdělí mezi A a C. Úspěšnost opět vzroste. Když pak ale vypustím A i B, mohou být výsledky snadno mnohem horší než dříve. Přestože touto metodou nejsme schopni (lidskou silou) vyčerpat všechny možnosti a výsledek je nejistý, dopadl pokus uspokojivě, jak uvidíme dále.

Získali jsme tedy 102 výsledků, které vypovídaly o chování systému ochuzeného o jeden rys. Soustředili jsme se v nich zejména na dvě kritéria: změnu úspěšnosti a počet lemmat, pro která se rozhodovací stromy nezhroutily do jediného uzlu. V tomto duchu jsme sestavili tabulku 4.6.

Teď bylo potřeba určit, které rysy skutečně zahodit. Intuitivně se nám zdály pro zahození nejvhodnější všechny ty rysy, bez jejichž přítomnosti se úspěšnost zvýšila a počet stromů se snížil. Takový případ dává tušit, že rys byl používán, „košatil“ rozhodovací strom – ale výsledky zhoršoval. Rozdělili jsme tedy rysy na čtyři skupiny, zmiňované „nejdůvěryhodnější pro vypuštění“ padly do první z nich:

1. zlepší se skóre, zmenší se strom
   substantiva: ef_nof (lepší o 1 chybu), ef_LR (2), ef_voic (2), ef_degr (2), gv_LR (4), gp_LR (2), gp_voic (1)
   adjektiva: gs_spos (2), m_degr (2), m_gender (3), m_case (2)

2. zlepší se skóre, strom cokoli
   substantiva: ef_case (2), s_case (5), lb_pers (1), lb_afun (1), rb_pos (6), rb_afun (3), gv_voic (1)
   adjektiva: ef_LR (1), rb_degr (1), gp_LR (1), gs_case (1)

Substantiva		
Skóre se	Rozhod. stromy	Počet rysů
zlepší	ubydou	7
zlepší	zůstanou	5
zlepší	přibydou	2
nezmění	*	12
zhorší	*	22
Celkem		**48**

Adjektiva		
Skóre se	Rozhod. stromy	Počet rysů
zlepší	ubydou	4
zlepší	zůstanou	4
zlepší	přibydou	0
nezmění	*	41
zhorší	*	5
Celkem		**54**

Tabulka 4.6: Jak pomůže vypuštění jednoho rysu

3. skóre se nezmění
   substantiva: ef_pos, lb_pos, [lr]b_voic, [lr]b_degr, gv_pers, gp_pos, gp_pers, empos, m_subpos, m_gender
   adjektiva: afun, ef_nof, ef_pos, ef_case, ef_pers, ef_voic, ef_degr, ef_afun, s_nof, s_LR, s_pos, s_case, s_pers, s_voic, s_degr, s_afun, celý zbytek ?b_ (b_nof, [lr]b_pos, [lr]b_case, [lr]b_pers, [lr]b_voic, lb_degr, [lr]b_afun), gv_LR, celý zbytek gp_ (gp_pos, gp_pers, gp_voic, gp_tens), celý zbytek gs_ (gs_LR, gs_numb, gs_afun), msense, empos, m_subpos, m_number, m_neg

4. skóre se zhorší

Čtvrtá skupina nás nebude při vypouštění zajímat, protože tyto rysy se zdají být pro trénování a testování důležité a nebudeme se jich tedy snažit zbavit. Zbývají tři skupiny a pro ně jsme pustili tři testy: vypuštění 1, vypuštění 2 a vypuštění 3. Jak je patrné z tabulky úspěšností 4.5, substantiva se s každým dalším testem zlepšovala, kdežto adjektiva dosáhla zenitu ve druhém testu. To není až tak překvapivé, uvědomíme-li si, že po 3. vypouštění (pro adjektiva velice masivním) už jim zbylo jen pět rysů. Za zmínku asi stojí, že ačkoli byl v tomto testu dosažen výsledek shodný s baseline, byla to spíše náhoda: rozhodovací stromy se i na pěti rysech vytvořily a dokonce rozhodovaly jinak než baseline. Jen výsledek vyšel stejně.

Poslední fáze testů vzala nejlepší dosavadní množinu rysů (čili vypuštění 3 pro substantiva a vypuštění 2 pro adjektiva) a zkusila ještě jednou přidat kompletní tagset: pro otce, syny, levé bratry, pravé bratry, řídící sloveso, řídící predikát a řídící substantivum (tedy 7 testů). Pro substantiva dal jeden test stejný výsledek (s přidáním řídícího predikátu), všechny ostatní úspěšnost zhoršily. Zato u adjektiv bylo dosaženo nového maxima: s přidáním tagsetu synů, stejně jako s přidáním tagsetů řídícího predikátu je to 95,793 %. Přidání obou množin naráz už dále nepomohlo.

Je ale vidět, že dat je na tento úkol ještě spíše nedostatečně. Třeba i proto, že opravy mezi jednotlivými fázemi testů, obnášející několik desetin procenta, se týkaly jen desítek případů. To zavdává podklad obavě, zda jsme těmito testy neprovedli nepřímé natrénování na dtest datech – a zda výsledky na evaluačním vzorku nebudou znatelně horší.

4.4 Evaluace a diskuse výsledku

4.4.1 Testy na evaluačních datech

Na závěr ověřme kvalitu rozhodovacích stromů (a možná také naši obavu z předchozího odstavce) testem na evaluačních testovacích datech, na adresáři `etest/`. Proběhlo namapování PDT 1.0 (částí, které neměly dosud svůj protějšek) na etest, vygenerovali jsme nové symbolické linky a v trénovacím a testovacím skriptu bylo potřeba zaměnit v definici proměnné `TEST=dtest` za `TEST=etest`. Výsledky zachycuje tabulka 4.7.

`etest/`	Substantiva	Adjektiva
Baseline	90,024 %	93,238 %
Finální sada rysů	91,391 %	93,905 %

Tabulka 4.7: Úspěšnost na evaluačních testovacích datech

Vidíme v ní, že baseline pro substantiva se nepatrně zvýšila, zatímco baseline pro adjektiva je na evaluačních datech významně nižší. Pro oba slovní druhy ji pak naše metoda překročila. U substantiv o 1,4 procentního bodu (výsledek srovnatelný s dtestem), u adjektiv o 0,7 p. b. (horší výsledek než pro dtest, a to i s přihlédnutím k nižší baseline a tedy obtížnější úloze).

4.4.2 Diskuse

Dospěli jsme k závěru, že dosáhnout výsledků znatelně lepších, než je baseline, bude velice obtížné ze dvou důvodů. První je přirozený: baseline je nastavena velice vysoko, což je dáno nerovnoměrným zastoupením významů v datech. Je známou věcí, že zlepšení nad 90 %, nebo dokonce 95 % už jsou náročná a řídká.

A druhý důvod vychází ze stejné podstaty: výskyty významů jsou rozloženy nerovnoměrně. A to natolik, že se vlastně ani nedá předpokládat, že při nešťastné souhře s nepříliš rozsáhlými daty bude C4.5 „ochotna“ vytvářet složité rozhodovací stromy. Vezmeme-li totiž v úvahu, že v průměru je jedno lemma viděno méně než 13× (viz tabulka 4.2) a 90 % (resp. 95 % u adjektiv) významů pokrývá jeden jediný (to vyplývá z baseline), vychází nám, že při budování průměrného rozhodovacího stromu připadá jen 1,3 až 0,65 výskytů na důvěryhodné vytvoření všech vedlejších větví; zatímco všechny zbylé výskyty přispívají na účet nejčastějšímu významu. Není potom divu, že při prořezávání větví, pro které nemluví mnoho výskytů, dochází velice často k ořezání stromu až na jediný list.

Při pohledu touto optikou se nakonec zdají naše výsledky 91,391 % (proti 90,024 %) pro substantiva a 93,905 % (proti 93,238 %) pro adjektiva jako velice úspěšné. Jedná se totiž

o opravení 13,7 % chyb (10,0 %), kde se zmýlila baseline. U vývojových testovacích dat to bylo pro adjektiva dokonce rovných 20 %.

4.5 Krátká zmínka o rysech t-roviny

Na závěr této kapitoly bychom rádi zmínili původní pokusy, které jsme prováděli. Prvotní snaha byla využít v maximální možné míře t-rovinu, proto i trénování jsme se snažili provádět na průniku dat, která mají anotaci na t-rovině, a dat, která mají anotaci významu dle ČWN. Jak už jsme řekli, tento průnik je velice malý (obr. 3.1 a kapitola 3.1.2). Tuto oblast jsme se však rozhodli opustit, neboť – jak už bylo také řečeno – z dat, která jsme získali anotací WordNetem (507 470 tokenů), má pouze 26 % t-rovinu: a to je skutečně nedostatečné množství k natrénování rozhodovacích stromů.

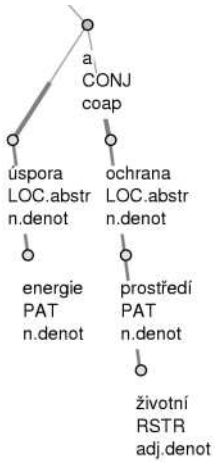
Substantiva mají v trénovacích datech 14 502 výskytů, adjektiva 2 173. To při počtu 2 258 (resp. 338) lemmat dává v průměru 6,4 výskytů na jedno lemma (pro oba). V těch šesti výskytech jsou všechny významy. To jsou skutečně velice chabé základy pro nějaké kvalitní trénování.

Po dokončení testů na a-rovině jsme se ještě krátce podívali na t-rovinu, s použitím v tu dobu již vyvinutých trénovacích nástrojů.

Protože v PDT vedou odkazy mezi rovinami vždy od vyšší k nižší, není možné přejít například od ewn-sense na m-rovině k funktoru z t-roviny. Do struktury PDT tudíž tentokrát musíme „vcházet“ skrze t-rovinu. Tam nastává hlavní problém. Zatímco mapování mezi a-rovinou a m-rovinou (to jsme využívali dosud – pro zjištění morfologických informací k uzlům, které jsme nalezli přechodem po hranách stromu na a-rovině) je $1 : 1$, přechod od uzlů t-roviny k uzlům na a-rovině je $m : n$.

Náš průchod PDT vypadá takto: procházíme jeden uzel PDT za druhým, od každého se podíváme na odpovídající uzel a-roviny a z něj opět na m-rovinu. Tam zkontrolujeme, zda je přítomna anotace ewn-sense. Pokud je, víme, že se musíme tomuto uzlu věnovat podrobně: dodat mu vhodné informace z m-roviny a a-roviny už dokážeme, záměr je tedy přidat ještě nějaké z t-roviny.

A teď zpět k problému $m : n$. Když k jednomu uzlu na a-rovině vedou odkazy od dvou na t-rovině, může se stát (pokud by onen a-uzel representoval lemma, kterému byl přiřazen ewn-sense), že budeme totéž lemma zpracovávat dvakrát. Konkrétní příklad vidíme na obrázku 4.3: lemma „oblast“ není na t-rovině vůbec realisováno jako samostatný uzel, ale je vyjádřeno subfunktorem abstr u lemmatu „úspora“ („abstr je subfunktor pro abstraktní význam *v oblasti*. Náleží funktoru LOC.“ – [10] str. 592) a stejně tak i u lemmatu „ochrana“. Protože zrovna oblast je anotována pomocí synsetu, zpracovávala by se dvakrát. (To by znamenalo také zvýšení celkového počtu slov atd.)



Obrázek 4.3: Sloučení uzlů v jeden na t-rovině (část věty „. . . v oblasti úspor energií a ochrany životního prostředí“)

Opačný případ nevadí. Když od jednoho uzlu na t-rovině vedou odkazy ke dvěma na a-rovině budou mít jednoduše všechny hodnoty tektogramatických složek vektoru vyplněny stejně.

Tento problém jsme se rozhodli vyřešit způsobem jednoduchým (možná na úkor přesnosti). Při přechodu z t-roviny využíváme pouze odkazu na ten a-uzel, který je nejdůležitějším předchůdcem t-uzlu (to je v PDT jasně odděleno, jeden uzel je vždy hlavní (označen odkazem `rf.lex`) a případné ostatní jsou označeny odlišně (`rf.aux`)). Tím se může stát, že ne všechna lemmata využijeme, na nich natrénujeme a otestujeme (konkrétně v příkladu z našeho obrázku se v tomto stavu naopak nebude „oblast“ zpracovávat ani jednou).

První testy jsme provedli jednoduše s nejlepší sadou rysů z pokusů s daty na a-rovině (říkejme jim a-rysy), bez přidání jakéhokoli rysu pro t-rovinu. Jak je vidět z tabulky 4.8, pro tento pokus je dat skutečně málo: pro substantiva se dokonce sadou „nejlepších“ rysů z a-roviny ani nepodařilo překonat baseline.

Experiment	Substantiva	Adjektiva
Baseline	91,657 %	93,380 %
Nejlepší a-rysy	91,373 %	93,728 %
Nejlepší t-rysy	91,544 %	93,380 %

Tabulka 4.8: Úspěšnost experimentu na t-rovině v porovnání s baseline a se sadou nejlepších a-rysů

Kapitola 5

Automatické přiřazování rámců

S valenčními rámci jsme se seznámili v kapitole 2.2.1, kde jsme také zmínili, že pokud máme dva syntakticky odlišné rámce jednoho slova, většinou se od sebe odlišují i významy těch dvou rámců. Úloha na přiřazení správného syntaktického rámce je tedy zároveň úlohou WSD. Valenčními rámci je anotovaná t-rovina PDT 2.0, což budou naše trénovací a testovací data.

5.1 Známé výsledky

S problémem automatického přiřazování rámců už byly konány některé pokusy (například již několikrát citované [2, 16]). V tomto případě šlo dokonce také o přiřazování rámců podle valenčního slovníku, velice podobného našemu. Tyto pokusy se však týkaly výhradně sloves. Dosáhli úspěšnosti 79,16 % ([2]), posléze až 80,55 % ([16]) (nevážená úspěšnost: na takovém rozložení dat, jaká používali) oproti baseline 67,94 %. My se proto slovesům věnovat nebudeme.

5.2 Specifikace úlohy

Problém jsme pro jednoduchost postavili stejně jako při přiřazování synsetů. Tedy na vstup si připravíme jen ta lemmata, která jsou desambiguovaná v ruční anotaci, a snažíme se jim přiřadit tentýž rámec. Zatímco ve zmíněné úloze jsme to zdůvodnili tím, že není možné automaticky poznat, které výskyty jsou anotovány (neboť to záleželo mj. na shodě anotátorů), proto jsme o tom nerozhodovali, tak zde tvrdíme, že naopak všechna slova, která jsou ve slovníku PDT-VALLEX, by měla být v datech anotována: proto opět není o čem rozhodovat a budeme postupovat, jak jsme uvedli: tj. přiřazovat pouze anotovaným lemmatům.

Dalo by se namítat, že to není tak docela pravda. Že neexistuje záruka, že slovo existující ve slovníku bude v datech vždy anotováno. Ačkoli všechna data byla ručně kontrolována, substantiva a adjektiva zatím neprošly kompletními automatickými testy, které by právě toto opominutí testovaly. V tom případě ale neexistuje kritérium, podle kterého rozhodnout, zda uzel ve stromě byl opominut, nebo zda je správně anotován: a je tedy zbytečné chtít po strojovém učení, aby tuto vlastnost trénovala. Dostali jsme se tedy opět ke stavu, kdy je rozumné všechny uzly, které jsou ručně anotovány a mají být strojově desambiguovány, předávat programu připravené a nechtít po něm žádný výběr.

Pro výběr rysů jsme se rozhodli využít dvou zdrojů: jednak naší nejlepší sady rysů z ČWN, jednak se necháme inspirovat syntaktickými rysy popsanými v [16]. Tam se jednalo o slovesné valenční rámce a syntaxi povrchovou, na a-rovině – my se pokusíme rysy převést do tvaru pro substantivní valenční rámce a syntaxi hloubkovou, na t-rovině.

5.3 Statistiky a baseline:

Slovesa jsme z této úlohy vyloučili už na počátku, teď ukážeme, proč se nebudeme zabývat ani adjektivy. V celých trénovacích datech se vyskytuje 59 adjektiv ve 294 instancích (tj. 5 výskytů na lemma). Průnik s testovacími daty pak tvoří pouze 5 adjektiv. A baseline vychází 100 %. Tady není co zlepšovat. O čem to vypovídá? Nejvíce o tom, že PDT-VALLEX není dobrý slovník adjektiv.

Zbývají nám tedy pouze substantiva. V trénovacích datech je 13 306 výskytů 1 856 lemmat (tedy 7,2 výskytu na lemma). Jim je přiřazeno 2 214 PDT-VALLEXových rámců, tedy přibližně 1,2 významu na lemma. Z nich však pouze 284 je víceznačných; na ně připadá 639 významů, tedy 2,25 významu na jedno.

Za baseline vezměme opět přiřazení nejčastějšího rámce. Vychází pro ni úspěšnost 84,87 %.

5.4 Převzaté rysy

Zde předkládáme seznam rysů tak, jak bude publikován v [16], a jak jsme ho pozměnili pro náš účel. Všechny původní i naše mají dvě možné hodnoty, TRUE a FALSE. Pro přehlednost se budeme v tomto soupisu vyjadřovat zkratkovitě: vždy jen kolik rysů je ve skupině a co popisují.

- 2: Je přítomno zvratné zájmeno se, si?
  ⇒ Bez náhrady.
- 1: Závisí sloveso na jiném?
  ⇒ 1: závisí podstatné jméno na jiném?

- 1: Nalézá se v podstromu slovesa závislé sloveso?
  ⇒ 6: Nalézá se ve větě (přímo) závislé substantivum/adjektivum/sloveso?
- 6: Přítomnost vybraných 6 spojek závislých na slovesu.
  ⇒ 7: Nalézá se ve větě přímo závislá spojka „jak“, „až“, „ať“, „jestli(že)“, „zda(li)“, „že“, nebo „aby“?
- 7: Předložková skupina závislá na slovesu (v každém z pádů).
  ⇒ Totéž, závislá na substantivu.
- 7: Substantivum/substantivní zájmeno přímo závislé (v 7 pádech).
  ⇒ Totéž.
- 7: Adjektivum/adjektivní zájmeno přímo závislé (v 7 pádech).
  ⇒ Totéž.
- 3: Adjektivum v daném stupni, přímo závislé.
  ⇒ Totéž.
- 69: Všechny kombinace předložek a pádů, přímo závislé.
  ⇒ 7: Cokoli přímo závislé (v 7 pádech).

5.5 Vlastní experiment, výsledky

Po technické stránce je úloha poměrně snadná, zvláště s využitím některých modulů z předchozí kapitoly. Valenční rámec se v PDT uvádí u slov přímo na t-rovině, není tedy třeba ani slučovat více dat, ani prohledávat PDT skrz roviny. Samotný rámec se v PDT značí takto

```
<val_frame.rf>v#v-w5386f2</val_frame.rf>
```

Počáteční v# značí odkaz do PDT-VALLEXu, 5386 je číslo lemmatu v něm a f2 je označení valenčního rámce. Protože naše úloha probíhá odděleně pro každé lemma, můžeme si dovolit celý prefix odříznout a pracujeme jen s číslem rámce.

Program na základě seznamu požadovaných rysů vygeneruje z PDT vektor rysů pro každý výskyt substantiva s rámcem, tyto vektory jsou roztříděny podle lemmat a na každém lemmatu zvlášť je natrénováno a otestováno přiřazování pomocí C4.5.

Experiment	Úspěšnost
Baseline	84,87 %
T-BEST	85,81 %
T-BEST + JST	86,45 %
T-BEST + JST − spojky, cokoli v pádě	86,50 %

Tabulka 5.1: Úspěšnost přiřazování valenčních rámců s různými sadami rysů

Se sadou nejúspěšnějších rysů z kapitoly 4.5 (nazýváme T-BEST) a se sadou přejatých z kapitoly 5.4 (nazýváme jst) jsme dosáhli úspěšnosti 86,45 % (připomeňme: baseline je 84,87 %, viz tabulka 5.1). Sledovali jsme, jakého uplatnění dojdou nově přidané rysy. S výjimkou *všech* rysů pro podřadící spojky (a čtyř méně častých pádů) byly alespoň pro nějaké lemma využity všechny. O to zajímavější je, že pokud celou sadu JST nepožijeme, výsledky jsou téměř stejné (pokles o polovinu procentního bodu). Provedli jsme ještě několik experimentů a nepatrně lepších výsledků jsme dosáhli, pokud jsme vyřadili rysy pro spojky (nevyužité) a pro závislý uzel (jakýkoli) v daném pádě.

Evaluační test	Úspěšnost
etest – baseline	84,92 %
etest – rysy	86,91 %

Tabulka 5.2: Evaluační test přiřazování valenčních rámců

Testy na evaluačních datech úspěšnost potvrdily (tabulka 5.2).

Kapitola 6

Závěry...

Testovali jsme úspěšnost počítačové desambiguace na základě synsetů z Českého WordNetu. Nejprve jsme museli slít data těmito synsety anotovaná, s týmiž daty PDT, protože byla v jiném formátu. Věnovali jsme tomu značnou část úsilí a podařilo se nám to. Testovali jsme přiřazování synsetů substantivům a adjektivům. I přesto, že baseline (stanovená jako přiřazení nejčastějšího rámce) dosahovala téměř 90 % (resp. 95 %), podařilo se nám ji překonat o více než jedno procento. Tím jsme opravili 13,7 % (resp. 10,0 %) chyb oproti baseline. Pracovali jsme převážně na a-rovině PDT, některé testy jsme však prováděli i na t-rovině: ukázali jsme, že zde je na směrodatné výsledky k disposici příliš málo dat.

Poté jsme zkusili tytéž postupy aplikovat na přiřazování valenčních rámců a spolu s nimi se přesunout o rovinu výše, na t-rovinu. Vypomohli jsme si ještě „tektogramatickou" variantou úspěšných rysů z jiného pokusu, upravenou pro substantiva. Obě naše úlohy byly z podstaty různé, proto výsledky nelze porovnávat. Můžeme ale říci, že t-rysy neměly pro konečnou úspěšnost zdaleka takový přínos, v jaký jsme doufali.

6.1 Srovnání slovníků

Pokud se máme vyjádřit k oběma slovníkům, které jsme měli k disposici, je třeba na prvním místě zdůraznit, že je těžké představit si jiné dva, které by se lišily více. Zaměříme se na jejich největší rozdíly.

WordNet pokrývá mnohem větší množinu slov. Jednak je starší a jednak teoreticky nebrání nic představě, že budou ve WordNetu všechna plnovýznamová slova. Jeho základními prvky jsou relace synonymie, hyperonymie a hyponymie. Ale jak jsme uvedli již v kapitole 3.1.1, je málo precisní. Mnoho slov tak sice má ve WN několik významů, ale bez jistoty, jak se přesně liší, jsou bezcenné. Není to však chybou

autorů. Nakolik můžeme možnosti WN posoudit (z hlediska jiného projektu [1], kde jsme se jím zabývali), nebude snadné jej vylepšit či lépe uspořádat. Otázkou dokonce je, zda to pro přirozený jazyk bude vůbec kdy možné.

PDT-VALLEX je rozhodně mnohem propracovanější (do hloubky v jednotlivých lemmatech, ne do šířky), je vidět, že prošel několika kontrolami. Nepokrývá však zdaleka tolik lemmat (WN: 23 050 synsetů, v nich 34 131 literálů; PDT-VALLEX: 14 951 rámců v 10 020 lemmatech; navíc je potřeba si uvědomit, že většina lemmat PDT-VALLEXu jsou slovesa). Do budoucna se také nedá očekávat, že by se PDT-VALLEX rozšířil na všechna slova, půjde to pro slovesa, deverbativní substantiva, adjektiva a adverbia a pro další omezené skupiny, které mají valenční vlastnosti. Rozhodně jako slovník adjektiv PDT-VALLEX nedoporučujeme.

6.2 Budoucnost

Na závěr bychom chtěli poznamenat několik myšlenek či námětů, které by mohlo být zajímavé realisovat, ale v rámci této práce se to již nepodařilo.

- Rádi bychom podrobně prozkoumali, které konkrétní rysy se uplatnily u kterého lemmatu, jak byly úspěšné a mezi kterými významy rozhodovaly. Pro několik nejčastějších, pro ty, které nejvíce „pomáhaly“, ...
- Výsledný program by bylo možná (po úpravě) použít jako jeden z modulů pro jiné nástroje NLP. Očekáváme, že desambiguace významu substantiv ve větě by mohla pomoci třeba desambiguaci sloves.
- Všechny skripty byly navrženy tak, aby dokázaly zpracovat i jiné slovní druhy. Nakonec jsme se ale z uvedených důvodů zaměřili na substantiva a adjektiva, takže pro verba a adverbia jsme nenavrhovali žádné další rysy. Rádi bychom jim věnovali alespoň krátkou pozornost.
- Výsledné úspěšnosti by byly průkaznější, kdyby bylo k disposici více dat. Druhou možností je měřit výsledky pomocí tzv. „crossvalidace“. Tam se výsledná úspěšnost určí jako průměr z několika pokusů, v nichž vždy jiná část hraje roli testovacích dat a na zbytku se trénuje.
- Mnoho hodnot rysů máme tvořených kumulativním způsobem (např. `s_case=s_4_4_6` vyjadřuje tři synovské uzly: dvakrát v akuzativu a jednou v lokativu). Stálo by za zvážení, zda nebude úspěšnější druhý způsob, který zmnoží počet rysů na úkor jejich

hodnot (v tomto případě například `s_1=F; s_2=F; s_3=F; s_4=T; s_5=F; s_6=T; s_7=F`).

Literatura

[1] Eduard Bejček, Petra Möllerová, and Pavel Straňák. The Lexico-Semantic Annotation of the Prague Dependency Treebank: Some results, problems and solutions. Rukopis, zaslán na Ninth International Conference on TEXT, SPEECH and DIALOGUE, 2006.

[2] Ondřej Bojar, Jiří Semecký, and Václava Benešová. Testing VALLEX Consistency and Experimenting with Word-Frame Disambiguation. *Prague Bulletin of Mathematical Linguistics*, (83):5–17, 2005.

[3] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[4] William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *Proceedings of the 4th DARPA Speech and Natrual Language Workshop*, pages 233–237, 1992.

[5] Jan Hajič, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová, Veronika Kolářová, and Petr Pajas. PDT-VALLEX: Creating a Large-coverage Valency Lexicon for Treebank Annotation. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of The Second Workshop on Treebanks and Linguistic Theories*, volume 9 of *Mathematical Modeling in Physics, Engineering and Cognitive Sciences*, pages 57—68, Vaxjo, Sweden, 2003. Vaxjo University Press.

[6] Jan Hajič. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, Prague, 2004.

[7] Jan Hajič, Eva Hajičová, Jaroslava Hlaváčová, Václav Klimeš, Jiří Mírovský, Petr Pajas, Jan Štěpánek, Barbora Vidová Hladká, and Zdeněk Žabokrtský. Průvodce PDT 2.0. http://ufal.mff.cuni.cz/pdt2.0/doc/pdt-guide/cz/html/index.html, 2005.

[8] Jiří Hana, Daniel Zeman, Jan Hajič, Hana Hanová, Barbora Hladká, and Emil Jeřábek. Manual for Morphological Annotation, Revision for the Prague Dependency Treebank 2.0. Technical Report TR-2005-27, ÚFAL MFF UK, Prague, Czech Rep., 2005.

[9] Aleš Horák and Pavel Smrž. Visdic – wordnet browsing and editing tool. In *Proceedings of the Second International WordNet Conference – GWC*, pages 136–141, Brno, Czech Republic: Marasyk University, 2004.

[10] Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský. Anotace na tektogramatické rovině Pražského závislostního korpusu. Anotátorská příručka. Technical Report TR-2005-28, ÚFAL MFF UK, Prague, Prague, 2005.

[11] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill Higher Education, 1997.

[12] Petr Pajas. Tree Editor TrEd. http://ufal.mff.cuni.cz/pdt2.0/doc/tools/tred/, 2000–2005. Software.

[13] Petr Pajas and Jan Štěpánek. A Generic XML-Based Format for Structured Linguistic Annotation and Its Application to Prague DependencyTreebank 2.0. Technical Report TR-2005-29, ÚFAL MFF UK, Prague, Czech Rep., 2005.

[14] Karel Pala and Pavel Smrž. Building czech wordnet. *Romanian Journal of Information Science and Technology*, 7(1–2):79–88, 2004.

[15] Martin Pavlík. Sémantická disambiguace termů v DIS. Master's thesis, Prague, Czech Rep., 2002.

[16] Jiři Semecký. On automatic assignment of verb valency frames in czech. In *Proceedings of LREC*, 2006. V tisku.

[17] Petr Sgall, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands, 1986.

Seznam obrázků

2.1 Zachycení věty „Vedle dosavadních programů jsou to nové programy Special a Záruka.“ na a-rovině a t-rovině PDT 8
2.2 Propojení rovin PDT (převzato z [7]) 9
2.3 Sloveso „krýt“, „krýt se“ a substantivum „krytí“ v PDT-VALLEXu 10
2.4 Slovo „dílo“ v Českém WordNetu (zobrazeno ve VisDicu [9]) 13

3.1 Prolínání rovin PDT a anotace významu 27

4.1 Počet výskytů jednotlivých významů 39
4.2 Zastoupení lemmat s daným počtem významů (osa y logaritmická) 40
4.3 Sloučení uzlů v jeden na t-rovině (část věty „. . . v oblasti úspor energií a ochrany životního prostředí“) 53

Seznam tabulek

3.1 Rozložení anotovaných dat v a-rovině a t-rovině, procento anotace 27

4.1 Počty slov v anotovaných datech 37
4.2 Frekvenční zastoupení v trénovacích datech 40
4.3 Baseline pro přiřazování synsetů, pokud „nespatřená“ lemmata vyřazujeme, resp. započítáváme jako chyby 41
4.4 Frekvenční zastoupení v testovacích a v celých datech 43
4.5 Postupné zlepšování úspěšnosti 49
4.6 Jak pomůže vypuštění jednoho rysu 50
4.7 Úspěšnost na evaluačních testovacích datech 51
4.8 Úspěšnost experimentu na t-rovině v porovnání s baseline a se sadou nejlepších a-rysů 53

5.1 Úspěšnost přiřazování valenčních rámců s různými sadami rysů 57
5.2 Evaluační test přiřazování valenčních rámců 57

Dodatek A

Programátorská dokumentace

Veškeré manipulace s daty – slévání, mapování, trénování, testování, měření apod. – které jsme prováděli, jsou reprodukovatelné s využitím již existujících nástrojů a skriptů[1] v jazycích Perl a Bash, která jsou na přiloženém CD.

A.1 Struktura CD

Proto je vhodné zde také popsat strukturu CD. Celé CD je zamýšleno tak, že ho uživatel zkopíruje na pevný disk tak, jak je, a jednotlivé skripty budou fungovat. Trochu potíž je v tom, že není možné na CD umístit data, na kterých experimenty probíhaly, v množství větším než ukázkovém. Přesto jsou ukázková data umístěna tak, jako by byla skutečná, tedy rozmístěna v mnoha (povětšinou prázdných) adresářích. V případě nahrazení těchto ukázkových dat skutečnými, bude vše fungovat lépe, proto je to vřele doporučeno.

Podobně jsme nemohli (vzhledem k licenčním podmínkám) umístit na CD ani volně stažitelný program C4.5 a také tam není přiložen btred. Opět, jsou tam adresáře tak, jako by na CD zmiňované programy byly.

`data` sem patří všechna data; jsou zde však pouze krátké ukázky, není možno zveřejňovat plnou versi;

`CWN_sample` data PDT 1.0 anotovaná synsety z ČWN;

`amw` tato data se v PDT dostala jen na a-rovinu;

`tamw` tato data se v PDT dostala až na t-rovinu;

[1] Upozornění: Všechny skripty byly psány pro naši potřebu. Při dodržení všech vstupních podmínek popsaných v této práci, a to zejména v této a následujících přílohách, pracují správně. Jiná data a jiné použití nebylo předpokládáno a chování skriptů není v takových případech testováno.

`PDT_sample` data PDT 2.0;

- `amw` data anotovaná na w-, m- a a-rovině;
- `tamw` data anotovaná též na t-rovině;
- `train-1`
- `train-8` dva z osmi trénovacích adresářů;
- `dtest` vývojová testovací data;

`nastroje`

- `C4.5` adresář vhodný pro nainstalování programu pro strojové učení, který nemůže být distribuován (kap. 2.5);
- `tred` adresář vhodný pro nainstalování tredu; je k disposici na webu (kap. 2.4);

`output` prázdný adresář, který je potom využíván všemi skripty k výpočtům;

`skripty` obsahuje naše skripty v Perlu a Bashi;

`analyt-syns` skripty pro trénování synsetů ČWN na a-rovině

- `baseline_lemma.pl` jednoduchý skript, počítající pro jedno konkrétní lemma jeho baseline založenou na přiřazení nejčastějšího významu; používá ho `results_process.pl`;
- `build_names.pl` vytváří soubor `universal.names`, kap. C.3 níže;
- `classify.pl` třídí lemmata zvlášť do souborů, kap. C.4;
- `get_features.btred` makro pro btred, slouží k získání vektorů rysů z PDT, kap. C.2;
- `results_process.pl` spočítá celkovou úspěšnost (i baseline) z dílčích výsledků pro jednotlivá lemmata, kap. C.5;
- `run` skript, který předchozí postupně pouští se správnými parametry, kap. C.6;
- `rysy_tred` konfigurační soubor, určující sadu rysů k pokusu, kap. C.1;
- `rysy_tred_dalsi` adresář s připravenými sadami rysů pro pokusy;

`namapovani` skripty pro úlohu namapování anotace pomocí ČWN z PDT 1.0 na PDT 2.0;

`get_senses_from_ewn-pdt.pl` vytáhne jen významy z anotačních souborů a určí jméno cílového, kap. B.1;

`map_senses_to_pdt.pl` provede namapování získaných významů na m-rovinu PDT, kap. B.2;

`link_fake_data` vytvoří „falešnou“ versi korpusu tím, že podvrhne m-soubory s přidanou anotací významu místo původních, kap. B.3;

`run` skript na pohodlné spuštění předchozích;

`tekto-ramce` stejné (jen lehce pozměněné) skripty jako v `analyt-syns`, jen pro valenční rámce na t-rovině;

`tekto-syns` stejné (jen lehce pozměněné) skripty jako v `analyt-syns`, jen pro ČWN na t-rovině;

`vypust_rys.pl` skript pro postupné vypuštění každého jednotlivého rysu, kap. C.7;

`run_fragment` obdoba `run`, sloužící k testování vypuštěných rysů;

`text` tento text diplomové práce.

A.2 Jak udělat...

- Pokud chci namapovat synsety na PDT, spustím `./run` z adresáře `skripty/namapovani`. (Kapitola B.)
- Pokud chci natrénovat synsety na a-rovině, spustím `./run` z adresáře `skripty/analyt-syns`. (Kapitola C, zejména pak C.6.)
- Pokud chci natrénovat synsety na t-rovině, spustím `./run` z adresáře `skripty/tekto-syns`. (Kapitola C.)
- Pokud chci natrénovat valenční rámce na t-rovině, spustím `./run` z adresáře `skripty/tekto-ramce`. (Kapitola C.)
- Chci-li zkusit jinou sadu rysů, budu editovat soubor `skripty/[tekto|analyt]-[syns|ramce]/rysy_tred`. (Kapitoly C.1 a D.)
- Chci-li vypustit rysy a zjistit, jak to dopadne, použiji skripty `/skripty/vypust_rysy.pl` a `/skripty/run_fragment`. (Kapitola C.7.)

Dodatek B

Programátorská dokumentace – část slévání

B.1 `get_senses_from_ewn-pdt.pl`

Umístění na CD: `skripty/namapovani/get_senses_from_ewn-pdt.pl`

Úkolem tohoto skriptu je ze dvou souborů anotací od anotátora A a anotátora B slít soubory výsledné, které budou obsahovat již jen anotaci, na které se oba shodli. Výsledné soubory jsou již dělené v souladu s PDT 2.0, proto jich může být více. Chování skriptu je stručně a přehledně nastíněno v kapitole 3.2.1, což sem nebudeme opisovat, a přistoupíme detailněji k samotnému skriptu. Je psán v perlu a je hojně komentován. Na vstupu očekává minimálně čtyři argumenty:

1) Jméno souboru, do něhož se vypisuje podrobný výstup, ve formě čitelnější pro člověka. Jelikož tento výstup slouží pouze ke kontrolním účelům, nebudeme se o něm již dále zmiňovat.

2) Prefix souborů, do nichž se bude vypisovat výsledek slévání anotací a které budou sloužit jako vstup druhému skriptu. Nazývejme je třeba ryzí anotace. Jména souborů za prefixem budou doplněna odpovídajícím způsobem v souladu s názvy v datech PDT 2.0. Součástí prefixu může být i adresář. Zvolíme-li například prefix `shoda_anotace/ryzi` vznikne v adresáři `./shoda_anotace/` mj. soubor `ryzi_cmpr9413_031` s vydestilovanými (souhlasnými) anotacemi pro soubory `cmpr9413_031.[tamw].gz` z PDT 2.0.

3) Komprimovaný `soubor_A` od prvního anotátora ve formátu popsaném v kapitole 3.1.2.

4) Komprimovaný tentýž `soubor_B` od druhého anotátora (v témže formátu).

*) Dále může následovat libovolný počet dvojic souborů `s1_A s1_B s2_A s2_B` ...

Skript vezme první dva anotované soubory, zkopíruje je do aktuálního adresáře a dekomprimuje. Zkontroluje, že se jmenují stejně. Oba soubory otevře naráz a opakovaně z obou načítá řádek po řádku. Řídí se ovšem pouze prvním ze souborů, v něm vyhledává řídící řetězce. Jejich regulární výrazy vypadají následovně:

`/^<s /` ... Značí začátek nové věty. Každá věta má svůj identifikátor ve tvaru `id="cmpr9410:016-p10s3"`. Řetězec až do dvojtečky značí jméno novin, ročník a výtisk, ze kterého věta pochází. Spolu s druhým – blíže specifikujícím – číslem se stal názvem souborů v PDT 2.0. (Dál následuje číslo odstavce a číslo věty v něm.) Pokud se začátek identifikátoru věty liší od předchozího (což značí, že se plynule přešlo do dalšího výtisku novin), skript otevře nový výstupní soubor. V každém případě ID věty vypíše na výstup.

`/^<d>/` ... Značí token obsahující interpunkci. Pouze se vypíše na výstup.

`/^<f(>|\ \w+)/` ... Značí token obsahující slovo. Kromě nejčastějšího typu (`<f>proti`) může začínat také kupříkladu `<f cap>Kazimír`, `<f num>1`, `<f abbr>tel`, `<f upper>ČSFR`, `<f mixed>McGill` apod. Slovo se nejprve opíše na výstup. Dále se zkontroluje, zda bylo vůbec anotováno (z čehož plyne, že to musí být autosémantické slovo a musí mít alespoň jeden výskyt v Českém WordNetu). Pokud ne, pokračuje se dalším řádkem. Jestliže bylo anotováno, vyhledá se, jaký synset zvolil který z anotátorů. Oba synsety se potom porovnají. Jestliže se anotátoři neshodli, pouze se ke slovu poznamená „`\\ Neshoda`“. Jestliže se shodli, ale jen na chybové kategorii, připíše se „`// Err. categ.`“. Jestliže se shodli, ke slovu se připojí řetězec tvaru „`|| literal--n-3=324534`“. První dvě značky (tedy lomítka) se vypisují jen pro kontrolu, jinak jsou dále ignorovány. Takže ani není potřeba rozlišovat neshodu na chybě od neshody na synsetu apod. – tato anotace rozhodně nebude použita k trénování.

Po dosažení konců obou souborů jsou smazány z aktuálního adresáře. Pokračuje se případnými dalšími soubory, byly-li nějaké zadány. Na úplný závěr se vypíše statistika, kolik bylo nalezeno neanotovaných dat, na kolika se anotátoři shodli, kolikrát zvolili chybovou kategorii a kolikrát se neshodli.

Ukázka části výstupního souboru:

```
===> cmpr9413_031_p13s4 <===
V
dubnu || duben--n-1==9133250
se
k
tomu
má // Err. categ.
vyjádřit || vyjádřit se--v-4==529407
```

```
Rada || rada--n-5==5292131
hospodářské \\ Neshoda
a
sociální || sociální--a-1==2047116
dohody \\ Neshoda
.
```

B.2 map_senses_to_pdt.pl

Umístění na CD: `skripty/namapovani/map_senses_to_pdt.pl`

Skript namapuje výstup z `get_senses_from_ewn-pdt.pl` na m-rovinu PDT 2.0, kam přidá nový element nesoucí informaci o přiřazeném synsetu. Bodový přehled činnosti je v kapitole 3.2.2, zde nabízíme pohled podrobný. Skript očekává minimálně tři argumenty:

1) Vstupní adresář, který obsahuje soubory z PDT 2.0 s m-rovinou. Tento adresář nemusí obsahovat soubory přímo, jsou prohledávány i podadresáře. To je kvůli standardnímu rozmístění do adresářů `train-1/` až `train-8/`, stačí tedy uvést cestu k vyššímu adresáři. Jsou hledány soubory s příponou `.m.gz`.

2) Výstupní adresář, do kterého budou ukládány soubory s příponou `.e`. To jsou soubory m-roviny s přidanou anotací významu pomocí ČWN. Místo jména adresáře je možno použít `"-"`, potom je `souborX.e` umístěn k odpovídajícímu `souborX.m.gz`, ze kterého vznikl, a navíc ještě komprimován.

3) Soubor s ryzí anotací, který vznikl jako výstup skriptu `get_senses_from_ewn.pl` (kapitola 3.2.1), tedy něco jako `ryzi_cmpr9413_031`. V něm jsou všechna slova z daného anotovaného dokumentu spolu se synsety – pokud byl ke slovu přiřazen synset (při shodě anotátorů).

*) Libovolný počet souborů z bodu (3).

Skript bere postupně jednu ryzí anotaci po druhé a začne vždy tím, že se pokusí ve vstupním adresáři nalézt odpovídající soubor na m-rovině. Konkrétně tedy třeba k souboru `shoda_anotace/ryzi_cmpr9413_031` nalezne soubor `train-3/cmpr9413_031.m.gz` a začne je postupně slévat do výsledného `cmpr9413_031.e`. Pokud soubor není nalezen, je to ohlášeno a pokračuje se dalším. Máme tedy dva vstupní soubory, oba otevřené. Mapování se provádí za prvé pomocí ID vět a za druhé pomocí pořadového čísla slova ve větě.

Postupně se prochází ryzí anotace. Když se narazí na řádek označující začátek nové věty (`===> cmpr9413_031_p13s4 <===`), vytvoří se z něho řetězec, který se bude vyhledávat v souboru s m-rovinou (`<. id=\"m-cmpr9413-031-p13s4`). Dále se zvyšuje čítač za každé

slovo, které ve větě je. Když se narazí na slovo, u něhož je poznamenán synset, začne se hledat v souboru s morfologií. Bylo-li to řekněme 7. slovo ve větě, hledá se regulární výraz `/<. id=\"m-cmpr9413-031-p13s4[A-Z]?w7/`.[1] Nalezne se, tam tedy začíná element pro dané slovo. Uvnitř něho se dojde až k řádku s lemmatem a za něj se vloží nový řádek s významem reprezentovaným synsetem. To je celé.

Uvědomujeme si, že přílišné spoléhání na pravidelnost budování ID se nemusí vyplatit, neboť není zaručena. Nicméně v PDT 1.0 se ID slov neobjevovala, a tak nic jiného nezbývá. Dále také selhaly původní pokusy porovnat dané slovo buď jako lemma z m-roviny oproti literálu ze synsetu (lemma nebylo v anotovaných datech snadno dostupné), nebo jako „form“ z m-roviny oproti formě v PDT 1.0. První selhalo například na spojení „a. s.“, ve kterém je „s“ anotováno jako „akciová společnost“, kdežto lemma je „společnost“. Druhé selhalo například na opravených chybách: některé již byly opravené v 1.0 (takže se tam vyskytoval element `<w>`), bohužel však ne všechny, takže třeba ve větě „Během stáže jsem strávil dva týdny v ekonomickém týdeníku Les Affaires – kanadské období Profitu.“ se slovo „období“ dohledat nedá. Problémů tedy bylo hodně, ale praxe naštěstí po mnoha omylech ukázala, že počítání slov a odhadování ID je způsob poměrně spolehlivý.

B.3 `link_fake_data`

Umístění na CD: `skripty/namapovani/link_fake_data`

Tento velice jednoduchý skript má jediný účel: podvrhnout nové soubory vzniklé předchozím skriptem (`soubor.e.gz`, což je soubor m-roviny a navíc značky z ČWN) coby původní m-soubory. Na vstupu dostane dva adresáře: jeden s daty a druhý, který má vytvořit. Pracuje tak, že všem souborům z podadresářů (pro účely `train-1` až `train-8`) vstupního adresáře vyrobí symbolické linky do cílového adresáře na soubory stejného jména. Jen soubory `.e.gz` nalinkuje na `.m.gz`, jako by to byla pravá m-rovina.

[1] Pokud si pozorný čtenář povšiml potenciálních velkých písmen v ID (`[A-Z]?`) a překvapují ho, zde je vysvětlení: když docházelo ke slučování a rozdělování vět (při ručních kontrolách před vydáním PDT 2.0), docházelo k jejich označování právě jako např. p7s13A a p7s13B, aby se nemuselo vše okolní přečíslovávat.)

Dodatek C

Programátorská dokumentace – část trénování

V tomto dodatku rozebereme postupně všechny součásti (kapitoly C.2 až C.5) trénování a testování strojového učení na analytické rovině tak, jak jsme o nich mluvili v kapitole 4.3.2. (Jedna z pěti součástí je samotná C4.5, o ní již mluvit nebudeme.) Všechny skripty jsou okomentované a jako součást dokumentace uložené na CD v adresáři `skripty/analyt-syns`; pro pohodlí je tam i skript `run`, který usnadňuje pouštění jednotlivých součástí. Ten si projdeme v kapitole C.6.

Protože zbylé dvě testované úlohy (přiřazování synsetů na základě t-roviny a přiřazování valenčních rámců na základě t-roviny) využívají klony zde vysvětlovaných skriptů, nebudeme se jim věnovat zvláště. Nacházejí se v adresářích `skripty/tekto-syns` a `skripty/tekto-ramce`.

Poté se v kapitole C.7 krátce zastavíme u skriptu `vypust_rysy.pl`, sloužícího v poslední fázi testování k vytipování nepotřebných rysů.

Ze všeho nejdříve se ale seznámíme se syntaxí souboru `rysy_tred`, neboť tento konfigurační soubor pak řídí další dění ovlivňujíc hned tři z pěti fází.

C.1 `rysy_tred`

Jméno tohoto textového souboru je pevně dáno. Postupně tři skripty ho načítají a informacemi v něm se řídí:

- `get_features.btred` z něj získá informace, které slovní druhy má vyhledávat a jaké rysy má ke kterému slovnímu druhu vypisovat;

- `build_names.pl` z něj načte názvy jednotlivých rysů a zjistí, které z vyhledaných rysů se budou používat pro natrénování;
- stejně tak i `classify.pl` zjistí, které rysy je možné zapomenout.

Formát je jednoduchý:

- komentář je řádka začínající `#`, před tím mohou být jen mezery a tabelátory;
- řádek s rysy je uvozen *návěstím* `<PoS>:`, kde PoS je značka pro některý rozpoznávaný slovní druh (v tomto tvaru):
  - `noun:` pro podstatná jména;
  - `adject:` pro přídavná jména;
  - `verb:` pro slovesa;
- rysy následují za návěstím oddělované mezerou;
- rysy pro jeden slovní druh mohou být na více řádcích (i proložené): pak musejí vždy začínat návěstím;
- každému rysu lze připojit prefix `-` (pomlčka), který zajistí, že rys bude vypsán, ale pro samotné trénování a testování nepoužit – testovací důvody;
- pořadí rysů v souboru určuje též i výsledné pořadí
  - proto je nutné zachovat *cíl* na posledním místě a
  - `mlemma` (nebo případně jiný rozlišovací rys – bude se podle něj třídit do souborů a na nich trénovat) dát na první místo. Typicky jako `-mlemma` (tedy s pomlčkou), protože jako rys je nepoužitelné.

Na konci souboru jsou ještě pro úplnost vyjmenovány a zakomentovány všechny rozpoznávané rysy, ale to není vlastnost formátu.

C.2 `get_features.btred`

Toto makro btredu prohledává všechny zadané soubory t-roviny (z nich je dostupná i m-rovina) a na výstup vypisuje vektory rysů.

Nejprve (`start_hook`) vyčistí (existuje-li od předchozího použití), nebo vytvoří (neexistuje-li) adresář `output/`. Potom načte seznam rysů zvlášť pro každý slovní druh (z `rysy_tred`, podle uvedené specifikace). Ignoruje deaktivující `-` (pomlčku) před rysem: v této fázi to není podstatné. Otevře soubory pro výpis vektorů rysů pro jednotlivé slovní druhy.

Potom (`main`) prochází všechny uzly všech stromů a když najde takový, který má přiřazenu anotaci ČWN, spustí na něj hlavní funkci `&print_attributes()`. Z hodnoty `ewn-sense` se zjistí slovní druh. Pozor, může se lišit slovní druh z WordNetu od slovního druhu z tagsetu na m-rovině. My se držíme přiřazení z WordNetu. Podle slovního druhu je až do konce zpracování tohoto uzlu přiřazen příslušný seznam rysů a výstupní soubor pro vektory.

Pro zrychlení následného vyhledávání rysů jsou některé často potřebné hodnoty přiřazeny jednou, na začátku. Konkrétně morfologický tagset (použit 11×), přímý otec (2×) a (budou-li potřeba) efektivní otec (10×), syn (8×), bratr (15×), řídící sloveso (7×), řídící predikát (6×) a řídící substantivum (6×). Vše další probíhá v cyklu, který prochází seznam všech požadovaných rysů a postupně je vyhledává a vypisuje do souboru, oddělené čárkami (syntax pro C4.5).

Pro každý rys je jeden blok kódu, zpravidla šest řádek, uvnitř `if`-testu. Výjimku tvoří rysy typu `ef_t_XX`, kde `XX` je libovolné číslo od 01 do 15, které vypisují danou posici morfologické značky: ty jsou zpracovány v jednom bloku. Vypisované hodnoty (povětšinou) opatřujeme textovým prefixem pro snadnější čitelnost (například `mgend_` pro gender z m-roviny, `ef_` pro rysy efektivního otce apod.).

V případě, že hodnot může být více (například slovní druh efektivního syna: efektivních synů může být mnoho), řeší se to zřetězením přes podtržítko a vyřazením duplikátů. (Například tedy pro čtyři syny se slovními druhy `TT`, `TT`, `TT` a `Z:` se vyrobí hodnota rysu `s_TT_Z\:` a ostatní `TT` se ztratí.) Část informace je ještě v rysu zachycujícím počet synů, část je ztracena – domníváme se, že je však podstatné, **že** se slovní druh může pod uzlem vyskytovat, a nikoli to, **kolikrát** je (konkrétně v této větě) znásoben.

Znaky, které C4.5 zakazuje v hodnotách rysů, již v této fázi „escapujeme“, jak se ukazuje u dvojtečky v předchozím odstavci.

Nemáme-li k disposici hodnotu rysu (uzel nemá žádného syna nebo otec uzlu je technický kořen stromu a nenese tedy žádnou morfologickou informaci), píšeme na výstup hodnotu `N/A`, kterou C4.5 interpretuje tak, že nemá smysl pro tento případ o hodnotě rysu hovořit. V tom případě žádný prefix nepřipojujeme.

U rysu `mlemma` odřezáváme přídavnou informace připojenou pomocí následujících sekvencí: `- ‘ _: _; _ _^(...)`

Když jsou vypsány všechny rysy, pokračuje se dalším uzlem. Po vyčerpání všech uzlů ve všech souborech (`exit_hook`) už jenom zavřeme všechny výstupní soubory.

Makro se spouští následovně:

```
btred -I get_features.btred cesta_k_PDT/train-?/*.a.gz
```

Parametr `-I` určuje jméno makra použitého btredem. O průchod každým uzlem se již stará makro samo. Jeho první řádek

```
#!btred -TN -e main()
```

říká, že ke zpracování makra má být použit btred, parametr `-T` zaručuje průchod všech stromů, `-N` všech uzlů v nich a `-e` určuje jméno funkce, která má být na každý uzel zavolána.

C.3 `build_names.pl`

Skript `build_names.pl` vyrábí z výstupního souboru btredu jeden ze vstupů pro C4.5, soubor typu `.names`, obsahující všechny přípustné hodnoty všech rysů. K tomu opět používá konfigurační soubor `rysy_tred`. Výsledný soubor vypisuje na `STDOUT`. Očekává dva parametry, typické volání je

```
./build_names.pl adject pos_a.data > universal_a.names
```

V tomto případě projde řádek za řádkem všechny vektory ze souboru `pos_a.data`, bude je považovat za vektory adjektiv a seznam hodnot rysů uloží do `universal_a.names`.

Skript nejprve uloží první parametr – slovní druh, nad kterým bude pracovat. Otevře soubor `rysy_tred` a načte z něj řádky začínající (v našem případě) `adject:` a za tím následující rysy rozseká do pole. Potom pole projde, aby si uložil indexy těch rysů, které začínaly pomlčkou – a které má ignorovat.

Poslední věcí, která je nutná před průchodem souborů vektorů, je takový trik. Budeme mít pole řetězců, kde v každé položce budeme řetězit možné hodnoty jednoho rysu. A začneme tím, že každému rysu tam uložíme speciální hodnotu „fake“ (dál nikdy nepoužitou). Tím zajistíme (pro alespoň jeden nalezený vektor ve vstupním souboru) dvě hodnoty pro každý rys. A to je požadavek programu C4.5 zmíněný v kapitole 4.3.2: nenajde-li u některého rysu alespoň dvě hodnoty, hlásí chybu. Pokud se ovšem provádějí testy jen na malém vzorku dat, může se stát, že některý rys bude nabývat stále jen jediné hodnoty.

V cyklu se načte jeden řádek, rozseká na jednotlivé rysy. Ty se projdou. Pokud daný rys není ignorovaný a pokud ještě není v řetězci uveden, připojí se. Zvlášť se ošetřuje poslední rys, cíl. Z něj se odřízne lemma a ponechá se jen číslo a ID synsetu.

Na konci vypíše v souladu s požadavky C4.5 na první řádek jméno cíle s tečkou, na další řádek všechny jeho možné hodnoty. A pak dvojice řádek: jméno rysu, dvojtečka; hodnoty rysů oddělené čárkami, tečka. Opět, ignorované rysy jsou vynechávány.

C.4 `classify.pl`

Stejně jako předchozí, i tento skript zpracovává soubor obsahující všechny vektory rysů pro jeden slovní druh. Jeho úkolem je roztřídit vektory do **skutečně mnoha** souborů podle první složky vektorů – tedy lemmatu. Soubory pojmenovává pomocí názvu společného souboru a lemmatu. Z `pos_n.data` tedy vzniknou soubory `pos_n_`*`lemma`*`.data`. V názvu souboru je lemma, zbavené diakritiky (případné kolise se řeší přidáváním podtržítek: `lemma_`, `lemma__`, ...). Proto je na prvním řádku souboru uloženo jako komentář lemma správně, aby bylo možno porovnat, do kterého souboru vektor skutečně patří.

Stejně jako předchozí skript i tento používá `rysy_tred`. Na druhý řádek souboru je jako komentář vložen seznam všech rysů získaných z PDT a na třetí řádek seznam těch, které nejsou ignorovány (tedy bez pomlčky). Dál už následují řádek za řádkem vektory, které přísluší právě k tomuto lemmatu.

Takto to celé vypadá pro trénovací data. S testovacími daty se to celé komplikuje. Důvodem je fakt, že v nich se mohou vyskytnout hodnoty rysů, které v trénovacích datech nebyly a proto nejsou ani zahrnuty do souboru `universal_?.names`. Spuštění skriptu tedy vypadá například takto:

```
./classify.pl [--test universal_n.names] noun pos_n.data
```

Je-li skript zavolán s prvním argumentem `--test`, očekává se v druhém argumentu jméno souboru s uloženým seznamem povolených hodnot. Na začátku je tento soubor načten. Pak ve skutečnosti ve chvíli, kdy se zjistí, že zpracovávaný rys není ignorován a má být vypsán, se místo

```
print(OUTPUT "$vector[$i], ");
```

volá funkce

```
print(OUTPUT &is_valid($vector[$i], $i).", ");
```

která v případě, že se zpracovávají trénovací data jednoduše vrátí `$vector[$i]`, zatímco jde-li o testovací data, zkontroluje, zda uvedená hodnota je přípustná (zachová se jako v prvním případě), nebo není (vrátí `?` (otazník), hodnotu chápanou v C4.5 jako „neznámá hodnota“).

C.5 results_process.pl

Po předchozí kapitole by logicky měla následovat čtvrtá o samotném trénování a testování rozhodovacích stromů. K tomu ale není již mnoho co dodat. Vše o programu bylo řečeno v kapitole 4.3.2. K zajištění správných vstupů do programu sloužily předchozí tři skripty. Takže pak je zkrátka C4.5 puštěna postupně na každé lemma a výsledek je uložen do souboru `pos_?_lemma.result`.

Poslední fází je zpracování právě těchto výstupních souborů a vypsání celkové úspěšnosti. Výsek ze souboru `.result` vypadá například následovně:

```
Evaluation on training data (8 items):
     Before Pruning           After Pruning
    ----------------   --------------------------
    Size      Errors   Size      Errors   Estimate
      15    0( 0.0%)      1    1(12.5%)    (30.0%)   <<
Evaluation on test data (2 items):
     Before Pruning           After Pruning
    ----------------   --------------------------
    Size      Errors   Size      Errors   Estimate
      15    1(50.0%)      1    0( 0.0%)    (30.0%)   <<
```

Nebudeme zde popisovat přesnou činnost skriptu, neboť počítá úspěšnost ve více kategoriích. Úspěšnost našeho experimentu pomocí rozhodovacích stromů, úspěšnost bez prořezání, počet případů, počet chyb (z toho kvůli významům neznámým z trénovacích dat) a baseline: pro každé lemma zvlášť, celkem pro všechna lemmata, z nich jen pro ta víceznačná a z nich jen pro ta, kde zbyl neprořezaný rozhodovací strom. To celé se navíc počítá jak pro trénovací data, tak pro testovací. Výstup vypadá například takto, kursivou jsou vyznačena nejdůležitější čísla:

```
FILE    TR.SIZE #TARGET #CASES #ERRORS THAT'S IN % BASELINE
ztrata:  (32)    5 in    40  4- 10 err 10.0-25.0% < 32.5%    >1 tsize
zvyk:     (1)    2 in     7   0-  1 err  0.0-14.3% = 14.3%    >1 targ
============================================================
TRAIN D Vyskyty: Z toho chyb: Unpruned: Procento:  Baseline:
Celkove:  53128      2791     2.30387%   5.25335%   8.53411%
>1 targ:  24321      2791     5.03269%  11.47568%  18.64233%
tree >1:   7471       826     5.13987%  11.05608%  34.38629%
============================================================
EVAL D. Vyskyty: Z toho chyb: Unpruned: Procento:  Baseline:
Celkove:   7154   649 (117)  10.19010%   9.07185%  10.13419%
>1 targ:   3382   587  (55)  19.72206%  17.35659%  19.60378%
tree >1:   1033   269  (26)  28.65440%  26.04066%  33.30106
```

C.6 run

Pro větší pohodlí při testování (jak našem testování programů, tak případně při testování dalších rysů a hypothes) existuje skript `run`, který nedělá nic jiného, než že předchozí skripty pouští ve správném pořadí, na správná data a se správnými parametry. Je psaný v Bashi. Argumenty příkazové řádky:

`0`	nejkratší varianta, z PDT vybere jen jediný soubor a nekontroluje úspěšnost na testovacích datech;
`1`	skupina několika souborů z `train-1/`;
`2`	soubory Českomoravského profitu z `train-1/` plus část testovacích dat;
`3`	všechny soubory z `train-1/` plus část testovacích dat;
jiný	testuje na souborech, které jsou zadány;
žádný	kompletní test na všech trénovacích datech a všech testovacích.

Předpokládejme nadále, že byl skript zavolán bez parametrů a čeká nás kompletní běh programu. Nejprve se spustí btred s makrem `get_features.btred` pro testovací data. Výsledek se přesune do adresáře `outest/`. Poté se btred zopakuje na trénovacích datech. V obou případech je z výstupu do konsole odfiltrováno chybové hlášení btredu o „Broken pipe“.

Dále je pro všechny slovní druhy vygenerován soupis hodnot rysů.

Dochází k roztřídění vektorů do souborů podle lemmatu. Nejprve pro testovací data – s vhodným parametrem `--test` – a všechny soubory jsou přejmenovány na soubory s příponou `.test`, následují trénovací data.

Pro každý soubor `pos_`*x*`_`*lemma*`.data` je vytvořen symbolický link `pos_`*x*`_`*lemma*`.names` → `universal_`*x*`.names`.

Každé lemma je předáno C4.5, aby ho zpracovala a vypsala výsledky. V našem případě kompletního testu pouze každé lemma mající testovací data.

Na závěr vyhodnocení úspěšnosti, přepočítání baseline pro danou množinu dat. A počet nespatřených lemmat a jejich výskytů.

C.7 vypust_rysy.pl

Poněkud stranou od předchozích skriptů stojí tento jednoduchý pracovní skript, který sloužil k testování hypothesy z kapitoly 4.3.3. Umožňuje z předem nagenerovaných souborů vektorů vypustit postupně všechny složky a v souladu s tím upravit i soubor `universal.names`.

První parametr je adresář se soubory vektorů, druhý parametr je maska (perlovský regulární výraz) popisující soubory, na kterých se má výpuštění provést, třetí je soubor `.names` a čtvrtý parametr je cílový adresář, kam budou uloženy zúžené varianty původních souborů.

Skript postupně vypustí první až $n-1$-ní složku a výstup ukládá do odpovídajících podadresářů `1` až `n`. Skončí v okamžiku, kdy už není další složka, kterou by bylo možno vypustit.

Pro i-tou složku postupuje takto: vytvoří adresář *`cíl`*`/i`, prochází všechny soubory vyhovující masce. Otevře soubor pro čtení a druhý vytvoří v cílovém adresáři pro zápis. Pokud od tohoto lemmatu nevede symbolický link k souboru `.names`, vytvoří ho. Pak prochází soubor a z každého řádku, kde je alespoň $i+1$ složek, vypíše složky $1, \ldots, i-1, i+1, \ldots, n$.

Když projde všechny soubory, otevře soubor `.names` a okopíruje ho celý bez příslušného řádku odpovídajícího vypouštěnému rysu.

Zkontroluje, jestli při posledním vypouštění $i+1 = n$. Pokud ano, práce skončila, neboť zbývající rys už je cíl a ten se nevypouští. V opačném případě zvedne čítač o jedna a začne nové kolo.

Dodatek D

Seznam používaných rysů

V této příloze přinášíme seznam všech rysů, které jsme v rámci naší diplomové práce implementovali. U každého uvádíme jméno, stručný popis a několik příkladů hodnot. Rysy rozdělíme do několika skupin.

Pro všechny rysy, které se vztahují k více uzlům (pád všech synů), platí, že se duplicity zahazují (zmíněno v C.2). Výjimku tvoří pouze kompletní výpisy z tagsetu (rysy typu `něco_t_07`).

D.1 ČWN, a-rovina

Rysy získané z desambiguovaného uzlu samotného:

`mlemma` morfologické lemma (`specifický, týden, postupný`);

`afun` analytická funkce (`Adv, AuxP, Sb`);

`msense` přívesek rozdělující významy u lemmatu na m-rovině (`msens_4B, msens_2`);

`form` tvar slova (`specifických, týdnech, postupnému`);

`tag` posiční morfologická značka (`AANS2----1A----, NNFS4-----A----`);

`empos` slovní druh z tagu m-roviny + slovní druh z Euro WordNetu (`AA_n, NN_n, AG_a`);

`m_degr` stupeň *(pro adjektiva)*, (`mdegr_1, mdegr_3`);

`m_subpos` slovní poddruh (`mspos_A, mspos_G, mspos_b`);

`m_gender` rod *(pro substantiva a adjektiva)* (`mgend_F, mgend_N, mgend_T`);

`m_number` číslo *(pro substantiva a adjektiva)* (`mnumb_P, mnumb_W`);

`m_case` pád *(pro substantiva a adjektiva)* (`mcase_2, mcase_7, mcase_X`);

`m_person` osoba *(pro slovesa)* (`mpers_3`, `mpers_1`);

`m_tense` čas *(pro slovesa)* (`mtens_P`, `mtens_R`);

`m_neg` negace (`mneg_A, mneg_N`);

`m_voice` slovesný rod *(pro slovesa)* (`mvoic_A`, `mvoic_P`);

`id` unikátní identifikátor (`m#m-cmpr9410-001-p4s3w25`);

Rysy získané z efektivních otců:

`ef_nof` počet efektivních rodičů; pokud je jen jeden, je porovnán s přímým rodičem – jsou-li stejné, připíše se značka '+'; je-li efektivním rodičem technický uzel, počet se nastaví na 0 (`f_0, f_1, f_1+, f_7`);

`ef_LR` leží otec vlevo nebo vpravo od uzlu ve stromě? (`f_L, f_R, f_R_L`);

`ef_pos` slovní druh otce (`f_PL, f_Vp_Vf, f_C\=_NN_Db`);

`ef_case` pád otce *(otec substantivum/adjektivum)* (`f_4_2, f_-_4_1_7, f_4_X`);

`ef_pers` osoba otce *(otec sloveso)* (`f_1_3, f_X_3`);

`ef_voic` rod slovesa *(otec sloveso)* (`f_A_P, f_P`);

`ef_degr` stupeň otce *(otec adjektivum)* (`f_2, f_3`);

`ef_t_<1-15>` posice v morfologické značce otce, zastupuje 15 rysů (`ef_t01_N_N`);

`ef_afun` analytická funkce otce (`f_ExD_Sb, f_ObjAtr`);

Rys získaný z efektivního otce:

`df_afun` analytická funkce **přímého** otce (`df_AuxP, df_Pnom`);

Rysy získané z efektivních synů:

`s_nof` počet efektivních synů (`s_3, s_4, s_0`);

`s_LR` leží syn vlevo nebo vpravo od uzlu ve stromě? (`s_R, s_R_L`);

`s_pos` slovní druh syna (`s_AA_NN, s_AA_RR_J\^_Z\:`);

`s_case` pád syna (`s_-_4_3_2`);

`s_pers` osoba syna (`s_2, s_-_3, s_X`);

`s_voic` rod slovesa (`s_A, s_A_-`);

`s_degr` stupeň syna (`s_1, s_-_2`);

`s_t_<1-15>` posice v morfologické značce syna (`s_t01_N_N, s_t05_4_4_7`);

`s_afun` analytická funkce syna (`s_Atr_AuxP_AuxZ_AuxX_Pred, s_Atr_ExD`);

Rysy získané z bratrů:

`b_nof` počet efektivních bratrů ve stromě (`b_19, b_2, b_59`);

`lb_pos` slovní druh levých bratrů (`lb_Cl_Vc_Vf_Db, lb_C\=_RR_Dg_Z\:`);

`rb_pos` slovní druh pravých bratrů (`rb_AA, rb_AA_RV_J\^`);

`lb_case`

`rb_case` pád bratra (`lb_6_1_4_2, rb_6_1_3_X`);

`lb_pers`

`rb_pers` osoba bratra (`lb_1_X, rb_-_1_3`);

`lb_voic`

`rb_voic` slovesný rod bratra (`lb_-_A_P, rb_A`);

`lb_degr`

`rb_degr` stupeň bratra (`lb_1_2, rb_-_3_2`);

`lb_t_<1-15>`

`rb_t_<1-15>` posice v morfologické značce bratra (`rb_t01_N_N, lb_t05_4_4_7`);

`lb_afun`

`rb_afun` analytická funkce bratra (`lb_AtvV_AuxX_AuxV_Obj, rb_Atr_AuxY_AuxP_AuxZ_AuxX`);

Rysy získané z řídícího slovesa:

`gv_LR` leží vlevo/vpravo (`gv_L, gv_R`);

`gv_spos` slovní poddruh (`gv_i, gv_p, gv_B`);

`gv_pers` osoba (`gv_3, gv_X`);

`gv_voic` slovesný rod (`gv_A, gv_P`);

`gv_tens` čas (`gv_F, gv_P, gv_R`);

`gv_t_<1-15>` posice v morfologické značce řídícího slovesa (`gv_t05_4_4_7`);

`gv_afun` analytická funkce (`gv_AtrAtr, gv_ExD`);

Rysy získané z řídícího predikátu:

`gp_LR` leží vlevo/vpravo (`gp_L, gp_R`);

`gp_pos` slovní druh+poddruh (`gp_Db, gp_Vi, gp_Vp`);

`gp_pers` osoba (`gp_1, gp_2`);

`gp_voic` slovesný rod (`gp_A, gp_P);`

`gp_t_<1-15>` posice v morfologické značce řídícího predikátu (`gp_t05_4_4_7`);

`gp_tens` čas (`gp_P, gp_R, gp_X`);

Rysy získané z řídícího substantiva *(pro adjektiva):*

`gs_LR` leží vlevo/vpravo (`gs_L, gs_R`);

`gs_spos` slovesný poddruh (`gs_N, N/A`);

`gs_case` pád (`gs_4, gs_5`);

`gs_numb` číslo (`gs_S, gs_X`);

`gs_t_<1-15>` posice v morfologické značce řídícího substantiva (`gs_t01_N_N`);

`gs_afun` analytická funkce (`gs_AdvAtr, gs_ObjAtr`);

Cíl:

`ewn` přiřazený EWN synset (`2==995280, 3==1002703`);

D.2 ČWN, t-rovina

`t_lemma` tektogramatické lemma (`cítění`, `zavraždění`);

`functor` funktor (`RESL`, `COND`);

`sempos` sémantický slovní druh (`n.denot`, `n.denot.neg`);

`mpos` kombinace slovního druhu z m-roviny a slovního druhu z ČWN (`AC_n`, `NN_n`);

`gender` gramatém rodu (*pro substantiva a adjektiva*) (`fem`, `inan`, `neut`);

`number` gramatém čísla (*pro substantiva a adjektiva*) (`nr`, `pl`);

`degcmp` gramatém stupně (*pro adjektiva*) (`pos`, `comp`);

`sentmod` větná modalita (oznamovací, rozkazovací, ...) (`enunc`, `excl`);

`verbmod` slovesná modalita (*pro slovesa*) (`cdn`, `imp`, `ind`);

`tense` gramatém času (*pro slovesa*) (`post`, `sim`);

`aspect` gramatém slovesného vidu (*pro slovesa*) (`cpl`, `nr`, `proc`);

`lemmatch` shoda morfologického a tektogramatického lemmatu; nabývá následujících hodnot:

- `match` jsou-li totožné („auto“ a „auto“);
- `sense` jsou-li totožné až na příponu lemmatu („známý-2_^(co_známe)“ a „známý“);
- `reflex` jsou-li totožné až na reflexivitu („zbavit“ a „zbavit se“);
- `case` jsou-li totožné až na velikost písmen („úřad“ a „Úřad“);
- `differs` pokud se liší („základ“ a „novelizace“).

D.3 PDT-VALLEX

Všechna následující rysy nabývají pouze hodnot typu boolean, proto ani nebudeme uvádět příklady hodnot:

`jst_dep-on-sbs` je slovo přímo závislé na substantivu?

`jst_is-und-sbs` je slovo pod substantivem, někde v podstromu?

`jst_has-dep-sbs` má přímo závislé substantivum?

`jst_is-abv-sbs` má substantivum někde ve svém podstromu?

`jst_dep-on-adj` je slovo přímo závislé na adjektivu?

`jst_is-und-adj` je slovo pod adjektivem?

`jst_has-dep-adj` má závislé adjektivum?

`jst_is-abv-adj` je někde nad adjektivem?

`jst_has-dep-vrb` má závislé sloveso?

`jst_is-abv-vrb` je někde nad slovesem?

`jst_has-dep-con-ze` má pod sebou spojku „že“?

`jst_has-dep-con-az` má pod sebou spojku „až“?

`jst_has-dep-con-at` má pod sebou spojku „ať“?

`jst_has-dep-con-jestli` má pod sebou spojku „jestli“, nebo „jestliže“?

`jst_has-dep-con-zda` má pod sebou spojku „zda“, nebo „zdali“?

`jst_has-dep-con-aby` má pod sebou spojku „aby“?

`jst_has-dep-con-jak` má pod sebou spojku „jak“?

`jst_any-dep-in<1-7>` má pod sebou cokoli v daném pádě?

`jst_N-dep-in<1-7>` má pod sebou substantivum nebo substantivní zájmeno v daném pádě?

`jst_A-dep-in<1-7>` má pod sebou adjektivum nebo adjektivní zájmeno v daném pádě?

`jst_R-dep-in<1-7>` má pod sebou předložkovou skupinu v daném pádě?

`jst_has-dep-degr<1-3>` má pod sebou adjektivum daného stupně?

`valframe` přiřazený valenční rámec (`f1, f5`);