Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Bc. Lenka Smejkalová

Typické vzory užívání anglických sloves

Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: RNDr. Martin Holub, Ph.D.
Studijní program: Informatika, matematická lingvistika

2010

Na tomto místě bych ráda poděkovala hlavně RNDr. Martinu Holubovi, Ph.D. za vedení diplomové práce, poskytnuté cenné rady a množství času věnované konzultacím. Další dík patří Mgr. Silvii Cinkové, Ph.D. za konzultace ohledně anglické gramatiky a Ing. Zdeňkovi Žabokrtskému, Ph.D. za pomoc s nástrojem TectoMT. V neposlední řadě bych také ráda poděkovala své rodině a přátelům, kteří mě podporovali při studiu a měli se mnou trpělivost i během psaní této práce.

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

V Praze dne 6. srpna 2010 Bc. Lenka Smejkalová

Obsah

1 Úvod **6**

2 Pattern Dictionary of English Verbs (PDEV) **8**
2.1 CPA a PDEV 8
2.1.1 Struktura patternu 8
2.1.2 Vytváření patternů a značkování konkordancí 10
2.2 Podobné projekty - pro angličtinu 10
2.2.1 FrameNet 10
2.2.2 PropBank 11
2.2.3 VerbNet 11
2.3 Podobné projekty - pro češtinu 15
2.3.1 VALLEX 2.5 15
2.3.2 VerbaLex 15

3 Současný stav projektu PDEV **17**
3.1 Platforma PDEV 17
3.1.1 Definice patternů ve formátu XML 17
3.2 Statistiky sloves 20
3.3 Mezianotátorská shoda 21
3.3.1 Diskuze 26
3.4 Další vývoj 27

4 Závislostní mikrokontext slovesa **29**
4.1 Stanfordský parser 29
4.1.1 Zpracování výstupu stanfordských závislostí 33
4.2 Další parsery a metody 38
4.2.1 Charniak-Johnsonův parser 38
4.2.2 McDonaldův parser 38
4.2.3 Sketch Engine 39
4.3 Evaluace metod pro syntaktickou analýzu 40
4.3.1 Evaluační míra 40
4.3.2 Testovací data 40
4.3.3 Evaluace zpracování výstupu stanfordských závislostí . . 44
4.3.4 Evaluace parserů a metod 44
4.4 Shrnutí 46

5 Systém sémantických typů **47**
5.1 Použité nástroje 47
5.1.1 Stanfordský NE klasifikátor 48
5.1.2 Slovník 48
5.2 Extrakce argumentů a přiřazení sémantických typů 49
5.3 Definice statistických charakteristik 50
5.4 Čištění seznamů a evaluace 52
5.4.1 Anotace a testovací data 52
5.5 Další vývoj 53

6 Automatické rozpoznávání patternů **56**
6.1 Motivace 56
6.2 Data 57
6.3 Heuristický klasifikátor 57
6.4 Evaluace úspěšnosti přiřazení patternů 60
6.4.1 Metoda evaluace 60
6.4.2 Průběh a výsledky evaluace 60
6.5 Další vývoj 63
6.6 Shrnutí 64

7 Závěr **65**

A Seznam použitých zkratek **66**

B Definice patternů **67**

C Populace sémantických typů **71**

D Obsah přiloženého CD **76**

E Popis programů na CD **77**

Literatura **80**

Název práce: Typické vzory užívání anglických sloves
Autor: Bc. Lenka Smejkalová
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: RNDr. Martin Holub, Ph.D.
e-mail vedoucího: holub@ufal.mff.cuni.cz

Abstrakt: Metoda CPA (Corpus Pattern Analysis) je korpusová metoda, která analyzuje typické vzory užívání slov v jazykovém korpusu a popisuje význam sloves pomocí kontextových preferencí definovaných jak syntakticky, tak sémanticky [1]. V současné době pomocí CPA a s využitím Britského národního korpusu (BNC) vzniká *Slovník vzorů užívání anglických sloves* (PDEV, Pattern Dictionary of English Verbs) [1, 2].

Diplomová práce popisuje současný stav slovníku PDEV, zabývá se podrobnou analýzou dostupných dat o typických vzorech užívání anglických sloves a zkoumá, jak lze využít PDEV pro automatickou analýzu lexikálního významu.

Dále diplomová práce obsahuje návrh a implementaci procedur pro podporu dalšího vývoje slovníku PDEV. První z nich je extrakce slovesných argumentů z výstupu syntaktické analýzy angličtiny. Druhá procedura využívá tyto extrahované argumenty k vytváření seznamů lexikálních jednotek realizujících sémantické typy. Poslední navržená procedura automaticky rozpoznává typické vzory užívání sloves za pomoci uvedených seznamů získaných předchozí procedurou.

Součástí práce je též vyhodnocení mezianotátorské shody, evaluace automatické extrakce slovesných argumentů z anglické věty a experimentální ověření účinnosti navržených procedur pro extrakci lexikálních jednotek realizujících jednotlivé sémantické typy a pro automatické rozpoznávání typických vzorů užívání.
Klíčová slova: korpus, slovník, závislostní syntax, anglická slovesa, významy sloves

Title: Typical Usage Patterns of English Verbs
Author: Bc. Lenka Smejkalová
Department: Institute of Formal and Applied Linguistics
Supervisor: RNDr. Martin Holub, Ph.D.
Supervisor's e-mail address: holub@ufal.mff.cuni.cz

Abstract: Corpus Pattern Analysis (CPA) is a corpus-based method that explores typical usage patterns of verbs in a text corpus, and describes meaning of verbs by means of contextual preferences defined both syntactically and semantically [1]. CPA in conjuction with the British National Corpus (BNC) is currently used to create The Pattern Dictionary of English Verbs (PDEV) [1, 2].

The thesis describes the current status of the PDEV, presents a thorough analysis of available data on typical usage patterns and explores possible applications of the PDEV for automatic lexical analysis.

In this thesis procedures usable in further PDEV development have been designed and implemented. The first of them automatically extracts arguments of verbs from an output of English syntactic analysis. The second one uses the extracted arguments to create lists of lexical units that realize semantic types. The last procedure uses these lists to automatically recognize typical usage patterns of verbs.

The thesis also evaluates inter-annotator agreement, automatic extraction of verb arguments in/from English sentence, and effectiveness of the proposed procedures in the extraction of lexical units that realize semantic types and in automatic recognition of typical usage patterns.
Keywords: corpus, dictionary, dependency syntax, English verbs, verb meanings

Kapitola 1

Úvod

Metoda CPA (z angl. Corpus Pattern Analysis) je korpusová metoda, která analyzuje typické vzory užívání slov (patterny[1]) v jazykovém korpusu a popisuje význam sloves pomocí kontextových preferencí definovaných jak syntakticky, tak sémanticky [1]. V současné době pomocí metody CPA a s využitím Britského národního korpusu (BNC) vzniká *Slovník vzorů užívání anglických sloves* (PDEV, z angl. Pattern Dictionary of English Verbs) [1, 2].

V rámci tohoto projektu bylo již zkompilováno téměř 700 anglických sloves, která pokrývají cca 10 % slovesných výskytů v BNC. Vedle slovníkových hesel obsahujících definici typických vzorů užívání sloves jsou též veřejně dostupná rozsáhlá korpusová data se slovesy označkovanými podle PDEV [1].

Další nedílnou součástí CPA je empiricky vybudovaný systém sémantických typů [1, 3]. Kombinací sémantických typů a syntaxe metoda CPA originálním způsobem konkuruje tradičním přístupům k automatickému rozpoznávání významu sloves [4].

Cílem této diplomové práce je analyzovat a co nejlépe využít dostupná data o typických vzorech užívání anglických sloves. Diplomová práce zkoumá, jak lze využít PDEV pro automatickou analýzu lexikálního významu, a má vést k návrhu procedur, které podpoří automatické rozpoznávání nebo pomohou zefektivnit další vývoj slovníku PDEV.

Dalším cílem bylo provést podrobnou analýzu dostupných dat včetně statistického vyhodnocení míry mezianotátorské shody a experimentální ověření účinnosti navržených procedur – extrakce lexikálních jednotek realizujících jednotlivé sémantické typy s využitím automatické syntaktické analýzy angličtiny a automatické rozpoznávání typických vzorů užívání.

Diplomová práce obsahuje 7 kapitol včetně úvodu a závěru. Kapitola 2 vysvětluje vztah mezi metodou CPA a slovníkem PDEV, popisuje jeho základní rysy a srovnává ho s podobnými projekty pro angličtinu a pro češtinu. Kapitola 3 se zabývá technickou stránkou a současným stavem projektu PDEV, uvádí výsledky mezianotátorské shody a nastiňuje další předpokládaný vývoj slovníku.

Kapitoly 4 a 5 se věnují nezbytným dílčím cílům – automatické syntaktické analýze a lexikální realizaci sémantických typů. Kapitola 4 se podrobně

[1] Anglický termín *pattern* budeme používat v celém textu.

zabývá automatickou syntaktickou analýzou se zaměřením na lokální kontext slovesa – extrakci slovesných argumentů. Je zde popsán použitý Stanfordský parser včetně zpracování jeho výstupu. Součástí kapitoly je také vyhodnocení úspěšnosti Stanfordského parseru a dalších parserů či metod, které jsou zde rovněž popsány. V kapitole 5 je extrakce argumentů prakticky využita pro srovnávání těchto argumentů se sémantickými typy definovanými na kolokačních pozicích popsaných v patternech a následné vytváření seznamů slov, která realizují jednotlivé sémantické typy.

V předposlední kapitole 6 je navržen a otestován jednoduchý heuristický klasifikátor pro přiřazování patternu slovesům. Nástroj využívá výstupy z předchozích kapitol – z kapitoly 4 využívá automatickou závislostní analýzu anglické věty včetně extrakce slovesných argumentů a z kapitoly 5 používá realizace sémantických typů. V rámci evaluace je zde navržen jednoduchý baseline, vůči kterému se provádí experimentální porovnání účinnosti navrženého klasifikátoru.

Práce je zakončena kapitolou 7, která shrnuje celou práci a hodnotí jakých cílů se podařilo dosáhnout a jakým způsobem.

Kapitola 2

Pattern Dictionary of English Verbs (PDEV)

2.1 CPA a PDEV

Metoda CPA (Corpus Pattern Analysis) vznikla na základě *Theory of Norms and Exploitations* ([2, 5]), jejímž autorem je Patrick Hanks.

Projekt Pattern Dictionary of English Verbs (PDEV, [2]) vzniká od roku 2004 použitím metody CPA a pomocí Britského národního korpusu (BNC).

Celý BNC korpus obsahuje přibližně 100 milionů slov. K vývoji slovníku PDEV je použita pouze jeho část – BNC50, která obsahuje 50 milionů slov. Tato část vznikla odstraněním mluvených textů a některých psaných textů, které mají specifický žánr.

Metoda CPA důsledně splňuje Sinclairův koncept zachycení významů v typických vzorech užívání jazyka. John Sinclair, nestor korpusové lingvistiky, kritizoval oddělování gramatiky a lexikonu. Gramatika v krajních případech pouze popisuje *formu* lexikální jednotky s ohledem na její potenciální kontext, zatímco slovník popisuje *význam* obsažený v základním tvaru lexikální jednotky bez ohledu na její kontext. Podle Sinclaira [6] jsou forma a význam nejenom úzce svázány, ale dokonce musí být identické, protože většina víceznačností v jazyce lze rozhodnout na základě znalosti kontextu.

Současná metoda CPA zachycuje „normální“, např. poměrně frekventované, použití daného slovesa pomocí typických vzorů užívání (patternů). Cílem není obsáhnout všechny možné realizace všech významů slovesa, ale relativně častá užití – tzv. *normy*.

2.1.1 Struktura patternu

Definice patternu se skládá z globálních atributů, popisů kolokačních pozic (agent, objekty, atd.) a z implikatury:

- *Globální atributy patternu* – globální atributy se vztahují k celému patternu a určují, zde se jedná o idiom, frázové sloveso, lze uvést také doménu slovesa v daném vzoru užívání, zda sloveso vyžaduje objekt atd.

- *Popis agenta* (v PDEV se nazývá subject) – agent je popsán pomocí sémantického typu, který může být upřesněn sémantickou rolí, jak je uvedeno v ukázce na obrázku 2.1, kde sémantický typ *Human* je upřesněn sémantickou rolí *Author*. Pokud je sémantický typ v patternu použit vícekrát, je k němu připojeno pořadové číslo (např. *[[Human 1 — Animal 1]] abandon [[Human 2 — Animal 2]] (to [[Anything = Bad]])*).

  Další možností je uvedení lexsetu – výčtu lexikálních jednotek, které typicky reprezentují daný sémantický typ. Lexset může být vyplněn i bez sémantického typu, například když uvedená slova nelze shrnout pod jeden sémantický typ nebo když chceme omezit množinu slov přípustných pro danou kolokační pozici (*[[Human]] tell* {*truth*}).

  Celou definici agenta lze libovolně opakovat, což je pak označováno za *alternaci agenta*.

- *Popis objektu/objektů* – objekt se dělí na přímý a nepřímý, který není příliš častý. Popis objektů je shodný jako popis agenta.

- *Popis adverbiálů* – adverbiál v PDEV je buď předložková fráze nebo příslovce. Adverbiálů může být v jednom patternů definováno více a každý může mít navíc své alternace. V popisu adverbiálu je možné vyplnit funkci adverbiálu, obligatornost, předložku a předložkový objekt (je popsán stejně jako agent a objekt) nebo příslovce.

- *Popis komplementu* – komplement neboli doplnění může být buď subjektové nebo objektové. Tento jev je v PDEV spíše ojedinělý. Příkladem je sloveso *bleed*, kde objektovým doplněním může být *white* nebo *dry*: *[[Human 1 — Institution 1]] bleed [[Human 2 — Institution 2]]* {*(white — dry)*}

- *Popis slovesných klauzí* – některé argumenty PDEV rozlišuje na základě jejich povrchové struktury: that-clause, wh-clause, to+inf, -ing a quote. Klauze může být objektová nebo příslovečná.[1]

- *Implikatura* – vysvětluje význam daného vzoru užívání slovesa pomocí stejných sémantických typů, které jsou uvedeny v popisu jednotlivých kolokačních pozic.

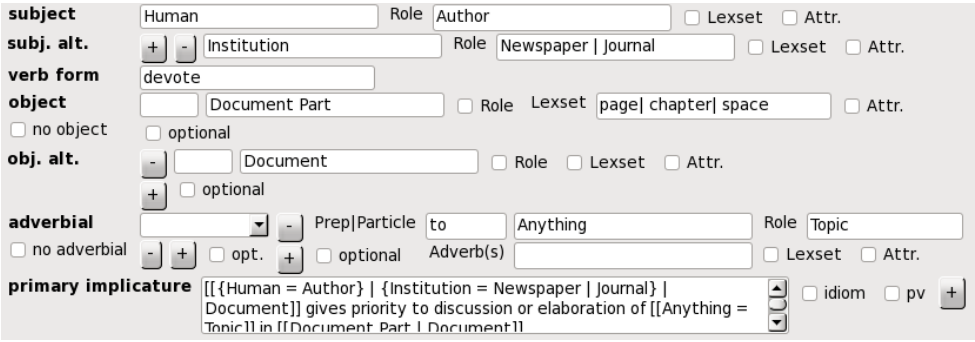
Stručný zápis patternu je určen tzv. propozicí, která pro uvedený pattern č. 2 slovesa *devote* vypadá následovně:

```
[[{Human = Author} | {Institution = Newspaper | Journal}]] devote
[[Document Part | Document]] {to  [[Anything = Topic]]}
```

Různé typy závorek v zápisu propozice znamenají:

- `[[ ]]` ... sémantický typ
- `( )` ... nepovinný argument
- `{ }` ... pouze shlukují

[1] V příští verzi by klauze měly být uvedeny přímo jako možnost realizace argumentu.



Obrázek 2.1: Pattern č. 2 slovesa *devote* – detail

2.1.2 Vytváření patternů a značkování konkordancí

Patterny vytváří manuálně s pomocí automatických procedur pouze jeden lexikograf – autor metody CPA Patrick Hanks. Na začátku zpracování nového slovesa se vybere náhodný referenční vzorek o standardní velikosti 250 vět (v některých případech i více, např. 500 nebo 1 000). Pomocí aplikace Sketch Engine lexikograf nejprve analyzuje kolokace slovesa a vytvoří seznam možných kolokátů. Na základě tohoto seznamu lexikograf ručně vytvoří první návrh patternů. Následně označkuje referenční vzorek a doladí patterny, např. spojí dva blízké patterny do jednoho, jiný rozdělí na dva, upraví sémantický typ, sémantickou roli či množinu lexikálních jednotek, popř. přidá další pattern.

Účelem vytváření patternů není pokrýt všechny možné případy, které mohou nastat. Většina konkordancí je označena číslem patternu. Užití slovesa, která neodpovídají žádnému patternu, jsou označena značkou u (*undecidable*) a ve větách, kde je slovo chybně rozpoznáno jako sloveso, se používá značka x (*unmarkable*).

Dále může být k číslu patternu připojeno písmeno e, které vyjadřuje, že ve větě nedochází k normálnímu použití patternu, ale jedná se o výjimku (*exploitation*). Hanks rozpoznává asi 10 základních druhů, např. figurativní použití (věta odpovídá struktuře patternu, ale význam je jiný), ironie, elipsa (chybějící argument), netypický argument, atd. [2].

2.2 Podobné projekty - pro angličtinu

Popisem některých zde uvedených projektů se zabývaly již dizertační práce Zdeňka Žabokrtského [7] a Dany Hlaváčkové [8].

2.2.1 FrameNet

Zázemím projektu FrameNet [9] je International Computer Science Institute v Berkeley (USA). Autorem projektu je Charles J. Fillmore, americký lingvista, který vymyslel teorii pádové gramatiky (case grammar). Základním prvkem je pádový rámec, který obsahuje údaje o tom, se kterými pády se sloveso pojí fa-

kultativně nebo obligatorně. Termínu „pád“ se zde používá pro označení druhu vztahu mezi slovesem a jeho doplněním.

Později na konceptu pádové gramatiky založil zásadní přístup k systematickému zpracování vztahů mezi syntaxí a lexikem. Výsledkem je projekt FrameNet, který hierarchicky uspořádává sémantické rámce.

Cílem projektu je zachytit všechny souvislosti všech slov ve všech významech na syntaktické i sémantické úrovni.

Lexikální jednotky se sdružují do sémantických rámců reprezentujích prototypické situace a stavy. Každý sémantický rámec má svou vlastní množinu sémantických rolí (např. *Speaker, Message, Adreessee, Topic* a *Medium* u rámce *Communication*), které jsou specifické pro danou skupinu lexikálních jednotek.

Na obrázku 2.2 je ukázka sémantického rámce *Abandonment*, který obsahuje lexikální jednotky *abandon.v, abandoned.a, abandonment.n, forget.v* a *leave.v.*

Sémantické rámce jsou uspořádány do hierarchie, takže specifičtější rámce dědí vlastnosti od obecnějších rámců.

V současné době lexikální databáze FrameNet obsahuje přibližně 11 600 lexikálních jednotek, z nichž 6 800 jsou plně anotovány v 960 sémantických rámcích.

2.2.2 PropBank

Pod vedením Marthy Palmerové z University of Pennsylvania vzniká projekt PropBank (Proposition Bank) [10] přidáváním sémantických informací do syntakticky anotovaného korpusu Penn TreeBank (PTB), který obsahuje texty z Wall Street Journal (WSJ).

Při sémantické anotaci bylo nejprve potřeba rozlišit od sebe významy daného slovesa. Slovníkové heslo je pak rozděleno do tzv. *rolesets*, které odpovídají jednotlivým významům. V každém roleset je uvedena množina argumentů se stručným vysvětlením (tzv. *roles*). Pro argumenty jsou používány zkratky *Arg0, ... Arg5*, kde *Arg0* většinou odpovídá logickému subjektu (Agent) a *Arg1* objektu (Patient). Sloveso je označeno jako *Rel* a modifikátory pomocí *ArgM.*

Dále následuje jedna či více příkladových vět s použitím slovesa v daném významu, ve kterých jsou explicitně označeny argumenty.

Na obrázku 2.3 je příklad slovesa *destroy*, které zde má pouze jeden význam, tedy jeden roleset. Jsou zde popsány tři argumenty *Arg0* – destroyer (konatel děje, agent), *Arg1* – think destroyed (zasažený předmět, patient) a *Arg2* – instrument of destruction (nástroj, instrument). Dále následují dva příklady – první je jeden s použitím instrumentu, druhý příklad je bez něj.

2.2.3 VerbNet

Dalším projektem pro zachycení anglických sloves je VerbNet [11], který navazuje na PropBank. Jeho hlavními autory jsou uváděny Martha Palmerová a Karin Kipper Schulerová. VerbNet využívá syntaktické rámce z PropBank a je kompatibilní se sémantickou sítí WordNet. Slovesa jsou rozdělována do hierarchicky uspořádaných syntakticko-sémantických tříd.

Abandonment

Definition:

An Agent leaves behind a Theme effectively rendering it no longer within their control or of the normal security as one's property.

Carolyn ABANDONED her car and jumped on a red double decker bus.

Perhaps he LEFT the key in the ignition

ABANDONMENT of a child is considered to be a serious crime in many jurisdictions.

FEs:

Core:

Agent [Age]	The Agent is the person who acts to leave behind the Theme.
Theme [The]	The Theme is the entity that is relinguished to no one from the Agent's possession.

Non-Core:

Depictive []	The FE Depictive describes the Agent during the abandoning event.
Duration [Dur]	For what expanse of time the Agent has given up the Theme.
Explanation []	Explanation denotes a proposition from which the act of abandonment logically follows.
Manner [Man]	The style in which the Agent gives up the Theme.
Place [Pla]	The location where the Agent gives up the Theme.
Time [Tim]	When the Agent gives up the Theme.

Inherits From:
Is Inherited By:
Subframe of:
Has Subframes:
Precedes:
Is Preceded by:
Uses:
Is Used By:
Perspective on:
Is perspectivized in:
Is Causative of:
See Also:

Lexical Units

abandon.v, abandoned.a, abandonment.n, forget.v, leave.v

Obrázek 2.2: FrameNet – sémantický rámec *Abandonment*

Predicate: *destroy*

destroy: Frames file for 'destroy' based on sentences in financial subcorpus. Verbnet class destroy-44, no other framed members.

Roleset id: destroy.01 , *destroy*, vncls: 44, framnet:

destroy.01: Verbnet predicts an instrumental subject. They also bring the distinction between agent and instrument down to intentionality, which we've been trying to avoid. If you feel you can consistently make that decision, go ahead; otherwise only use arg2 when there is also an arg0.

Roles:

Arg0: *destroyer* (vnrole: 44-Agent)
Arg1: *thing destroyed* (vnrole: 44-Patient)
Arg2: *instrument of destruction* (vnrole: 44-Instrument)

Example: no instrument

Program traders argue that a reinstatement of the rule would destroy the ``pricing efficiency'' of the futures and stock markets.

Arg0: a reinstatement of the rule
ArgM-MOD: would
Rel: destroy
Arg1: the ``pricing efficiency'' of the futures and stock markets

Example: with instrument

Mary destroyed John's fragile self-esteem with a single dirty look.

Arg0: Mary
Rel: destroyed
Arg1: John's fragile self-esteem
Arg2: with a single dirty look

Obrázek 2.3: PropBank – *destroy*

No Comments

destroy-44

Members: 25, Frames: 3

POST COMMENT

CLASS HIERARCHY

DESTROY-44

NO SUBCLASSES

MEMBERS KEY

ANNIHILATE (FN 1; WN 1)
BLITZ (WN 1)
DAMAGE (FN 1; WN 1; G 1)
DECIMATE (WN 2)
DEMOLISH (FN 1; WN 1; G 1)
DESECRATE (WN 1, 2)
DESTROY (FN 1; WN 1, 2; G 1)
DEVASTATE (FN 1; WN 1; G 1)
DISFIGURE (WN 1)
EFFACE (WN 3)
EXTERMINATE (FN 1; WN 1; G 1)
EXTIRPATE (WN 1)
LEVEL
MAIM (WN 1)
MUTILATE (WN 1, 2, 3)
OBLITERATE (FN 1; WN 4)
RAVAGE (WN 2)
RAZE (FN 1; WN 1)
RUIN (WN 1; G 1)
SHATTER (FN 1; WN 1, 3; G 1)
UNDO
UNMAKE
VAPORIZE
WASTE (WN 5, 9; G 3)
WRECK (WN 1; G 1)

ROLES REF

- AGENT [+INT_CONTROL]
- PATIENT [+CONCRETE]
- INSTRUMENT [+CONCRETE]

FRAMES REF KEY

NP V NP
EXAMPLE "The Romans destroyed the city."
SYNTAX AGENT V PATIENT
SEMANTICS CAUSE(AGENT, E) DESTROYED(RESULT(E), PATIENT)

NP V NP PP.INSTRUMENT
EXAMPLE "The builders destroyed the warehouse with explosives."
SYNTAX AGENT V PATIENT {WITH} INSTRUMENT
SEMANTICS CAUSE(AGENT, E) USE(DURING(E), AGENT, INSTRUMENT) DESTROYED(RESULT(E), PATIENT)

NP.INSTRUMENT V NP
EXAMPLE "The explosives destroyed the warehouse."
SYNTAX INSTRUMENT V PATIENT
SEMANTICS CAUSE(?AGENT, E) USE(DURING(E), ?AGENT, INSTRUMENT) DESTROYED(RESULT(E), PATIENT)

Obrázek 2.4: VerbNet – sémantický rámec *destroy*

Každá třída obsahuje:

- množinu slovesných lemmat – členů třídy (*members*)
- množinu tématických rolí (*roles*) – např. *Actor, Agent, Patient, Recipient*
- množinu rámců (*frames*) popisujících syntaktické a sémantické užití tématických rolí včetně konkrétních příkladů

Na obrázku 2.4 je ukázka třídy *destroy*. Tato třída neobsahuje žádné další podtřídy a má celkem 25 členů, např. *demage, demolish, devastate* atd. Dále definuje tři tématické role (*Agent, Patient, Instrument*) a tři syntakticko-sémantické rámce.

2.3 Podobné projekty - pro češtinu

2.3.1 VALLEX 2.5

Valenční slovník českých sloves VALLEX vzniká v Ústavu formální a aplikované lingvistiky na MFF UK již od roku 2001. Jeho aktuální verze je VALLEX 2.5 z roku 2008 [12, 13].

VALLEX formálně popisuje valenční charakteristiky českých sloves dle teorie Funkčního generativního popisu (FGP). VALLEX je úzce spojen s Pražským závislostním korpusem (PDT).

VALLEX poskytuje informace o valenční struktuře českých sloves v jejich jednotlivých významech, které charakterizuje pomocí glos a příkladů. Pro jednotlivá valenční doplnění uvádí možná morfematická vyjádření, pokud jsou jejich formy dány slovesnou rekcí. Kromě těchto základních údajů uvádí i některé další syntaktické, případně syntakticko-sémantické charakteristiky jako je vlastnost kontroly, možnost recipročního užití či syntakticko-sémantická třída slovesa.

Slovník obsahuje 2 730 lexémů, které zahrnují celkem 6 460 lexikálních jednotek, vidové protějšky jsou zachyceny v jednom lexému. Pokud bychom počítali dokonavá a nedokonavá slovesa zvlášť, měli bychom 4 250 sloves. Hlavním kritériem pro výběr sloves byla jejich vysoká frekvence v Českém národním korpusu a následně byly přidávány jejich vidové protějšky.

Slovníková hesla byla zpracována manuálně s přihlédnutím ke korpusovému materiálu a již existujícím slovníkům. Důraz byl kladen na přesnost, konzistenci a lingvistickou adekvátnost popisu valence.

VALLEX se snaží být slovníkem pro lidi i pro automatické využití, proto je dostupný nejen jako kniha, ale i přes webové rozhraní, které umožňuje vyhledávat slovesa podle různých kritérií (lemmatu, syntakticko-sémantické třídy, atd.). Pro strojové zpracování a využití v dalších lingvistických aplikacích je VALLEX dostupný také ve formátu XML.

2.3.2 VerbaLex

Jak uvádí Hlaváčková ve své dizertační práci [8], VerbaLex je databáze valenčních rámců českých sloves. Slovník navazuje na tři základní zdroje:

1. Slovník povrchových rámců BRIEF (FI MU)
2. Valenční slovník českých sloves VALLEX 1.0 (MFF UK)
3. soubor valenčních rámců zapsaných v české sémantické síti WordNet

Výsledný slovník VerbaLex obsahuje všechny informace z těchto tří zdrojů, navíc jsou doplněny relevantní informace o slovesech jako je slovesný vid, synonymie sloves, způsob užití slovesa a sémantické třídy sloves, založené na projektu VerbNet.

Základní a komplexní valenční rámce jsou zapsány v podobě datových struktur s realizací v morfologické, syntaktické a sémantické rovině.

věnovat^{impf}

1 ≈ **dát; darovat; připsat**

-frame: $\mathbf{ACT}_{1}^{obl}$ $\mathbf{ADDR}_{3}^{obl}$ $\mathbf{PAT}_{4}^{obl}$ $\mathrm{CAUS}_{k+3}^{typ}$ $\mathrm{AIM}_{k+3,na+4}^{typ}$ $\mathrm{RCMP}_{za+4}^{typ}$

-example: věnoval Martě krásný servis k potěšení (ale: k Vánocům.CAUS); věnoval peníze na dobročinné účely; věnoval své ženě pět krásných koncertů

-rfl: cor3: věnoval si k výročí dar
pass: peníze se věnovaly nadaci

-rcp: ACT-ADDR:

-class: exchange

2 ≈ **obětovat; zasvětit**

-frame: $\mathbf{ACT}_{1}^{obl}$ $\mathbf{ADDR}_{3}^{obl}$ $\mathbf{PAT}_{4}^{obl}$

-example: věnoval dítěti péči / celý život

-rfl: cor3: nevěnoval si péči
pass: dětem se věnuje náležitá péče

-rcp: ACT-ADDR:

Obrázek 2.5: Slovníkové heslo *věnovat* ve valenčním slovníku VALLEX 2.5

Pro záznam sémantické povahy slovesných doplnění byl navržen nový inventář dvojúrovňových sémantických rolí. Použitý seznam sémantických tříd umožňuje podrobnou klasifikaci slovesných významů. VerbaLex je uložen v několika formátech, které umožňují jeho prohlížení, editaci a další využití v oblasti jazykovědy a počítačového zpracování přirozeného jazyka.

Databáze VerbaLex je vytvářena v Centru zpracování přirozeného jazyka na Fakultě informatiky Masarykovy Univerzity v Brně.

Motivací k budování VerbaLexu je vytvořit rozsáhlou databázi zachycující slovesná doplnění v přirozeném kontextu. VerbaLex může být používán nejen v oblasti lingvistiky, ale i pro strojové zpracování češtiny.

Kapitola 3

Současný stav projektu PDEV

3.1 Platforma PDEV

Technicky je projekt PDEV podporován Fakultou Informatiky Masarykovy Univerzity v Brně [14]. Projekt je z velké části veřejně přístupný. Uživatel může k datům přistupovat přímo přes webové rozhraní nebo pomocí instalace rozšíření do aplikace Mozilla Firefox (tzn. add-on). Dále se budeme zabývat pouze druhým způsobem, který je pohodlnější a patterny lze kromě prohlížení i editovat a vytvářet.

Po spuštění se otevřou dvě okna – PDEV Entry Manager a Sketch Engine. V okně Sketch Engine lze vyhledávat slovesa a jejich konkordance, prohlížet Word Sketches, Thezaurus atd.

V Entry Manager (viz obrázek 3.1) lze prohlížet slovesa s jejich statistikami, jako je např. frekvence v BNC, BNC50, OEC (Oxford English Corpus), počet patternů atd. Nezaregistrovanému uživateli se zobrazí pouze zkompilovaná slovesa.

Poklepáním na sloveso se otevře nové okno (viz obrázek 3.2) s patterny vybraného slovesa (v horní části okna). Rozkliknutím patternu se zobrazí další detaily v dolní části okna. V tomto dialogu lze patterny editovat. Číslo v procentech uvedené na řádce s definicí patternu říká, jak velké části všech označkovaných konkordancí byl přiřazen právě tento pattern. Pomocí tlačítka *Corpora* umístěného v záhlaví okna můžeme tyto konkordance zobrazit.

Databáze PDEV se skládá ze tří částí:

- definice patternů
- ručně označkovaný náhodně vybraný referenční vzorek ke každému slovesu
- hierarchie sémantických typů (v PDEV nazýváno *shallow semantic ontology*)

3.1.1 Definice patternů ve formátu XML

Definice patternů jsme získali ve formátu XML. Bohužel zatím neexistuje žádná dokumentace ani XML schéma. Pro usnadnění práce jsme pomocí programu

PDEV Entry Manager

File Entry

d | all | Filter | Types | Ontology | Difference | Print | Export

Entry	N...	OEC freq	BNC freq	BNC50 freq	Created	Created by	Last edited	Edited by	Status
dab	4	1163	216	18	2009-06-01	patrick	2009-10-20	patrick	complete
dabble	2	1822	182	82	2009-06-01	patrick	2009-06-29	patrick	complete
dash	5	8166	910	300	2007-02-27	patrick	2009-02-18	patrick	complete
debate	3	15383	1321	912	2007-01-18	cpa04	2007-06-12	ypokor	complete
declassify	1	642	7	6	2008-05-21	ipajerova	2008-11-19	patrick	complete
deify	2	268	25	20	2008-07-11	ipajerova	2009-12-17	patrick	complete
delve	3	3902	238	117	2007-01-19	cpa06	2009-02-18	patrick	complete
demystify	1	585	50	39	2008-07-18	ipajerova	2008-10-14	patrick	complete
denigrate	1	1608	133	92	2006-12-21	patrick	2009-10-30	patrick	complete
deny	9	66591	7509	4811	2007-02-24	patrick	2010-01-27	patrick	complete
deride	1	2166	132	86	2006-12-20	patrick	2009-10-30	patrick	complete
deter	2	8007	882	601	2009-04-25	patrick	2009-11-15	patrick	complete
detoxify	4	420	32	23	2008-07-11	ipajerova	2009-11-09	patrick	complete
devote	6	23117	2082	1530	2006-12-15	patrick	2010-02-20	patrick	complete
devour	4	4398	370	152	2006-11-22	cpa01	2009-02-10	patrick	complete
digest	3	4260	604	261	2006-11-20	patrick	2009-10-28	patrick	complete
dignify	1	1783	26	14	2008-07-11	ipajerova	2008-11-17	patrick	complete
dine	3	6051	562	189	2006-11-17	patrick	2008-10-20	patrick	complete
disparage	1	1336	52	38	2006-12-21	patrick	2009-10-31	patrick	complete
dispense	5	5254	763	455	2007-01-08	cpa07	2009-11-02	patrick	complete
disqualify	3	3771	443	227	2009-06-01	patrick	2009-11-10	patrick	complete
dog	2	4828	217	115	2007-04-05	patrick	2008-12-01	patrick	complete
doze	2	1950	259	45	2007-03-03	patrick	2009-11-04	patrick	complete
drain	15	12536	1596	500	2007-02-08	cpa01	2009-12-18	patrick	complete
drink	5	80582	7086	1844	2006-11-14	patrick	2009-12-28	patrick	complete
dust	5	5474	559	131	2008-06-30	al12	2009-02-16	patrick	complete

Filtered verbs: 450 | Total verbs: 5756, patterns: 9424 | Completed verbs: 690, patterns: 2639 | Draft verbs: 13

Obrázek 3.1: PDEV Entry Manager – seznam sloves

devote; CPA Patterns

Patterns for: devote | Add | Copy | Corpora | Preview | Renumber | Delete | Close

Save | Sample size 250 | **250** Semantic class | Erlangen No

1 46% **[[Human | Institution]] devote [[Resource]] {to [[Activity]]}**
[[Human | Institution]] expend [[Resource]] on [[Activity]]

2 18% **[[{Human = Author} | {Institution = Newspaper | Journal}]] devote [[Document Part | Document]] {to [[Anything = Topic]]}**
[[{Human = Author} | {Institution = Newspaper | Journal} | Document]] gives priority to discussion or elaboration of [[Anything = Topic]] in [[Document Part | Document]]

3 10% **[[Human]] devote [[Activity 1 = Thought]] {to [[Anything]]}**
The mind or attention of [[Human]] is focused on [[Anything]]

4 5% **[[Human | Institution]] devote [[Activity]] {to [[Anything]]}**
The main [[Activity]] of [[Human | Institution]] is concerned with [[Anything]]

5 5% **[[Human | Institution]] devote [[Building | Building Part | Container]] {to [[Anything]]}**
The main contents of [[Building | Building Part | Container]] are [[Anything]]}

6 11% **[[Human]] devote [[Self]] {to [[Activity]]}**
[[Human]] spends time on [[Activity]]

devote Pattern 2 | show: | Save | Save & close | Close | Test

FrameNet NOT YET IN. 12 January 2010. | Open FrameNet

subject Human | Role Author | Lexset | Attr.
subj. alt. + - Institution | Role Newspaper | Journal | Lexset | Attr.
verb form devote
object Document Part | Role | Lexset page| chapter| space | Attr.
no object | optional
obj. alt. - Document | Role | Lexset | Attr.
+ optional
adverbial - Prep|Particle to | Anything | Role Topic | Lexset | Attr.
no adverbial - + opt. + optional Adverb(s)
primary implicature [[{Human = Author} | {Institution = Newspaper | Journal} | Document]] gives priority to discussion or elaboration of [[Anything = Topic]] in [[Document Part | Document]] | idiom | pv | +

Count: 6

Obrázek 3.2: PDEV Entry Manager – patterny slovesa *devote*

`trang` [15] automaticky vygenerovali DTD schéma. Tento krok velmi usnadnil další práci s XML souborem. S definicí patternu pracujeme na dvou místech:

- při přiřazování možných sémantických typů slovesným argumentům získaných na základě syntaktické analýzy a
- při automatickém přiřazování patternů

Z tohoto důvodu byl vytvořen vlastní zjednodušený formát reprezentace patternů. Všechny relevantní informace o patternu jsou zapsány na jednom řádku v CSV formátu.

Při strojovém zpracování je zbytečné pracovat s elementy či atributy, které jsou vyplněné jen zřídka a jejich význam (díky chybějící dokumentaci) nebyl zcela jasný.

V příloze B je na obrázku B.1 uvedena část XML souboru, která obsahuje definici druhého patternu slovesa devote. Program pro konverzi XML formátu nejdříve vytvoří vnitřní strukturu (viz dump na obrázku B.2) a následně zapíše informace oddělené pomocí středníků do jedné řádky (viz obrázek B.3), která je součástí CSV souboru.

Převod probíhá ve scriptu `convert_xml_into_csv.pl`. Program nejprve načte celý XML soubor a postupně zpracovává slovesa a jejich patterny. Definice jednoho patternu je načtena do vnitřní struktury, která je tvořena pomocí polí a asociativních polí (hashů). Z této struktury se následně vytiskne řádka do výstupního souboru pomocí procedury `intern_to_csv_line()` implementovanou v modulu `Patterns.pm`.

Problémem je, že nikde nedochází ke kontrole dat vkládaných pomocí webového formuláře – tedy ani při zpracování formuláře, ani při ukládání do databáze. Tento fakt způsobuje, že do polí ve formuláři může být napsáno prakticky cokoliv. Při zpracování XML bylo potřeba vyrovnat se s tím, že obsah některých elementů a atributů neodpovídá tomu, co by se zde dalo očekávat.

Např. nelze očekávat, že obsahem atributu *name* elementu *BSO_type* bude vždy název sémantického typu popř. doplněného číslem (pokud je sémantický typ v propozici použit na více kolokačních pozicích). V několika případech se stalo, že obsahem tohoto pole byla zároveň sémantická role (např. Human 2 = Monarch v patternu č. 1 slovesa abdicate).

Tato drobná nekonzistence zřejmě vznikla nepozorností při psaní definice patternu. Samozřejmě by nebylo příliš složité tyto případy rozpoznat a automaticky oddělit sémantický typ od sémantické role. Jenže pokud bychom měli přidávat nová a nová pravidla, zpracování XML by se značně znepřehlednilo, k opravám by mělo dojít raději v databázi patternů.

Některé nekonzistence by ani nemohly být řešeny automaticky. Příkladem toho je opět sloveso *abdicate*, konkrétně pattern č. 2. V popisu objektu očekáváme v poli *Lexset* výčet lexikálních jednotek. Zde se nachází řetězec „role | responsibility (for [[Anything]])“, který definuje další rozvití objektu. Takové případy nelze snadno upravit rozdělením řetězce do dvou polí, protože formulář ani neumožňuje popsat další rozvití slovesných argumentů.

Na základě těchto pozorování jsme se rozhodli, že patterny, které obsahují tyto nekonzistence, nebudeme při další práci zpracovávat, dokud nebudou

pokrytí (%)	počet sloves	bnc50 freq	pokrytí (%)	počet sloves	bnc50 freq
10	7	54 872	90	917	610
15	12	39 780	95	1 518	246
20	19	25 894	96	1 736	187
25	30	20 367	97	2 030	136
30	42	18 253	98	2 451	90
40	74	12 028	99	3 148	48
50	120	8 723	99,5	3 775	28
60	185	5 889	99,75	4 303	17
70	291	3 489	99,9	4 834	10
80	473	1 844	100	5 781	1

Tabulka 3.1: Pokrytí korpusu BNC50 – slovesa jsou seřazena podle jejich frekvence, např. k 50 % pokrytí korpusu (slovesných tokenů) by bylo potřeba zkompilovat 120 nejfrekventovanějších sloves.

opraveny. Týká se to cca 150 patternů, které pokrývají cca 100 sloves. Tyto „nedokonalé“ patterny mají jako první hodnotu nulu v CSV souboru. Detailní popis polí v CSV řádce lze nalézt v příloze B, kde je také ukázka zápisu definic patternů několika vybraných sloves v CSV formátu.

3.2 Statistiky sloves

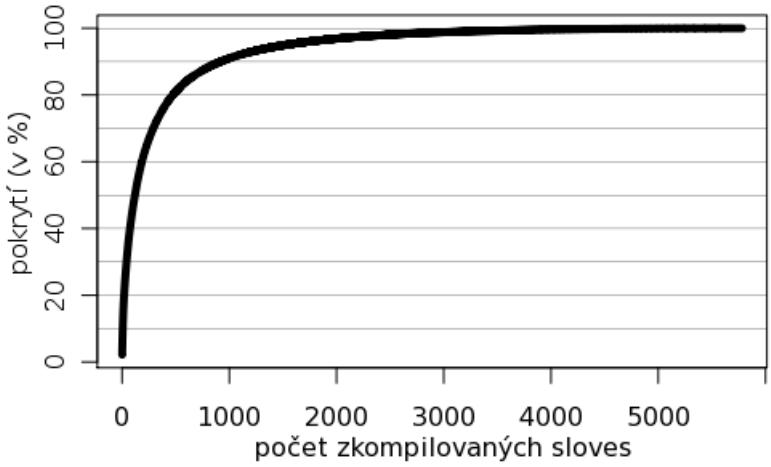
PDEV obsahuje celkem 5 794 sloves s celkem 7 944 682 výskyty[1]. Po odečtení pomocných (*be, do, have, will*) a modálních (*can, could, may, might, must, ought, shall, should, would*) sloves, kterých je celkem 13, nám zbývá 5 781 sloves, ale jen 4 673 093 slovesných výskytů. Pomocná a modální slovesa pokrývají 41,2 % korpusu BNC50. Dále se budeme zabývat pouze lexikálními slovesy, tedy níže uvedené statistiky nebudou již uvažovat pomocná a modální slovesa.

Celkový počet zkompilovaných (tzn. mají status *complete*) sloves ke dni 5. 3. 2010 je 678, tedy 11,7 % všech lexikálních sloves, a jejich výskyty pokrývají celkem 10,6 % korpusu (495 724 slovesných tokenů).

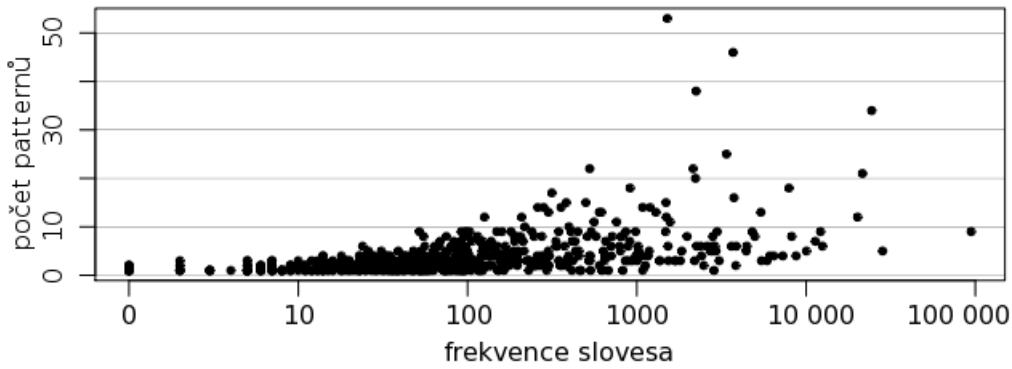
V tabulce 3.1 je uvedeno, kolik nejfrekventovanějších sloves je třeba zkompilovat k dosažení pokrytí korpusu. Např. 7 sloves s největšími počty výskytů ($\geq$ 54 872) pokrývají 10 % korpusu. Pokud bychom chtěli mít pokrytých alespoň 90 % slovesných tokenů, bylo by potřeba zkompilovat celkem 917 sloves, která mají frekvenci vyšší než 610. Z těchto sloves je zpracováno pouze 97, tedy zbylých 581 sloves mají frekvenci menší než 610. Poslední 1 % pokrytí korpusu tvoří slova s frekvencí $\leq$ 48, kterých je v PDEV zpracováno 260, tedy více než jedna třetina.

Celkový počet patternů je 2 572, průměrný počet patternů na sloveso je 3,79, ale očekávaný počet patternů na sloveso je 9,72 (slovesa s větším počtem výskytů mají více patternů). Korelace mezi frekvencí slovesa a počtem jeho patternů je na obrázku 3.4.

[1] Všechny uvedené statistiky se vztahují ke dni 5. 3. 2010



Obrázek 3.3: Pokrytí korpusu BNC50 – na ose x je počet zkompilovaných nejfrekventovanějších sloves, na ose y pokrytí slovesných tokenů



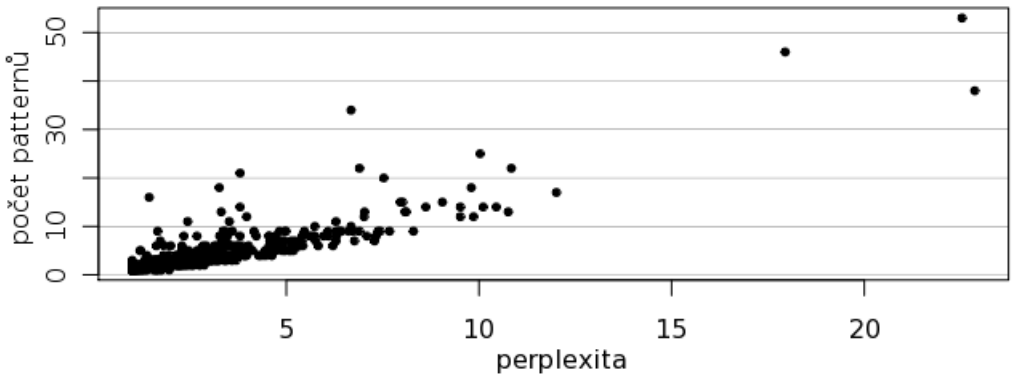
Obrázek 3.4: Korelace mezi frekvencí slovesa a počtem jeho patternů

Na základě označkovaného referenčního vzorku jsme spočítali entropii a perplexitu pro patterny každého slovesa. Závislost mezi perplexitou a počtem patternů slovesa je na obrázku 3.5.

3.3 Mezianotátorská shoda

Slovník PDEV by hypoteticky mohl být používán v mnoha NLP aplikacích, např. při strojovém překladu. Předtím je však třeba ověřit, zda i jiní lidé než sám autor dokáží definice patternů sloves nejen *pochopit*, ale také se *shodnout* na přiřazení patternů. Dosud vytvářel patterny a značkoval konkordance převážně Patrick Hanks a neproběhla žádná kontrola, která by prokázala využitelnost slovníku.

K tomuto účelu jsme navrhli experiment, který změří mezianotátorskou



Obrázek 3.5: Korelace mezi perplexitou slovesa a počtem jeho patternů

shodu [16]. Rozumný výsledek je základem dalšího automatického zpracování – pokud chceme naučit počítač, aby dobře značkoval patterny, musíme nejprve ověřit, zda je tato úloha zpracovatelná člověkem.

Bylo vybráno celkem 30 sloves, která jsou zkompilována v PDEV. 20 z nich bylo anotováno v BNC50 a 18 sloves bylo anotováno v PEDT.

Slovesa anotovaná v BNC50 byla vybrána tak, aby se co nejvíce přiblížila reprezentativnímu vzorku, tedy aby byly pokryty všechny frekvenční hladiny a počet patternů jednotlivých sloves byl pestrý.

Pro každé sloveso byl v BNC50 vybrán náhodný vzorek o velikosti 50 vět, který označkovali 2–4 anotátoři. Kromě čísla patternu mohl anotátor použít značky u a x, popř. číslo paternu doplnit písmenem e (viz kap. 2.1.2).

Slovesa anotovaná v PEDT byla vybrána pomocí jiných kritérií, protože výsledná data měla posloužit jako podklad pro výzkum Jana Popelky [17]. Důraz byl kladen především na překladovou víceznačnost slovesa a také na dostatečnou frekvenci v PEDT.

Při anotaci patternů v PEDT byla možnost rozlišovat kromě normálního užití patternů další dva způsoby. Místo doplňující značky e se používaly značky a (nepřesná shoda patternu, např. chybějící argument nebo neodpovídající sémantický typ) a f (figurativní užití).

Evaluační míry

Při měření mezianotátorské shody na datech PEDT jsme používali Cohenovo kappa, protože všechna slovesa anotovali pouze dva anotátoři (SC – Silvie Cinková a PH – Patrick Hanks). V BNC50 používáme Fleissovo kappa, protože zde byl počet anotátorů 2–4.

Průběh evaluace

Před samotnou evaluací bylo potřeba připravit označkovaná data do vhodného formátu pro následné měření mezianotátorské shody. K tomuto účelu použili programy `prepare_annot_data_bnc.pl`, `prepare_annot_data_bnc_hanks.pl` a `prepare_annot_data_pedt.pl`. K následnému spočítání mezianotátorské shody

PH	SC	1	1.e	2	3	3.e	5	u	x	#	p
1		30	–	1	–	–	–	–	–	31	0.62
1.e		1	–	–	–	–	–	–	–	1	0.02
2		–	–	3	–	–	–	–	–	3	0.06
3		–	–	–	4	–	3	1	–	8	0.16
3.e		–	–	–	–	–	2	1	–	3	0.06
5		–	–	–	–	–	1	–	–	1	0.02
u		–	–	2	–	–	–	–	–	2	0.04
x		–	–	–	–	–	–	–	1	1	0.02
#		31	0	6	4	0	6	2	1	50	
p		0.62	0.00	0.12	0.08	0.00	0.12	0.04	0.02		1.00

Tabulka 3.2: Shody a neshody anotátorské dvojice PH a SC na slovesu *claim* v BNC50

jsme použili program `iaa.pl`, jehož výstupem jsou nejen samotné výsledky měření, ale také podrobné podklady pro následnou analýzu neshod.

Tyto podklady tvoří:

1. Tabulky zachycující počty jednotlivých shod (na diagonále) a neshod (mimo diagonálu) pro každé sloveso a každou anotátorskou dvojici. Tyto hodnoty pro sloveso *claim* označkované v BNC50 Silvií Cinkovou a Patrickem Hanksem jsou uvedeny v tabulce 3.2. V tabulce je také uvedeno, kolikrát anotátoři označili jednotlivé patterny (sloupec resp. řádka „#“) a jaké procento tento pattern tvoří v jejich anotaci daného slovesa (sloupec resp. řádka p)

2. Konkordance seřazené vzestupně podle počtu dvojic, které se shodli na značce (nejvýše jsou ty, kde byla nejmenší shoda). Ukázka je v tabulce 3.3. První sloupec udává počet shodujících se dvojic, další sloupce obsahují konkrétní značky anotátorů, jejichž iniciály jsou uvedeny záhlaví sloupců.

3. Statistika typů neshod pro každou dvojici, která je uvedena u jednotlivých sloves a nakonec souhrnně pro všechna slovesa označkovaná danou anotátorskou dvojicí. Ukázka se nachází v tabulce 3.6.

Uvedené programy a data použitá pro měření mezianotátorské shody včetně uvedených výstupů lze nalézt na přiloženém CD.

Výsledky měření mezianotátorské shody

Výsledky měření mezianotátorské shody jsou uvedeny v tabulkách 3.4 a 3.5 včetně statistických údajů o vybraných slovesech.

Popis sloupců:

- N ... počet patternů
- P ... perplexita výskytu patternů
- f ... frekvence slovesa v korpusu BNC50

#	JS	LS	PH	SC	konkordance
0	5.e	1	3.e	u	Over Malta on 7th. little had been seen , although the A.A....
0	5.e	3	3.e	5	That is why I came here : to <claim> kin with him , to be...
1	1.e	2	u	2	Cutting out surplus letters reduces the effort of reading and...
1	5.e	3	3	u	They <claim> 10,000 members , but if campaign finance forms...
1	1.e	2	u	2	Although the choice of task may , according to this notion ,...
2	5	3	3	5	There is a clear distinction here which is not always...
2	5	3	3	5	All these studies acknowledge that Mills and Boon is a...
3	1	1	1	2	The plaintiffs <claimed> that certain grass verges were part...
3	5	3	3	3	Yet another complicating factor is that the account given...
3	5	5	3.e	5	Any success that could be <claimed> to the credit of...
3	1	1	1.e	1	During the conversation , the Prince and Camilla tell each...
3	5	5	3	5	How can I <claim> any loyalty to separatism after what has...
3	3.e	3	3	3	Women have to <claim> their right to be angry , but men have...
6	1	1	1	1	He <claimed> she had instigated the intercourse by first ,...
6	1	1	1	1	In the Williams incident , a caller giving a false name and...
6	1	1	1	1	Although Douglas was eventually acquitted of any offence ,...
6	1	1	1	1	Wolfgang Luder , MP with the liberal Free Democrats , whose...
6	1	1	1	1	They <claim> that the prohibition in fact merely equalizes...
6	1	1	1	1	According to a report in NIN (9 November 1986) , the...
6	1	1	1	1	The programme <claimed> that there were 100 artistes in...

Tabulka 3.3: Přehled neshod u slovesa *claim* v BNC50, první sloupec označuje počet dvojic, které se shodly na přiřazení patternu

- A ... počet anotátorů (pouze v BNC50, v PEDT byli vždy 2)
- size ... velikost anotovaného vzorku (pouze v PEDT, v BNC50 je velikost vždy 50)
- +exp ... P a P.e se *považovalo* za chybu (P je číslo patternu)
- −exp ... P a P.e se *nepovažovalo* za chybu

V BNC50 jsme měli možnost také porovnat Hanksovy značky s jeho dřívější anotací. Dalo by se očekávat, že hodnoty budou blízké 1, ale ve skutečnosti se příliš neliší od výsledků shody ostatních anotátorů.

Ke každé anotátorské dvojici máme k dispozici statistiku typů neshod. V tabulce 3.6 je v prvním sloupci uveden typ neshody, ve druhém četnost typu neshody pro PH se svou dřívější anotací a ve třetím počet neshod daného typu mezi PH a SC.

U PH došlo k neshodě ve 155 případech. Z toho 46 je typu *P vs. P.e* (normální užití patternu versus jeho využití – exploitation). Neshody tohoto typu můžeme při výpočtu případně ignorovat (sloupec PH −*exp* v tabulce 3.4). Nejčastější neshodou bylo označení konkordance jako normálního užití různých patternů.

Pozorování vypovídá o tom, že ani sám tvůrce patternů není s odstupem času konzistentní.

	sloveso	N	P	f	A	Fleiss' κ			
						+exp	−exp	PH +exp	PH −exp
1	abstain	3	2,47	232	4	0,72	0,82	0,77	0,84
2	accept	8	4,16	12 190	3	0,54	0,64	0,65	0,80
3	address	6	3,85	3 628	3	0,37	0,49	0,71	0,78
4	admit	13	7,03	5 410	3	0,67	0,74	0,61	0,73
5	alter	2	1,28	2 489	3	0,59	0,78	0,74	0,74
6	announce	4	2,68	8 739	3	0,68	0,76	0,80	0,86
7	argue	7	1,73	11 362	3	0,83	1,00	0,74	1,00
8	call	34	6,68	24 439	2	0,69	0,68	0,87	0,90
9	claim	6	3,14	12 517	4	0,69	0,72	0,72	0,82
10	engage	9	4,82	2 985	2	0,58	0,66	0,76	0,84
11	explain	5	3,21	10 064	3	0,66	0,74	0,71	0,73
12	fire	15	7,96	1 488	2	0,34	0,42	0,59	0,72
13	lead	12	3,97	20 180	3	0,72	0,78	0,74	0,81
14	need	5	4,12	28 352	3	0,64	0,66	0,84	0,83
15	plan	4	3,08	7 294	3	0,57	0,60	0,78	0,86
16	rush	9	4,99	984	2	0,67	0,65	0,80	0,78
17	say	9	1,66	94 608	3	0,48	0,48	0,46	0,46
18	spoil	9	3,36	409	3	0,65	0,73	0,85	0,93
19	tell	21	3,80	21 550	2	0,27	0,27	0,74	0,84
20	visit	3	2,16	5 889	3	0,56	0,60	0,80	0,83

Tabulka 3.4: Výsledky měření mezianotátorské shody - BNC50

	sloveso	N	P	f	size	Cohen's κ	
						+exp	−exp
1	abandon	8	4,22	2 813	34	0,64	0,72
2	acknowledge	5	3,07	2 731	34	0,74	0,91
3	admit	13	7,03	5 410	48	0,56	0,68
4	anticipate	3	2,27	1 373	41	0,49	0,90
5	argue	7	1,73	11 362	92	0,81	0,93
6	call	34	6,68	24 439	100	0,65	0,72
7	claim	6	3,14	12 517	71	0,77	0,87
8	deny	9	6,09	4 811	63	0,85	0,88
9	execute	5	4,70	1 183	33	0,70	0,80
10	fire	15	7,96	1 488	26	0,65	0,71
11	handle	6	3,35	2 796	56	0,53	0,63
12	launch	6	3,19	3 874	65	0,82	0,94
13	lead	12	3,97	20 180	100	0,68	0,83
14	say	9	1,66	94 608	100	0,39	0,39
15	signal	6	3,76	892	39	0,66	0,80
16	tell	21	3,80	21 550	100	0,64	0,66
17	treat	4	2,60	6 455	31	1,00	1,00
18	urge	6	2,84	2 603	42	1,00	1,00

Tabulka 3.5: Výsledky měření mezianotátorské shody - PEDT

typ neshody	PH	PH vs. SC
P1 vs. P2	49	117
P1 vs. P2.e	14	20
P1.e vs. P2.e	2	3
P vs. P.e	46	34
u vs. P	27	35
u vs. P.e	10	15
x vs. any	7	17
celkem neshod	155	241
velikost vzorku	1000	1000

Tabulka 3.6: Statistika typů neshod, sloupec „PH“ označuje počty typů neshod Hankse se svou dřívější anotací a sloupec „PH vs. SC“ neshody mezi Hanksem a Cinkovou

3.3.1 Diskuze

Odhalili jsme následující typy neshod:

1. *Nejasné instrukce ohledně kontextu.* V teoretickém základu metody CPA není určeno, jak široký kontext se má brát v úvahu při značkování konkordancí.

2. *Označitelné – neoznačitelné.* V některých případech je obtížné rozhodnout se, zda se má slovo vůbec považovat za sloveso a lze mu přiřadit pattern (např. participium).

3. *Elipsy.* Elipsy (vypuštění argumentu) jsou problematické, protože díky nim dochází k víceznačnosti. Klasifikovali jsme dva typy víceznačností: a) na základě kontextu nelze rozhodnout mezi dvěma potenciálními relevantními patterny, z nichž první pattern daný argument vůbec neuvádí a druhý pattern povoluje vypuštění tohoto argumentu b) dva patterny s odlišnými implikaturami, které dovolují vypuštění argumentu. V těchto případech nelze jednoznačně říci, jak by argument vypadal po rekonstrukci elipsy. Význam by mohl být určen pouze na základě širšího kontextu, který je již nad rámec CPA.

4. *Argument odpovídá více sémantickým typům.* V několika málo případech se stalo, že na základě kontextu mohlo být přiřazeno více patternů, protože argument odpovídal více sémantickým typům v různých patternech mezi kterými nebylo možné jednoznačně se rozhodnout.

5. *Nedostatečná znalost angličtiny.* Nerodilí mluvčí občas špatně porozuměli větě.

6. *Chybějící pattern.* Většina vět anotovaných v BNC50 byla pokryta již existujícími patterny. Pro některé věty anotované v PEDT, který je považován za doménově omezený korpus, neexistoval vhodný pattern v PDEV. Anotátoři pak tento výskyt označkovali jako využití jiného patternu a

neshoda nastala právě u výběru tohoto patternu. Na základě těchto vět vznikl návrh na vytvoření nového patternu.

7. *Příliš jemné rozlišení implikatur.* V náhodně vybraném vzorku se ukázalo, že v některých případech nelze rozhodnout mezi implikaturami, které přísluší různým patternům, protože jejich rozdíly mezi nimi jsou velmi jemné.

8. *Příliš jemné rozlišení mezi sémantickými typy.* Celkem často se stalo, že konkordance neodpovídala patternu, protože argument nesouhlasil se sémantickým typem uvedeným na dané kolokační pozici, přestože intuitivně pattern odpovídal dané konkordanci.

Přestože výsledky měření mezianotátorské shody nejsou příliš vysoké, nebylo příliš mnoho neshod způsobeno nedostatky v patternech.

Anotace zatím není zcela rutinní záležitostí. Několik chyb bylo způsobeno pouhým přehlédnutím se. Např. si anotátor spletl čísla patternů a v celém jednom vzorku je tak zaměňoval. Dalším častým problémem bylo, že si anotátor neuvědomil, že stejná implikatura je rozdělena do dvou patternů podle povrchové realizace – v jednom patternu je argument realizován sémantickým typem a ve druhém slovesnou klauzí. Anotátor pak stále přiřazoval pouze první pattern.

Nejčastější nedostatky v patternech, se kterými jsme se setkali při analýze mezianotátorské shody, jsou snadno odstranitelné přidáním dalšího sémantického typu na kolokační pozici v definici patternu.

Konkordance, kde nesouhlasí sémantický typ argumentu se sémantickým typem uvedeným v patternu, ale přesto jsou intuitivně chápány jako „normy“, budeme striktně označovat jako nerozhodnutelné (u). Toto se stávalo celkem často, pravděpodobně je to tím, že zpracování některých anotovaných sloves bylo dokončeno v době, kdy množina sémantických typů ještě neměla dnešní podobu.

Pozitivním zjištěním je, že chybějící nebo překrývající se patterny byly spíše vzácností.

Díky tomuto pilotnímu experimentu jsme odhalili možné problémy, na které je třeba brát ohled při sepisování instrukcí pro anotátory. Neshody způsobené body 1, 2, 3 a 5 chceme omezit vytvořením kvalitního manuálu a najmutí rodilých mluvčích pro anotaci.

3.4 Další vývoj

V současné době vzniká podrobný manuál pro vytváření patternů společně s manuálem pro anotátory konkordancí.

Protože anotace je hodnotnou zpětnou vazbu pro vytváření patternů, byla navržena tato validační procedura:

1. Hanks bude vytvářet definice patternů jako dosud.

2. V momentě, kdy sloveso prohlásí za zkompilované a připravené k validaci, předloží se patterny anotátorům včetně ručně označkovaného referenčního vzorku.

3. Anotátoři označkují nově vybraný náhodný vzorek z BNC a zapíší si poznámky k případným chybějícím patternům, nesrozumitelnému kontextu atd.

4. Změří se mezianotátorská shoda a následně se analyzují neshody, které budou rozebrány s Hanksem.

5. Na základě analýzy neshod se zrevidují definice patternů a/nebo se zdokonalí instrukce pro anotátory.

6. Zrevidované patterny budou znovu předloženy anotátorům s novým náhodně vybraným vzorkem k anotaci.

7. Celý proces se bude opakovat, dokud mezianotátorská shoda nebude přijatelná (alespoň v nejdůležitějších bodech, jako je 4, 6 a 7, vyjmenovaných v kapitole 3.3.1).

8. Každé takto zrevidované sloveso bude označeno jako *validated* a připraveno pro experimenty strojového učení.

Další plánované změny jsou spíše technického charakteru a mají za cíl umožnit konzistentní kódování patternů v PDEV. Tyto změny se týkají jak webového formuláře pro editaci patternů, tak jejich následného uchování v XML souboru. Kromě XML schématu má vzniknout také podrobná technická dokumentace popisující jednotlivé elementy a atributy včetně jejich přípustných hodnot. Upravený webový formulář by především měl poskytnout možnost zápisu všech relevantních informací do příslušných polí, což umožní efektivnější strojové zpracování následně vygenerované XML struktury.

V závěru této kapitoly můžeme říci, že po dosažení plánovaných cílů může být PDEV využíván v NLP aplikacích. To dokazuje např. práce Jana Popelky *O klasifikaci anglických sloves dle PDEV a české překladové ekvivalenci* [17], ve které se autor zabývá otázkou, zda znalost patternu slovesa v anglické větě pomůže zjednoznačnit výběr českého překladového ekvivalentu.

Kapitola 4

Závislostní mikrokontext slovesa

Cílem této části je automaticky extrahovat z každé věty argumenty slovesa. Nejprve jsme provedli automatickou závislostní syntaktickou analýzu a nalezneme větné členy závisející na slovesu.

K získání hloubkových závislostí jsme použili Stanfordský parser, který kromě složkových stromů umí vydat též semisémantické závislosti.

Při evaluaci automatických metod jsme se zaměřili pouze na agenty a objekty slovesa. Agentem (značíme *agent*) zde rozumíme logický (hloubkový) subjekt, v PEDT označováno jako ACT (actor). Objekty rozdělujeme na přímý objekt (značíme *object*) a nepřímý objekt (značíme *iobject*). Zde není přesná ekvivalence s PEDT, ale většinou přímému objektu odpovídá PAT (patient) v PEDT a nepřímému objektu ADDR (addressee).

Pro úlohu evaluace jsme sjednotili přímé a nepříme objekty a značíme je jednotně jako *object*.

Neuvažujeme agenty a objekty, které jsou realizovány vedlejší větou. Další extrahované argumenty již přímo neodpovídají hloubkové realizaci, ale spíše povrchové.

Jako termín pro označení názvu závislosti, se kterým budeme dále pracovat, používáme pojem *funkce argumentu*.

Porovnali jsme metody založené na složkovém parsingu doplněném převodem na semisémantické závislosti (Stanfordský parser, Charniak-Johnsonův parser + převod na stanfordské závislosti), závislostní parser (McDonald) a metodu založenou na srovnávání textu s regulárními výrazy, která je použita v aplikaci Sketch Engine [18].

4.1 Stanfordský parser

Stanfordský parser je složkovým parserem, který byl později doplněn o převod na závislosti, které poskytují větší sémantickou informaci věty. Tyto závislosti nazýváme *stanfordské závislosti* (z angl. Stanford dependencies) [19, 20], popř. značíme zkratkou STD.

Kromě několika typů stanfordských závislostí lze získat i původní složkové stromy. Naopak lze složkové stromy vygenerované jiným parserem převést na stanfordské závislosti.

Kompletní manuál včetně vysvětlení stanfordských závislostí a jejich typů lze nalézt v [21], další užitečné informace jsou na stránce Stanfordského parseru v sekci FAQ [22].

Všechny (celkem 4) typy stanfordských závislostí mají společný formát – jedna věta je posloupností řádek, z nichž každá reprezentuje hranu v závislostním stromu, resp. orientovaném grafu. Jednotlivé věty jsou od sebe odděleny jednou prázdnou řádkou. Zápis hrany v STD má podobu *Rel(Gov, Dep)*, kde *Rel* je zkratka anglického názvu gramatického vztahu mezi *Gov* (z angl. governor) a *Dep* (dependent), což jsou tokeny doplněné jejich pozicí ve větě. Hrana vede vždy od *Gov* k *Dep* a má název *Rel*. Uzly v závislostním grafu reprezentují slova (interpunkce se v závislostním grafu nevyskytuje) a hrany reprezentují gramatické vztahy mezi slovy ve větě.

Jako příklad uveďme větu *He claimed to have links with Gadhafi, and inside knowledge about Libyan and Syrian backing for terrorists.* (zdroj BNC50).

Na výstupu Stanfordského parseru se objeví tato řádka (mimo jiné):

nsubj(claimed-2, He-1)

kde *nsubj* (nominal subject) je *Rel*, *claimed-2* je *Gov* a *He-1 Dep*. Čísla 1 a 2 označují pozici tokenů v dané větě. Celý strom je na obrázku 4.1.

Použitá verze Stanfordského parseru obsahuje celkem hierarchicky uspořádaných 55 základních závislostí. Nejvýše je nejobecnější závislost *dep*, v každé nižší vrstvě je konkrétnější závislost.

Př. hierarchie subjektů:

```
dep - dependent
   arg - argument
      subj - subject
         nsubj - nominal subject
            nsubjpass - passive nominal subject
         csubj - clausal subject
            csubjpass - passive clausal subject
```

Úplný seznam lze najít v manuálu [21] včetně podrobného popisu a příkladů ke každé závislosti. Další závislosti vznikají např. připojením předložky přímo do názvu závislosti v rámci redukovaných (collapsed) závislostí.

Nyní podrobně vysvětlíme typy stanfordských závislostí. Použijeme k tomu větu uvedenou výše (*He claimed to have...*[1]).

Pozn: Použité obrázky jsou vygenerovány pomocí software Graphviz [23]. Na přiloženém CD lze nalézt program pro převod stanfordských závislostí do zdrojového kódu pro Graphviz.

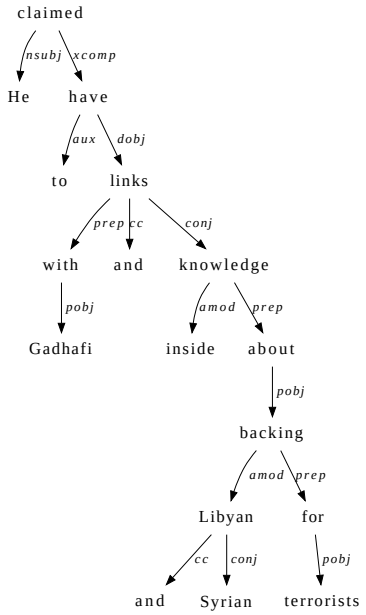
1. Základní

(Basic.) Obsahuje pouze základní závislosti a vytváří stromovou strukturu. Každé slovo obsažené ve větě (s výjimkou interpunkce) je samostatný uzel

[1] Tato věta nebyla zparsována úplně v pořádku, v příkladech jsou použity opravené grafy a závislosti.

v závislostním stromu. Tento typ není příliš praktický, protože např. předložkový objekt nezávisí přímo na slovesu, ale na předložce. Při sémantické analýze musí být uzel reprezentující předložku přeskočen.

nsubj(claimed-2, He-1)
aux(have-4, to-3)
xcomp(claimed-2, have-4)
dobj(have-4, links-5)
prep(links-5, with-6)
pobj(with-6, Gadhafi-7)
cc(links-5, and-9)
amod(knowledge-11, inside-10)
conj(links-5, knowledge-11)
prep(knowledge-11, about-12)
amod(backing-16, Libyan-13)
cc(Libyan-13, and-14)
conj(Libyan-13, Syrian-15)
pobj(about-12, backing-16)
prep(backing-16, for-17)
pobj(for-17, terrorists-18)

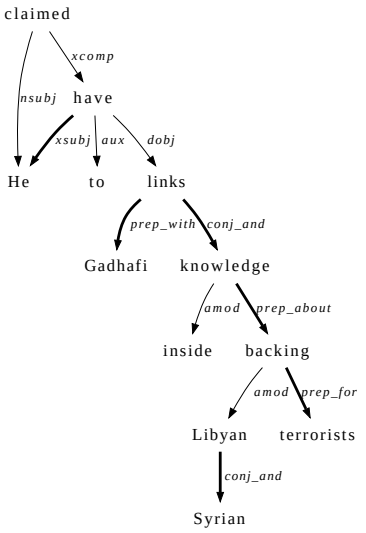


2. Redukované závislosti

(Collapsed dependencies.)

Stromová struktura je zde narušena, mohou vzniknout i cykly. Jsou přidány další závislosti, zde např. *xsubj(have-4, He-1)*. V případě předložek se hrana, která vstupuje do uzlu reprezentujícího předložku, spojí s hranou, která z něj vystupuje, a samotný uzel se odstraní. Nově vzniklá hrana bude mít v názvu předložku. Např. z hran *prep(links, with)* a *pobj(with, Gadhafi)* vznikne nová hrana *prep_with(links, Gadhafi)*. Redukovány jsou i víceslovné předložky. Podobně je to i s koordinacemi.

nsubj(claimed-2, He-1)
xsubj(have-4, He-1)
aux(have-4, to-3)
xcomp(claimed-2, have-4)
dobj(have-4, links-5)
prep_with(links-5, Gadhafi-7)
amod(knowledge-11, inside-10)
conj_and(links-5, knowledge-11)
amod(backing-16, Libyan-13)
conj_and(Libyan-13, Syrian-15)
prep_about(knowledge-11, backing-16)
prep_for(backing-16, terrorists-18)

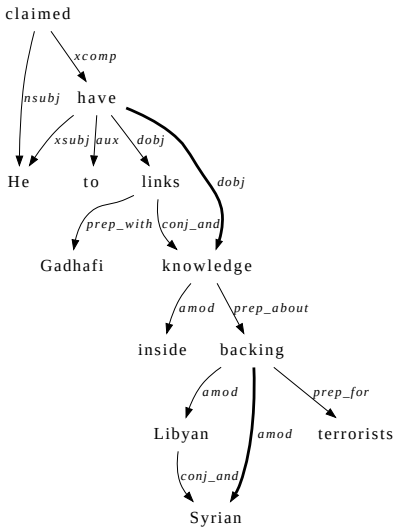


3. Redukované závislosti s propagací koordinací

(Collapsed dependencies with propagation of conjunct dependencies, někdy též *Standard Stanford dependencies.)*

Doplní se další hrany na základě koordinací. Pokud jsou B a C dva uzly, mezi kterými je vztah koordinace (jsou propojeny např. hranou *conj_and*) a jeden z nich (např. B) je závislý na uzlu A, pak nově přidaná hrana povede z uzlu A do uzlu C. Např. *dobj(have-4, links-5)* a *conj_and(links-5, knowledge-11)* → *dobj(have-4, knowledge-11)*

nsubj(claimed-2, He-1)
xsubj(have-4, He-1)
aux(have-4, to-3)
xcomp(claimed-2, have-4)
dobj(have-4, links-5)
prep_with(links-5, Gadhafi-7)
amod(knowledge-11, inside-10)
dobj(have-4, knowledge-11)
conj_and(links-5, knowledge-11)
amod(backing-16, Libyan-13)
conj_and(Libyan-13, Syrian-15)
amod(backing-16, Syrian-15)
prep_about(knowledge-11, backing-16)
prep_for(backing-16, terrorists-18)

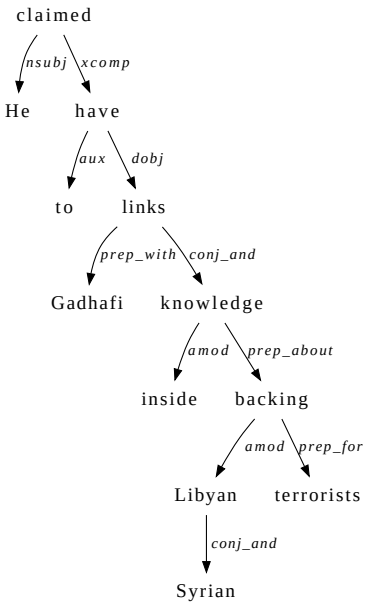


4. Redukované závislosti zachovávající stromovou strukturu

Originální název: *Collapsed dependencies preserving a tree structure.*

Hrany, které by narušily stromovou strukturu se nepřidají, grafem je tedy strom. V našem příkladu se tedy tedy nepřidá hrana *xsubj(have-4, He-1)*, ani hrany přidané na základě koordinací.

nsubj(claimed-2, He-1)
aux(have-4, to-3)
xcomp(claimed-2, have-4)
dobj(have-4, links-5)
prep_with(links-5, Gadhafi-7)
amod(knowledge-11, inside-10)
conj_and(links-5, knowledge-11)
amod(backing-16, Libyan-13)
conj_and(Libyan-13, Syrian-15)
prep_about(knowledge-11, backing-16)
prep_for(backing-16, terrorists-18)



Souhrný přehled jednotlivých stylů stanfordských závislostí lze nalézt v tabulce 4.1.

Stanfordský parser jsme vybrali kvůli jeho přiblížení se sémantické reprezentaci věty. Díky tomu není třeba mít složitou postproceduru pro řešení koordinací, předložek a předložkových skupin atd. Nepříjemný se může zdát přístup ke kopulárním (sponovým) slovesům. Např. ve větě *Tom is a boy* získáme závislost *nsubj(boy, Tom)*, tedy místo slovesa *is* je zde na místě řídícího členu slovo *boy*. Autoři se obhajují tím, že tato reprezentace je bližší k logickému zápisu věty. Ne vždy nám tento způsob může vyhovovovat a proto je třeba s tím počítat. Týká se to následujících sloves: *appear, be, become, disappear, remain, resemble, seem, stay*. Více bude vysvětleno v kapitole 4.1.1.

Mezi další výhody Stanfordského parseru lze uvést jednoduché používání, vstup není třeba předem nijak připravit (i když je tu možnost použít vlastní tokenizaci nebo morfologický tagger) a výstup je dobře zpracovatelný. Nevýhodou naopak může být fakt, že ze Stanfordského parseru nelze získat lemmata (základní tvar slova).

4.1.1 Zpracování výstupu stanfordských závislostí

V této kapitole bude vysvětleno, jak je dále zpracován výstup ze Stanfordského parseru. Pro naši úlohu používáme typ výstupu redukovaných závislostí s propagací koordinací, protože z tohoho typu výstupu je možno získat pro danou úlohu největší množství užitečných informací. Při zpracování jde především o převod názvů stanfordských závislostí a zpracování jazykových jevů apozice a kopuly. Ve snaze přiblížit se co nejvíce struktuře patternů v PDEV, zvolili jsme i podobnou terminologii, jakou používá Patrick Hanks v definicích patternů.

V tabulce 4.3 je ukázka, jaké argumenty a jejich funkce obdržíme z věty *He claimed to have links with Gadhafi, and inside knowledge about Libyan and Syrian backing for terrorists.* (BNC50).

Tento převod probíhá pouze na slovesných argumentech, tedy na slovech, která závisí na slovesu – slovesa poznáme tak, že mají POS tag začínající na VB. Před samotným převodem lze použít procedury `find_appos` a `find_copula`. První z nich řeší apozici a druhá kopulární (sponová) slovesa.

Apozice

Procedura *find_appos* nejdříve nalezne hranu $B \rightarrow C$, která je označena popiskem *appos*, a pro všechny hrany, které vedou z uzlu A do uzlu B, přidá hranu $A \rightarrow C$ se stejným popiskem jako má hrana $A \rightarrow B$.

V příkladu na obrázku 4.1 vidíme část závislostního grafu věty *Heath, a glutton for work, had also to devote large chunks of energy and concentration to the EEC and to Ireland as the Sunningdale Conference on power-sharing approached.* (BNC50).

V levé části obrázku je původní graf, hrana apozice spojuje slova *Heath* a *glutton*. V pravé části je graf s přidanými hranami *nsubj(had, glutton)* a *xsubj(devote, glutton)* – jsou vyznačeny tučnou čarou.

Basic	**Collapsed**	**Propagation**	**Tree**
nsubj(claimed, He)	nsubj(claimed, He)	nsubj(claimed, He)	nsubj(claimed, He)
	xsubj(have, He)	xsubj(have, He)	
aux(have, to)	aux(have, to)	aux(have, to)	aux(have, to)
xcomp(claimed, have)	xcomp(claimed, have)	xcomp(claimed, have)	xcomp(claimed, have)
dobj(have, links)	dobj(have, links)	dobj(have, links)	dobj(have, links)
prep(links, with) pobj(with, Gadhafi)	prep_with(links, Gadhafi)	prep_with(links, Gadhafi)	prep_with(links, Gadhafi)
amod(knowledge, inside)	amod(knowledge, inside)	amod(knowledge, inside)	amod(knowledge, inside)
		dobj(have, knowledge)	
conj(links, knowledge) cc(links, and)	conj_and(links, knowledge)	conj_and(links, knowledge)	conj_and(links, knowledge)
amod(backing, Libyan)	amod(backing, Libyan)	amod(backing, Libyan)	amod(backing, Libyan)
cc(Libyan, and) conj(Libyan, Syrian)	conj_and(Libyan, Syrian)	conj_and(Libyan, Syrian)	conj_and(Libyan, Syrian)
		amod(backing, Syrian)	
prep(knowledge, about) pobj(about, backing)	prep_about(knowledge, backing)	prep_about(knowledge, backing)	prep_about(knowledge, backing)
prep(backing, for) pobj(for, terrorists)	prep_for(backing, terrorists)	prep_for(backing, terrorists)	prep_for(backing, terrorists)

Tabulka 4.1: Přehled všech stylů STD pro větu *He claimed to have links with Gadhafi, and inside knowledge about Libyan and Syrian backing for terrorists.*

stanfordské závislosti	funkce argumentu
nsubj, agent, xsubj	agent
dobj, nsubjpass	object
iobj	iobject
advmod, prep_*	adverbial
prepc_*, advcl	adverbial-cls
ccomp	that-cls
xcomp + "to"	to+inf
xcomp + "-ing"	ing-cls
acomp, pred_comp	pred_comp
aux, prt, mark, auxpass, neg	–

Tabulka 4.2: Převod stanfordských závislostí na funkce argumentů

verb	argument	stdr rel	funkce argumentu
claimed-2/VBD	He-1/PRP	nsubj	agent
claimed-2/VBD	have-4/VB	xcomp	to+inf
have-4/VB	He-1/PRP	xsubj	agent
have-4/VB	links-5/NNS	dobj	object
have-4/VB	knowledge-11/NN	dobj	object
have-4/VB	terrorists-18/NNS	prep_for	adverbial

Tabulka 4.3: Extrakce argumentů z výstupu stanfordských závislostí

Kopula

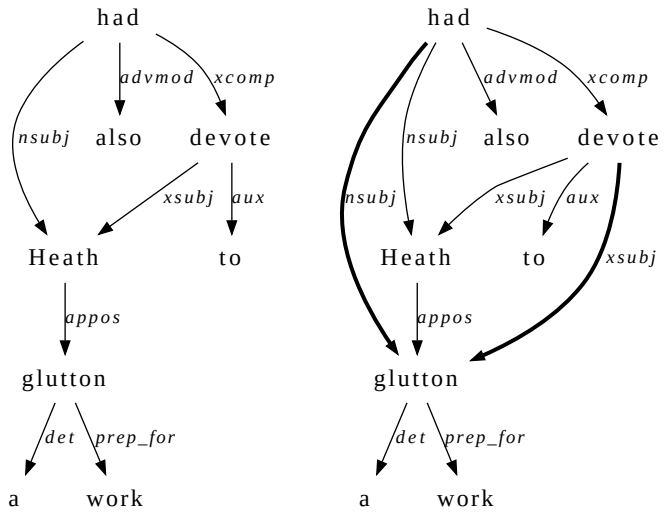
Ve stanfordských závislostech nastává jev, že hrana s označením *nsubj* vychází z uzlu, který není slovesem. Stává se tomu u kopulárních (sponových) sloves. Pokud chceme zpracovávat slovesné argumenty vždy stejným způsobem, je třeba tento jev nejdříve odstranit. Slovesa, u nichž k tomuto dochází jsou *appear, be, become, disappear, remain, resemble, seem, stay.*[2]

Procedura funguje tak, že nalezne hranu s názvem *cop* a vymění její konce. Graficky je to znázorněno na obrázku 4.2. Toto je jednodušší případ, protože stačí pouze zaměnit slova *seems* a *remarkable*, pak je v pořádku, že slovo *It* závisí na slovesu *seems* atd.

Problém nastává, když slovesným doplňkem je substantivum a na něm závisí např. přídavné jméno. Nestačí pouhá výměna uzlů hrany *cop*, ale je třeba určit, které závislosti patří ke jménu a které ke sponovému slovesu.

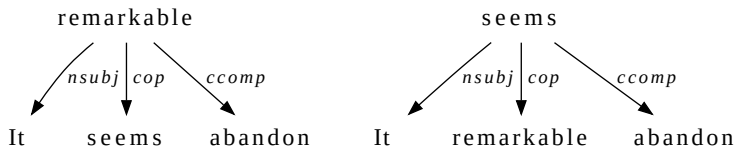
Na obrázku 4.3 se podívejme například na slovo *thinker*, které bylo původně řídícím uzlem. Po úpravě se na jeho místo dostává sponové sloveso *was* a zde dochází k přerozdělění hran. Hrana *complm* vede z *was*, ale hrany *det* a *amod* nyní vychází z *thinker*, protože člen a adjektivum závisí na jménu, ne na slovesu.

[2]K dnešnímu dni (1.7.2010) nemá ani jedno z uvedených sloves status *complete* v PDEV, tedy při zpracování konkordancí zatím nebylo třeba se tímto zabývat. Tato funkce se však uplatnila při evaluaci parsingu na datech z PEDT a může být uplatněna do budoucna, až bude zkompilováno některé z těchto sloves.



Obrázek 4.1: Zpracování apozice

It seems remarkable that the three who remained in France did not abandon their mission following the arrests in Ireland.



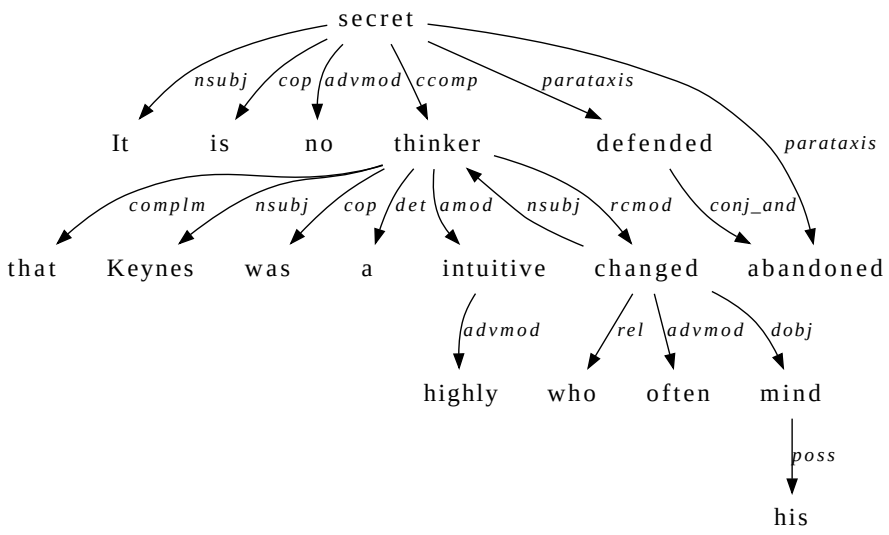
Obrázek 4.2: Zpracování kopuly

Popis algoritmu pro zpracování kopuly:

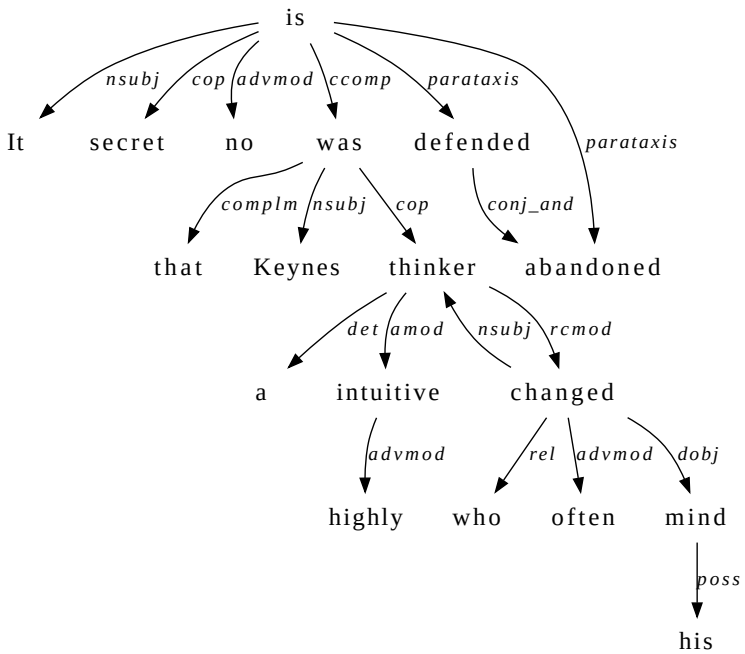
```
pro každou hranu G -> D s označením cop:
    (G = governor, D = dependent)
    pro každou hranu e (G -> X), kde X <> D
     & e <> det|prep_of|amod|rcmod|nn:
        odstraň hranu G -> X a přidej hranu D -> X
         (se stejným označením jako G -> X)
    pro každou hranu Y -> G, kde e <> nsubj|xsubj
        odstraň Y -> G a přidej Y -> D
    nakonec odstraň G -> D a přidej D -> G
```

Experimentální ověření účinnosti zapojení procedur pro zpracování apozice a kopuly lze nalézt v kapitole o evaluaci (viz 4.3.3).

It is no secret that Keynes was a highly intuitive thinker who often changed his mind: in 1930 he defended the gold standard, which he had previously attacked, and in 1931 he abandoned the principle of free trade, which he had previously upheld.



Obrázek 4.3: Část závislostního grafu před zpracováním kopuly



Obrázek 4.4: Část závislostního grafu po zpracování kopuly

4.2 Další parsery a metody

V rámci evaluace byly použity také další parsery (Charniak-Johnsonův, McDonaldův) a metoda založená na srovnávání textů s regulárními výrazy.

4.2.1 Charniak-Johnsonův parser

Dalším použitým nástrojem (pouze pro evaluaci) je Charniak-Johnsonův parser [24]. Jedná se o složkový parser, k převodu na závislostní strukturu je třeba použít proceduru, kterou poskytuje Stanfordský parser:
`class EnglishGrammaticalStructure`.

Výsledek je jedním ze čtyř typů stanfordských závislostí, pokud použijeme redukované závislosti s propagací koordinací, můžeme na výstup aplikovat naší proceduru pro zpracování stanfordských závislostí.

Charniak-Johnsonův parser již obsahuje model pro angličtinu natrénovaný na sekcích Penn Treebank (PTB). Vstup pro Charniak-Johnsonův parser není třeba předem upravovat, pouze označit hranice vět (věta se uzavře mezi značky `<s>` a `</s>`). Pro tokenizovaný vstup lze použít přepínač `-k`.

4.2.2 McDonaldův parser

McDonaldův parser [25] je závislostní parser. Pracuje na principu hledání minimální kostry grafu. Před použitím McDonaldova parseru je potřeba nejdříve připravit data a mít natrénovaný model. Data musí být v MST nebo CoNLL formátu [26] a opatřena morfologickými značkami (POS tag).

Původně jsme použili McDonaldův parser v rámci TectoMT [27]. V tomto případě jsme použili Morče tagger [28] a po vytvoření závislostního stromu následoval převod na tektogramatický strom, z něhož byly extrahovány závislosti (agenty a objekty) stejnou procedurou jako z testovacích dat (viz kapitola 4.3.2).

V dalším experimentu jsme tedy vzali argumenty slovesa z tektogramatické roviny (pomocí funkce `PML_En_T::GetEChildren`), ale použili jsme jejich funkce z analytické roviny (u pasivních vět jsme syntaktický subjekt považovali za hloubkový objekt).

Výstup McDonaldova parseru se dá přirovnat k základním stanfordským závislostem. Ideální by bylo, kdybychom tento výstup uměli převést na redukované závislosti s propagací koordinací. Je tedy potřeba vyřešit koordinaci a předložky. Existuje však i jiný způsob. Stačí použít model[3] pro McDonaldův parser natrénovaný na stanfordské závislosti, kterým se vytvoří základní závislosti, a následně použít převod přímo z CoNLL formátu na redukované závislosti s propagací koordinací opět pomocí `class EnglishGrammaticalStructure`. Poté lze použít zpracování stanfordských závislostí, které jsme uvedli v kapitole 4.1.1).

Při použití stanfordského modelu pro McDonaldův parser byl nejprve použit MXPOST tagger (Adwait Ratnaparkhi) [29]. Výměnou za Morče tagger jsme experimentálně zjistili vliv výběru morfologického taggeru.

[3] Tento model je volně ke stažení na webových stránkách Stanfordského parseru.

Object

```
1:"V..?" "RB.?"{0,2} [tag="DT.?"|tag="PP\$"]{0,1} "CD"{0,2}
  [tag="JJ.?"|tag="RB.?"|word=","]{0,3} "NN.?.?"{0,2} 2:"NN.?.?" [tag!="NN.?.?"]
2:"NN.?.?" "RB.?"{0,2} 1:"V.N"
2:"NN.?.?" [tag="WP"|tag="PNQ"|tag="CJT"]? [tag="RB.?"|tag="RB"|tag="VM"]{0,5}
  [lempos="be-v"] "RB.?"{0,2} 1:"V.N"
```

Agent

```
2:"NN.?.?" [tag="WP"|tag="PNQ"|tag="CJT"]? [tag="RB.?"|tag="RB"|tag="VM"]{0,3}
  [lempos="be-v"]? "RB.?"{0,2} 1:"V.[^N]?"
2:"NN.?.?" [tag="WP"|tag="PNQ"|tag="CJT"]? [tag="RB.?"|tag="RB"|tag="VM"]{0,3}
  [lempos="be-v"]? "RB.?"{0,2} [lempos="have-v"] "RB.?"{0,2} 1:"V.N"
1:"V.N" "RB.?"{0,2} [word="by"] [tag="DT.?"|tag="PP\$"]{0,1} "CD"{0,2}
  [tag="JJ.?"|tag="RB.?"|word=","]{0,3} "NN.?.?"{0,2} 2:"NN.?.?" [tag!="NN.?.?"]
```

Tabulka 4.4: Pravidla pro extrakci argumentů slovesa definovaná ve Sketch Engine

Výsledky všech popsaných experimentů jsou uvedeny v kapitole 4.3.4 v tabulce 4.9.

4.2.3 Sketch Engine

Sketch Engine [18] je systém, který analyzuje velký korpus a hledá typické argumenty sloves. Kromě standardních korpusových funkcí jako jsou konkordance, třídění a filtrování navíc poskytuje tzv. *word sketches*. Word sketch je automatický jednostránkový přehled gramatických vlastností daného slova a jeho typických kolokací.

Gramatické vztahy jsou v systému Sketch Engine definovány jako regulární výrazy využívající morfologické značky. Sketch Engine používá TreeTagger a lehce modifikovaný Penn tagset.

Př. Gramatický vztah mezi přídavným a podstaným jménem vypadá takto:

```
=modifier
2:"A.*" 1:"N.*"
```

První řádek definuje jméno gramatického vztahu a druhý řádek je druh korpusového dotazu. `1:` a `2:` označují slova, která se mají extrahovat jako první argument (klíčové slovo) a druhý argument (kolokace). Množina těchto pravidel se nazývá *Sketch Grammar*.

V naší úloze jsme se zabývali pouze agentem a objektem, které jsou definovány jako poslední substantivum ve jmenné frázi bezprostředně předcházející resp. následující sloveso.

Díky této definici jsou vyloučena zájmena a jiné slovní druhy na místě agenta/objektu.

Kompletní seznam pravidel použitých při extrakci argumentů pro evaluační experiment je uveden v tabulce 4.4.

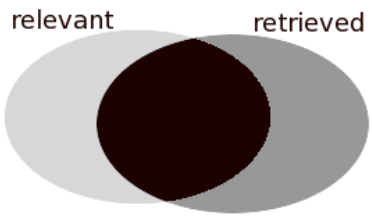
4.3 Evaluace metod pro syntaktickou analýzu

Evaluací metod používaných pro syntaktickou analýzu lokálního kontextu slovesa v anglické větě jsme se zabývali již v [30].

Zjišťovali jsme úspěšnost Stanfordského parseru, kombinace Charniak-Johnsonova parseru s převodem na stanfordské závislosti, McDonaldova parseru a Sketch Engine. Pro vyhodnocení úspěšnosti jsme použili část dat z PEDT.

4.3.1 Evaluační míra

Pro měření úspěšnosti parsingu používáme jednoduchou míru založenou na precision a recall, z nichž spočítáme F-measure jako vážený harmonický průměr precision a recall.



Obrázek 4.5: Obrázek k vysvětlení pojmů precision a recall

Označme množinu T relevantních trojic (sloveso, argument, funkce), které se nacházejí v testovacích datech. Trojice, které získáme pomocí některé parsovací metody označme jako retrieved S (viz obrázek 4.5). Pak je precision definován jako $P = |T \cap S|/|S|$ a recall $R = |T \cap S|/|T|$.

Obecná definice F-measure je

$$F_\beta = \frac{(1+\beta^2)\cdot(P\cdot R)}{(\beta^2\cdot P + R)}\,,$$

zde používáme tradiční F-measure s parametrem $\beta = 1$, tedy

$$F_1 = \frac{2\cdot(P\cdot R)}{P+R}\,.$$

V tabulce 4.5 jsou uvedeny extrahované argumenty ze dvou vět. Jedničky a nuly ve sloupci *Test data* znamenají, zda se trojice (sloveso, argument, funkce) nachází v testovacích datech. V ostatních sloupcích je jednička pouze v případě, že tato trojice byla na výstupu uvedené metody, jinak je zde nula. V další tabulce 4.6 je vyhodnocení úspěšnosti metod na tomto malém vzorku testovacích dat.

4.3.2 Testovací data

Jako testovací data byla použita část PEDT, která obsahuje ruční složkové stromy z WSJ obohacené o analytickou a tektogramatickou rovinu. Konkrétně

Věta 1. *Preliminary tallies by the Trade and Industry Ministry showed another trade deficit in October, the fifth monthly setback this year, casting a cloud on South Korea's export-oriented economy.*
Věta 2. *Esso said the fields were developed after the Australian government decided in 1987 to make the first 30 million barrels from new fields free of excise tax.*

sloveso	argument	funkce	Test data	Sketch Engine	Charniak-Johnson	McDonald	Stanford	Manual
showed-9	tallies-2	agent	1	0	1	1	1	1
showed-9	deficit-12	object	1	1	1	1	1	1
showed-9	setback-19	object	1	0	1	0	0	1
casting-23	cloud-25	object	1	1	1	1	1	1
casting-23	October-14	agent	0	0	0	0	1	0
casting-23	setback-19	agent	0	0	0	0	1	0
said-2	Esso-1	agent	1	0	1	1	1	1
developed-6	fields-4	object	1	1	1	1	1	1
decided-11	government-10	agent	1	0	1	1	0	1
make-15	barrels-20	object	1	0	0	1	1	1
said-2	fields-4	object	0	1	0	0	0	0
decided-11	government-10	object	0	1	0	0	0	0

Tabulka 4.5: Ukázka průběhu evaluace na dvou větách

Metoda	P	R	F_1
Sketch Engine	60,00	37,50	46,15
Charniak-Johnson	100,00	87,50	93,33
McDonald	100,00	87,50	93,33
Stanford	75,00	75,00	75,00
Manual	100,00	100,00	100,00

Tabulka 4.6: Ukázka evaluace výstupu jednotlivých metod

se jedná o sekce 0, 1, 22, 23 a 24, protože nebyly použity pro trénování parserů použitých při evaluaci.

Statistické údaje o testovacích datech
Pro úlohu evaluace bylo použito 9 325 vět, které obsahují celkem 223 032 slov. Celkem zde bylo 21 588 sloves[4] se 31 218 argumenty (subjekt nebo objekt). Různých sloves je 1 795. Po vyřazení pomocných a modálních sloves (pokud byla použita jako plnovýznamová = mají samostatný uzel na tektogramatické rovině) zbývá 18 125 sloves (1 789 různých) a 26 879 argumentů. Pokud se omezíme pouze na substantivní argumenty, máme pouze 15 434 sloves (1 701 různých) a 21 263 argumentů.

Automatická procedura pro extrakci argumentů z ručních stromů
K extrakci argumentů slovesa (pouze subjekty a objekty) jsme použili algoritmus, který na tektogramatické rovině nalezne slovesa (mají POS tag začínající VB a analytický uzel musí být ve stejné větě) a projde všechny jejich argumenty (použita funkce `PML_En_T::GetEChildren`). Funkce argumentu se určí na základě funktoru z tektogramatické roviny a morfologické značky (POS tagu) z analytické roviny.

Část algoritmu, ve které se rozhoduje o funkci argumentu je uvedena na obrázku 4.6.

Kontrola testovacích dat „gold standard data“
Pro kontrolu testovacích dat byl vybrán náhodný vzorek 100 vět (což tvoří 1,07 % všech vět z testovací sady) s 266 argumenty. Všechny argumenty jsme ručně prošli a zkontrolovali, ukázalo se, že dva argumenty jsou určené špatně – chyba v ručně anotovaných stromech. Precision testovacích dat je 99,25 %. Dále chyběly některé trojice (sloveso, argument, funkce) v testovacích datech z důvodu drobných chyb v ručních stromech (např. špatná morfologická značka, špatně přiřazený funktor), což způsobilo, že recall je 96,70 %.

Pro srovnání jsme do evaluace automatické extrakce slovesných argumentů zařadili také použití Stanfordského parseru na ruční stromy z PEDT k získání stanfordských závislostí. Úspěšnost této „metody“ byla překvapivě nízká. Je to především tím, že výstup procedury, která extrahuje argumenty z PEDT, není kompatibilní s výstupem stanfordských závislostí.

Rozdíly jsou následujících druhů:

- Rozdílné pojetí *subject/object raising*[5] a *kauzativních konstrukcí*[6] v testovacích datech a ve výstupu stanfordských závislostí.
- Slova, která byla v PEDT označena jako slovesa (a je u nich označen agent resp. objekt), ale parser jim přiřadil roli adjektiva nebo substantiva.
  Př. *Proper English bells are started off in “rounds,” from the highest-pitched bell to the lowest – a simple/JJ descending/VBG scale/NN using,*

[4] Počítáme pouze slovesa, která měla aspoň jeden argument typu agent nebo objekt.

[5] „Raising“ je forma kontroly argumentu, při které argument sémanticky náleží do podřízené klauze, ale syntakticky je realizován jako člen nadřazené klauze.

[6] Věty typu „make someone do something“, „get someone to do something“ a „have someone do something.“

```
1) functor RHEM, PREC, NEG => "ignored"
2) arg je sloveso (VB*)
   2.1 v aux.rf existuje tag "TO" => "to+inf"
   2.2 functor je EFF => "that-cls"
   2.3 tag je VBG => "ing-cls"
   2.4 v arg->children existuje tag W* => "wh-cls"
   2.5 jinak => "cls"
3) arg není sloveso (nemá tag VB*)
   3.1 functor je ACT => "agent"
   3.2 v aux.rf existuje předložka
      3.2.1 functor je ADDR a předložka je "to" => "iobject"
      3.2.2 jinak => "adverbial"
   3.3 functor je PAT
      3.3.1 sloveso je "be" => "pred_comp"
      3.3.2 jinak => "object"
   3.4 functor je ADDR => "iobject"
   3.5 functor je LOC, DIR*, T*, MANN => "adverbial"
   3.6 functor je CPHR
      3.6.1 tag je JJ* nebo RB* => "pred_comp"
      3.6.2 sloveso je "be" => "pred_comp"
      3.6.3 jinak => "object"
   3.7 tag je RB => "adverbial"
   3.8 functor je RSTR a sloveso je "be" => "pred_comp"
   3.9 functor je EFF => "object"
   3.10 functor je DIFF => "object"
   3.11 functor je EXT a tag je JJ* nebo RB* => "adverbial"
   3.12 functor je DPHR
      3.12.1 tag je NN* => "object"
      3.12.2 jinak => "pred_comp"
   3.13 jinak => "---" (neurčeno)
```

Obrázek 4.6: Algoritmus pro extrakci testovacích dat z PEDT

Metoda	test data			corrected test data		
	P	R	F_1	P	R	F_1
Sketch Engine	67,69	33,08	44,44	68,46	32,60	44,17
Charniak-Johnson + STD	84,07	85,34	84,70	86,30	85,35	85,82
Morče + McDonald + STD	86,36	78,57	82,28	88,84	78,75	83,50
Stanford	83,46	81,58	82,51	86,15	82,05	84,05
Manual + STD	87,60	84,96	86,26	89,92	84,98	87,38

Tabulka 4.7: Evaluace na náhodném vzorku 100 vět – opravená data

in larger churches , as many as 12 bells. V této větě má slovo *descending* POS tag *VBG*, PEDT určí, že agentem je *scale*. Na výstupu stanfordských závislostí se objeví *amod(scale-23, descending-22)*, kde *amod* znamená *adjectival modifier*.

- Ve vztažných větách v PEDT je jako subjekt/objekt označováno vztažné zájmeno. Ve výstupu STD je v tomto případě označeno přímo jméno, které je zastoupeno vztažným zájmenem.
  Př. *I'm the only one who said there would be...*
  Ve stanfordských závislostech je vztažné zájmeno obvykle připojeno hranou *rel* a ve většině případů zastupuje ve vztažné větě subjekt, tím zvýšíme recall. Ale precision se pokazí, protože zde bude navíc trojice *(said-8, one-6, agent)*. V tabulce 4.8 lze nalézt výsledky experimentu, kdy hranu *rel* vůbec neuvažujeme (no rel), nebo ji považujeme za subjekt (rel=sbj) resp. objekt (rel=obj).

- Některé argumenty chybí z důvodu drobných chyb v PEDT.

4.3.3 Evaluace zpracování výstupu stanfordských závislostí

Zpracování se mírně liší pro případ, kdy chceme evaluovat vůči datům z PEDT. V PEDT není např. označován tzv. *kontrolující subjekt*, takže by docházelo ke zhoršení precision (přesnosti), naopak v PEDT je jako subjekt/objekt vztažné věty označováno vztažné zájmeno. V tabulce 4.8 lze nalézt výsledky experimentu, kdy hranu *rel* vůbec neuvažujeme (no rel), nebo ji považujeme za subjekt (rel=sb) resp. objekt (rel=obj). Nejlepšího výsledku dosáhneme, pokud hranu *rel* považujeme vždy za subjekt. Dále v této tabulce vidíme, že největší úspěšnost je dosažena, pokud použijeme proceduru pro apozici i pro copulu.

4.3.4 Evaluace parserů a metod

Při evaluaci metod použitelných pro extrakci argumentů slovesa jsme se zabývali také tím, jak nejlépe použít McDonaldův parser. V kapitole 4.2.2 jsme uvedli několik možných způsobů použití:

1. *Morče + McDonald s modelem z TectoMT s následným sestavením tektogramatického stromu + extrakce argumentů z tektogramatické roviny*

Metoda	P	R	F_1
Stanford no rel	83,47	75,64	79,36
Stanford rel=sb	82,97	78,53	80,69
Stanford rel=obj	80,32	76,15	78,18
Stanford xsubj=sb	79,35	79,09	79,22
Stanford + Apozice (rel=sb)	82,49	81,21	81,84
Stanford + Copula (rel=sb)	83,01	79,01	80,96
Stanford + Apozice + Copula (rel=sb)	82,53	81,70	82,11

Tabulka 4.8: Evaluace zpracování stanfordských závislostí

	Metoda	P	R	F_1
1	Morče + McDonald + tecto extraction	57,98	61,72	59,79
2	Morče + McDonald + anal extraction	79,79	72,10	75,75
3	Morče + McDonald + STD + AC	85,12	79,53	82,23
4	MXPOST + McDonald + STD + AC	84,72	78,56	81,53

Tabulka 4.9: Evaluace způsobů použití McDonaldova parseru (STD = stanfordské závislosti, AC = použití procedury pro zpracování apozici a kopuly)

Výsledná úspěšnost (F_1) tohoto postupu byla 59,79 % (viz řádka *tecto extraction* v tabulce 4.9). Problematický byl pravděpodobně převod na tektogramatický strom – zvláště přiřazení tektogramatických funktorů.

2. *Morče + McDonald + extrakce argumentů z analytické roviny*
   Použití parseru bylo stejné jako v předchozím případě, ale argumenty byly extrahovány z analytické roviny. Tím se zvýšil precision o 21,8 %, recall o 10,4 % a F-measure o 16 % (viz *anal extraction* v tabulce 4.9).

3. *Morče + McDonald s modelem natrénovaným na stanfordské závislosti + převod na STD + zpracování STD se zapojením procedury pro apozici a kopulu*
   Nejlepší se ukázalo být použití procedur pro zpracování apozice i kopuly (viz kap. 4.3.3), použijeme je i zde. Výsledné hodnoty pro precision, recall a F-measure se zvětšily v průměru o 6-7 % v porovnání s předchozím způsobem.

4. *Stejné jako předchozí způsob, ale místo Morče taggeru použijeme MXPOST tagger.*
   Precision se snížil o 0,4 %, recall o 1 % a F-measure o 0,7 %.

V tabulce 4.10 jsou uvedeny souhrné výsledky našeho experimentu. Ke zpracování stanfordských závislostí jsme použili proceduru včetně vyřešení apozice a copuly. Jelikož Sketch Engine narozdíl od ostatních metod poskytuje pouze substantivní argumenty, udělali jsme i druhou verzi, kde nesubstantiva odfiltrujeme z testovacích dat i z výstupu ostatních metod. Tím se výrazněji zvýšil

Method	Všechny argumenty			Pouze substantiva		
	P	R	F_1	P	R	F_1
Sketch Engine	70,46	35,80	47,48	70,31	45,15	54,99
Charniak-Johnson + STD + AC	83,90	83,59	83,74	84,48	84,67	84,57
Morče + McDonald + STD + AC	85,12	79,53	82,23	86,27	79,52	82,76
Stanford + AC	82,53	81,70	82,11	82,97	82,54	82,76
Manual + STD + AC	86,61	82,53	84,52	87,79	86,60	87,19

Tabulka 4.10: Výsledky evaluace parsovacích metod

recall u Sketch Engine. Mírný pokles precision je způsobený tím, že Sketch Engine považuje za substantiva i některá slova, která v PEDT substantiva nejsou.

Překvapivý je velmi nízký rozdíl mezi ručně anotovanými stromy doplněnými stanfordskými závislostmi a ostatními metodami (s výjimkou Sketch Engine). Nejlépe (dle F-measure) dopadl Charniak-Johnsonův parser doplněný stanfordskými závislostmi. Stanfordský parser je pouze o 1,5 % horší, ale oproti ostatním metodám je jeho použití nejjednodušší, protože je vše v jednom.

Velmi nízký recall Sketch Engine byl očekávatelný, ale nepříliš vysoký precision je překvapivý. Sketch Engine dělá chyby především ve složitějších větách, kde jsou argumenty od slovesa více vzdáleny nebo v případě, že je argument sktrukturovaný, např. ve větě *A form of asbestos once used to make...* se jako objektem slovesa *used* určí *asbestos* místo *form*.

4.4 Shrnutí

V této kapitole jsme se zaměřili na extrahování argumentů slovesa za použití automatické syntaktické analýzy angličtiny. Extrakce argumentů je důležitá pro další práci – vytváření seznamů slov realizujících sémantické typy a při automatickém přiřazování slovesných patternů.

Při evaluaci jednotlivých metod jsme se zaměřili především na agenty a objekty. Podle uvedených výsledků (viz tabulka 4.10) je nejlepší použít kombinaci Charniak-Johnosonova parseru s následným převodem složkových stromů na stanfordské závislosti. Přesto však používáme Stanfordský parser, protože jeho úspěšnost není výrazně horší a použití jednoho nástroje je jednodušší než kombinace více nástrojů.

Kapitola 5

Systém sémantických typů

V definicích převážné většiny patternů jsou kolokační pozice určeny sémantickým typem. V dostupných označkovaných konkordancích, kterým je přiřazen pattern, však není nijak vyznačeno, které slovo ve větě realizuje sémantický typ uvedený v definici patternu.

V této kapitole se zabýváme způsobem, jak tato slova najít a přiřadit sémantickým typům. Toto propojení má význam pro sestavení seznamů slov, která mohou realizovat jednotlivé sémantické typy, což lze dále využít pro

- návrh klasifikátoru, který automaticky rozpoznává pattern slovesa v dané větě
- vytvoření definice pro každý sémantický typ
- vylepšení definic patternů, např. úpravou či přidáním sémantického typu
- zamyšlení se nad tím, zda je sémantický typ užitečný
- vytvoření hierarchie sémantických typů podložené reálnými daty

V předchozí kapitole jsme vysvětlili, jak z vět extrahujeme argumenty slovesa. V této kapitole ukážeme, že syntaktická analýza a následná extrakce argumentů slouží jako základ pro další zpracování, jehož výsledkem jsou seznamy slov (substantiv) realizujících dané sémantické typy.

K získání těchto seznamů použijeme označkované konkordance, z nichž extrahujeme slovesné argumenty a následně je srovnáme s kolokačními pozicemi v příslušném patternu. Na základě toho vytvoříme seznam dvojic (sémantický typ, substantivum), kde substantivum realizuje daný sémantický typ.

Tuto problematiku jsme částečně popsali již v [31].

5.1 Použité nástroje

Syntaktickou analýzu kontextu slovesa je možné doplnit použitím nástroje na rozpoznání jmenných entit, tzv. NE klasifikátorem. Pro další práci s extrahovanými argumenty – extrakce lexikálních jednotek reprezentujících sémantické typy v BNC50 a automatické rozpoznávání patternů – využijeme také zjednodušený slovník pro převod plurálu na singulár.

5.1.1 Stanfordský NE klasifikátor

Analýzu kontextu slovesa je možné doplnit využitím některého klasifikátoru pro rozpoznávání jmenných entit (NER, Named Entity Recognizer). Zvolili jsme Stanfordský NE klasifikátor [32] s modelem, který rozlišuje tři typy jmenných entit: *Person* pro jména osob, *Organization* pro jména organizací a *Location* pro zeměpisné názvy.

Použití NE klasifikátoru je nezávislé na syntaktické analýze. Výstup z něj je možné zadat jako volitelný parametr do procedury, která zpracovává výstup Stanfordských závislostí. Stanfordský NER nabízí několik stylů, jak mají být ve výstupním souboru označeny jmenné entity. Zvolili jsme takový, kde je jmenná entita uzavřena mezi značky

`<typ_jmenné_entity>jmenná entita</typ_jmenné_entity>`.

Věta na výstupu vypadá takto: `<PERSON>`Abdallah`</PERSON>` never abandoned his ambition to rule `<LOCATION>`Syria`</LOCATION>`.

Při zapojení NER do zpracování Stanfordských závislostí se pouze spojí jednotlivé tokeny jmenné entity, které pak vystupují jako jeden celek. Z technických důvodů jsou mezery nahrazeny podtržítky. Extrahovaný argument, který je částí jmenné entity se nahradí tímto celkem a připojí se k němu nový POS tag popisující typ jmenné entity – *NEP* pro typ *Person*, *NEO* pro typ *Organization* a *NEL* pro typ *Location*. V současné verzi není tento tag nijak dále využíván, ale v budoucnu by měl pomoci částečně disambiguovat sémantické typy.

5.1.2 Slovník

V průběhu řešení diplomové práce vznikla potřeba automaticky převádět množné číslo na jednotné. Použitý Stanfordský parser nepřevádí slova na jejich základní tvary (lemmata). Převod na singulár potřebujeme pouze pro sjednocení slovních forem při vytváření seznamů lexikálních jednotek reprezentující sémantické typy, protože až na pár výjimek (např. *Human Group*) není podstatné, zde je slovo použito v singuláru či plurálu.

Po syntaktické analýze označkovaných konkordancí jsme vytvořili unikátní seznam všech substantiv v singuláru (POS tag je NN) seřazený podle jejich frekvence v těchto datech. Následně jsme pomocí funkce `PL_N()` implementované v modulu `Lingua::EN::Inflect` [33] vygenerovali plurál. Tento zjednodušený slovník byl uložen setříděný podle singuláru v souboru jako prostý text.

Slovník je následně použit při populaci sémantických typů. Pokud se narazí na slovo POS tagem NNS, zkusí se vyhledat ve slovníku mezi plurály. Při úspěšném nalezení se převede na singulár. V opačném případě se ponechá v plurálu, protože singulár se v textu zřejmě nevyskytuje nebo ani neexistuje. Víceznačnosti se vyřeší již při načítání slovníku ze souboru, tedy se použije první nalezená možnost – podle abecedy.

Toto řešení nabízí několik možných vylepšení, např. vylepšit práci se slovníkem, rozumnější nakládání s víceznačnostmi, popř. použití již nějakého existujícího slovníku či provedení lemmatizace.

5.2 Extrakce argumentů a přiřazení sémantických typů

K dispozici máme cca 200 000 označkovaných vět z korpusu BNC50. Věty jsou rozděleny podle sloves a značkou je číslo patternu slovesa použitého v dané větě (viz 2.1.2). Ze souboru vět odstraníme příliš dlouhé věty (≥ 100 tokenů) a věty se značkou u nebo x. Ostatní věty syntakticky zanalyzujeme a nalezneme argumenty označkovaného slovesa. Z každé věty získáme množinu dvojic *(argument, funkce)*..

Dále nás zajímají pouze argumenty na pozicích, které by mohly být popsány sémantickým typem, což je agent, objekt a adverbiál vyjádřené předložkovou frází. U adverbiálů potřebujeme znát také původní název gramatického vztahu (*Rel*) ze stanfordských závislostí, protože obsahuje předložku.

V tabulce 5.1 jsou uvedeny extrahované argumenty z věty *We've devoted the project to social awareness.* (BNC50).

stanfordské závislosti	extrahované argumenty
nsubj(devoted-4, We-2)	agent: We/PRP
aux(devoted-4, 've-3)	
det(project-6, the-5)	
dobj(devoted-4, project-6)	object: project/NN
amod(awareness-9, social-8)	
prep_to(devoted-4, awareness-9)	adverbial: awareness/NN rel: prep_to

Tabulka 5.1: Extrakce argumentů ze stanfordských závislostí

Tato věta je označena patternem č. 4 (viz obrázek 3.2), který na pozici agenta povoluje sémantické typy *Human* nebo *Institution*, na pozici objektu sém. typ *Activity* a dále definuje povinnou předložkovou frázi s předložkou *to* a sémantickým typem *Anything*.

Srovnáním pozic argumentů extrahovaných z věty s pozicemi v patternu získáme posloupnost dvojic $(S_i, w_i), i = 1, \ldots, N$, kde S_i ($S_i = \{t_1, \ldots, t_n\}; n \geq 1$) je množina sémantických typů přípustných na dané pozici a w_i je slovo, které realizuje některý sémantický typ z množiny S_i. Celkový počet dvojic získaných ze všech označkovaných vět je N.

Množinu všech sémantických typů označme T, pak platí:

$$\bigcup_i S_i \subseteq T$$

Dvojice z ukázkové věty jsou uvedeny v tabulce 5.2, kde slovo je navíc doplněno morfologickou značkou (POS tag). Sémantické role a množiny lexikálních jednotek zde vůbec neuvažujeme.

Z celkového počtu 202 348 použitých vět získáme 224 259 dvojic (S, w). V každém patternu je průměrně 1,77 slotů pro sémantické typy. Pokud uvažujeme i rozložení patternů, průměrný počet slotů je také 1,77. Maximální počet dvojic (S, w) je 358 944 (maximální je pouze pokud neuvažujeme koordinace a

sémantický typ	slovo/POS_tag
Human | Institution	We/PRP
Activity	project/NN
Anything	awareness/NN

Tabulka 5.2: Propojení se sémantickými typy z patternu č. 4 slovesa *devote*

apozice). Toto číslo jsme obdrželi tak, že pro každou větu označenou patternem p započítáme maximální počet slotů pro sémantické typy definovaných v patternu p. Není to tedy počet argumentů, v některých patternech je argument vyjádřen pouze výčtem lexikálních jednotek (např. *[[Human]] abandon ship*)

Důvodů, proč skutečný počet dvojic je o více než třetinu menší, může být několik. V pasivních větách často bývá nevyjádřený agent. Vezměme např. sloveso *append*, které má 2 patterny a v každém z nich jsou tři sloty pro sémantický typ. Celkem máme označkovaných a zanalyzovaných 81 vět, tedy maximální počet dvojic je 243. V těchto větách bylo nalezeno pouze 27 agentů, což je pouhá třetina, agent bývá častěji nevyjádřený.

Další možností je nepovinný argument, protože v našem odhadu ho také započítáme.

V některých případech udělá chybu parser – špatně zanalyzuje větu a nenajde argumenty.

Ve čtvrtině nalezených dvojic (S, w) je w nesubstantivum (většinou zájmena), se kterými nebudeme dále pracovat. Ze zbylých 165 278 dvojic potřebujeme odstranit víceznačnost – určit, který typ z množiny S je realizován slovem w. Tím se zabývá následující kapitola.

5.3 Definice statistických charakteristik

V patternech se velmi často vyskytují alternace sémantických typů, např. ve druhém patternu slovesa devote v pozici agenta alternují sémantické typy *Human* a *Institution* (viz kap. 2.1.1).

Pokud chceme vytvořit pro každý sémantický typ seznam slov, která ho nejčastěji realizují, musíme nejprve odstranit víceznačnost.

Definujme $c(S, w)$ jako počet dvojic (S, w) nalezených v datech. Dále definujme čtyři statistické charakteristiky (atributy) $F_1, \ldots, F_4$ pro každou dvojici (t, w):

$$F_1(t, w) = \sum_{t \in S} c(S, w)$$

$$F_2(t, w) = c(\{t\}, w)$$

$$F_3(t, w) = \sum_{t \in S} \frac{1}{|S|} \cdot c(S, w)$$

$$F_4(t, w) = \sum_{t \in S} \frac{c(\{t\}, w)}{\sum\limits_{|S|=1} c(S, w)} \cdot c(S, w)$$

S	w	$c(S, w)$
Plane	aircraft	20
Plane | Human	aircraft	12
Ship | Plane | Road Vehicle	aircraft	1
Boat | Plane	aircraft	3
Human	aircraft	3
Road Vehicle	aircraft	0
Ship	aircraft	0
Boat	aircraft	0

Tabulka 5.3: Relevantní dvojice (S, w) pro výpočet atributů dvojice (*Plane*, *aircraft*)

S	w	$c(S, w)$	F_1	F_2	F_3	F_4
Plane	aircraft	20	20	20	20	20
Plane | Human	aircraft	12	12	0	6	10,43
Ship | Plane | Road Vehicle	aircraft	1	1	0	0,33	1
Boat | Plane	aircraft	3	3	0	1,5	3
$\sum$			36	20	27,83	34,43

Tabulka 5.4: Výpočet atributů pro dvojici (*Plane*, *aircraft*)

Jako ukázku vezměme například sémantický typ *Plane* a slovo *aircraft*. V tabulce 5.3 jsou vybrané dvojice, které jsou relevantní pro výpočet F_i(*Plane*, *aircraft*).

V tabulce 5.4 jsou přírůstky jednotlivých atributů $F_1, \ldots, F_4$ pro dvojici *(Plane, aircraft)*. Atribut F_1 udává, že slovo *aircraft* se v souvislosti se sémantickým typem *Plane* vyskytuje celkem 36×, tedy $c(S, w) = 36$, kde *Plane* $\in S$ a $w =$ *aircraft*. Další atribut F_2 říká, že se tato dvojice vyskytuje samostatně celkem 20×. Z těchto dvou hodnot můžeme zjistit, že problém víceznačnosti bylo třeba řešit celkem 16×.

Definice atributu F_3 vypovídá, že se počet výskytů dvojice (S, w) dělí rovným dílem mezi sémantické typy v množině S. Zde se *aircraft* se vyskytlo 12× současně s *Plane* a *Human* ($c(\{$*Plane*, *Human*$\}$, *aircraft*$) = 12$), množina S obsahuje dva prvky, počet výskytů se rozdělí na poloviny.

Složitější je výpočet přírustku atributu F_4. Zde je potřeba dohledat samostatné výskyty s ostatními sémantickými typy, které se nacházejí v množině (viz tabulka 5.3). Podívejme se opět na *Plane* a *Human*: *aircraft* se vyskytuje se sémantickým typem *Plane* samostatně 20× (F_2(*Plane*, *aircraft*) $= 20$) a s *Human* pouze 3× (F_2(*Human*, *aircraft*) $= 3$). Celkový počet výskytů (12) tedy rozdělíme v poměru $20 : 3$, tedy $\frac{20}{23} \cdot 12 = 10,43$, což je přírůstek atributu F_4 pro dvojici (*Plane*, *Human*).

Sečtením sloupců (přírůstků) v tabulce 5.4 získáme výsledné hodnoty atributů $F_1, \ldots, F_4$ (viz poslední řádek tabulky).

Další statistickou charakteristikou dvojice (t, w) jsou proměnné PMI_3 a

PMI_4. Jedná se o vzájemnou informaci (PMI, pointwise mutual information).

$$PMI_3(t,w) = \log_2 \frac{p(t,w)}{p(t) \cdot p(w)} ,$$

kde

$$p(t,w) = \frac{F_3(t,w)}{N} , \quad p(t) = \frac{\sum\limits_{w \in W} F_3(t,w)}{N} , \quad p(w) = \frac{\sum\limits_{t \in T} F_3(t,w)}{N} .$$

Tedy

$$PMI_3(t,w) = \log_2 \frac{N \cdot F_3(t,w)}{\sum\limits_{w \in W} F_3(t,w) \cdot \sum\limits_{t \in T} F_3(t,w)} .$$

Analogicky je definována také $PMI_4(t,w)$.

Seznam dvojic (t,w) s atributy $F_1, \ldots, F_4, PMI_3$ a PMI_4 je třeba dále pročistit, abychom získali co nejlepší *populaci sémantických typů*, což je seznam lexikálních jednotek realizujících jednotlivé sémantické typy.

Seznam dvojic i s jejich atributy je uložen ve formátu CSV a lze filtrovat, třídit a prohlížet pomocí scriptu `view_semtypes_lists.sh` (viz příloha E).

5.4 Čištění seznamů a evaluace

5.4.1 Anotace a testovací data

Ze seznamu dvojic (t,w) jsme vybrali 3 000 náhodných dvojic, které jsme předložili Patricku Hanksovi k ruční anotaci. Anotátorovi byly dány pouze samotné dvojice bez statistických charakteristik. Každé dvojici byla přiřazena jedna značka z množiny $\{T, C, M\}$ podle toho, jestli slovo w realizuje daný sémantický typ t.

- T (**t**ypical example) ... slovo typicky realizuje sémantický typ
  př.: Abstract – freedom, Animal – gophers, Disease – malaria
- C (possible **c**oercion) ... slovo není typickým reprezentantem, ale v nějakém kontextu lze použít s daným typem
  např. Beverage – cup (of tea)
- M (**m**istake) ... slovo nerealizuje sémantický typ
  př. Animal – reporter, Plane – gun

Celkem jsme získali 2 989 označkovaných dvojic (několik značek chybělo), z nichž bylo 1 139 (38,1 %) označeno jako typický příklad (T), 483 (16,2 %) jako „possible coercion“ (C) a zbylých 1 367 (45,7 %) jako chyby.

Na základě tohoto náhodně vybraného a ručně označkovaného vzorku bychom chtěli navrhnout vhodný filtr, který by ve výsledném seznamu dvojic zanechal pouze ty, kterým byla přiřazena značka T popř. T nebo C.

K sestavení filtru použijeme statistické charakteristiky dvojic. Vytvoříme jednoduchý filtr, který nejprve setřídí dvojice např. podle atributu F_3 a setříděný seznam uřízneme podle nastaveného prahu (treshold) atributu F_3. Následně

seznam setřídíme podle hodnot PMI_3 a opět uřízneme podle nastavené prahové hodnoty.

Nevýhodou je, že tímto způsobem roztřídíme dvojice pouze na dvě části. Při evaluaci musíme sjednotit značku C buď se značkou T nebo M.

Označme množinu všech dvojic, kterým byla při ruční anotaci přiřazena značka C jako množinu A_T, obdobně definujme A_C jako množinu dvojic, kterým byla přiřazena značka C. Dále označme množinou B ty dvojice, které filtr nevyřadil.

V případě, že sjednotíme značku C se značkou M, tak je precision P a recall R definován jako

$$P = \frac{|A_T \cap B|}{|B|}, \qquad R = \frac{|A_T \cap B|}{|A_T|}.$$

Pokud bychom sjednostili značku C se značkou T, pak bychom definovali precision a recall následovně:

$$P = \frac{|(A_T \cup A_C) \cap B|}{|B|}, \qquad R = \frac{|(A_T \cup A_C) \cap B|}{(|A_T \cup A_C|)}.$$

V našem experimentu jsme postupně nastavili prahovou hodnotu pro F_3 na 0, 1, 2, 3, 5 a pro prahovou hodnotu PMI_3 jsme postupně použili všechny možné hodnoty. Pro každou kombinaci jsme spočítali precision a recall a výsledek jsme zakreslili do PR grafu. Experiment jsme provedli pro případ, kdy sjednotíme C a M (viz obrázek 5.1) i pro sjednocení značek C a T (viz obrázek 5.2)

Celý experiment jsme opakovali, tentokrát s použitím atributů F_4 a PMI_4. Výsledné grafy se nacházejí na obrázcích 5.3 a 5.4.

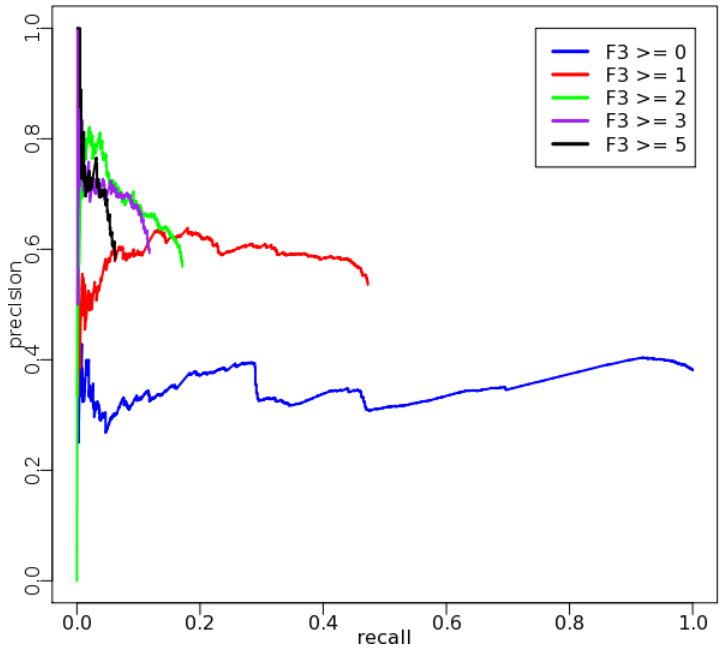
Martin Holub navrhnul a sestavil klasifikátor, který rozpoznává všechny tři značky s přesností cca 64 %. Klasifikátor využívá jednu z metod strojového učení – rozhodovací stromy. Je to pouze základní verze a je zde prostor pro další vylepšování. Klasifikátoru dělá problém rozpoznat právě kategorii C.

Příloze C lze nahlédnout populaci některých vybraných sémantických typů – *Plane*, *Animal*, *Weapon*, *Emotion* a *Beverage*. Hodnoty parametrů, se kterými byl vytvořen výsledný seznam, jsou uvedeny v popisku příslušné tabulky.

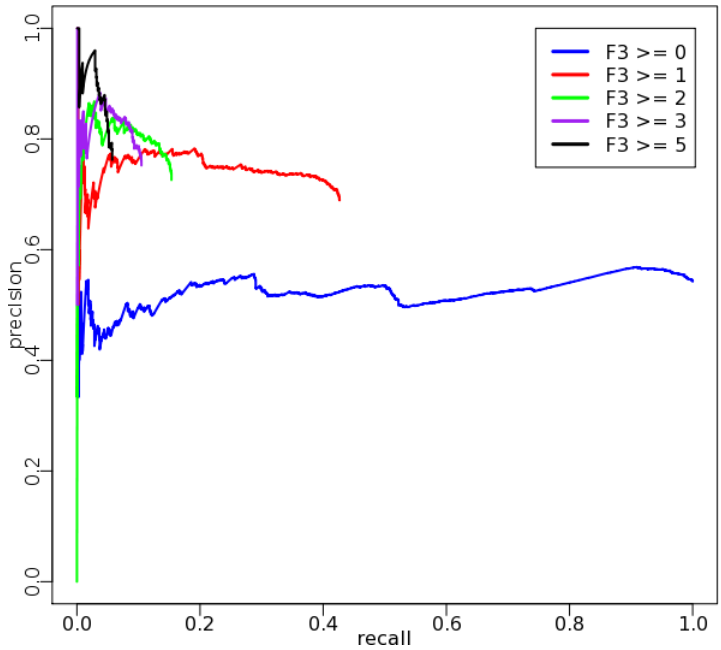
5.5 Další vývoj

Cílem dalšího vývoje je především zlepšení čištění seznamu dvojic. Máme přislíbena další ručně označkovaná data, celkově bychom měli mít 12 000 označkovaných dvojic, na základě kterých bychom chtěli vytvořit co nejlepší filtr.

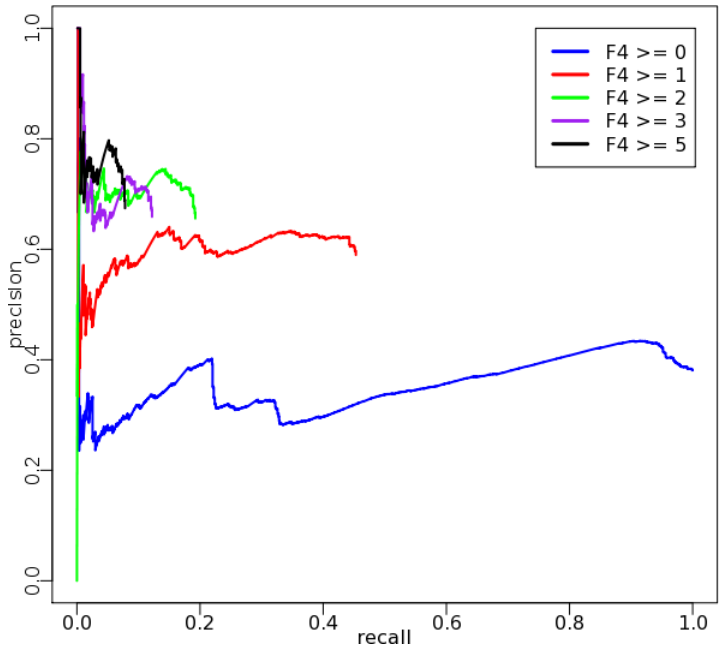
Víceznačnost v nalezených dvojicích by mohlo pomoci vyřešit využití informace získané NE klasifikátorem.



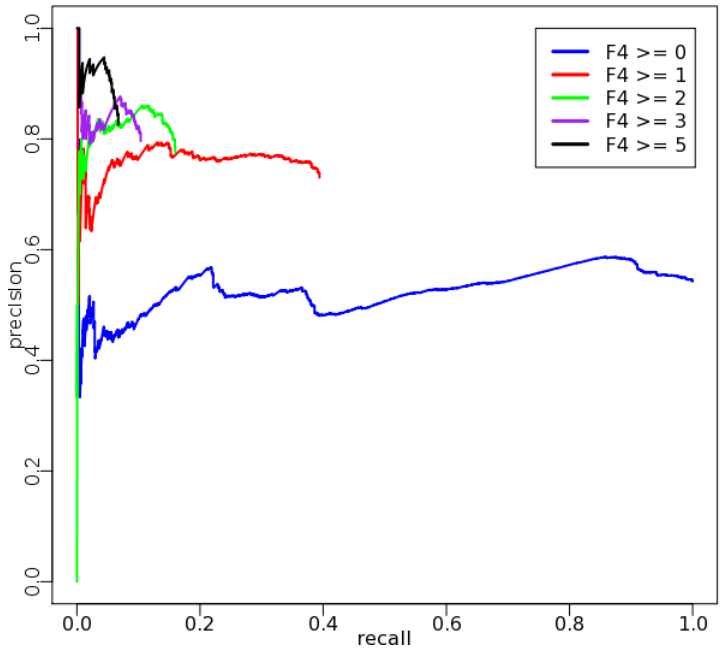
Obrázek 5.1: PR graf pro evaluaci filtru používajícího atributy F_3 a PMI_3, sjednocení značek C a M



Obrázek 5.2: PR graf pro evaluaci filtru používajícího atributy F_3 a PMI_3, sjednocení značek C a T



Obrázek 5.3: PR graf pro evaluaci filtru používajícího atributy F_4 a PMI_4, sjednocení značek C a M



Obrázek 5.4: PR graf pro evaluaci filtru používajícího atributy F_4 a PMI_4, sjednocení značek C a T

Kapitola 6

Automatické rozpoznávání patternů

6.1 Motivace

V současné verzi prohlížeče konkordancí slovníku PDEV je možné věty třídit podle pravého či levého kontextu nebo podle tvaru slovesa. Díky řazení podle kontextu lze označit více konkordancí jedním patternem, což může anotátorovi značně usnadnit práci.

Dosud však neexistuje nástroj – klasifikátor, který by slovesné patterny uměl přiřazovat. Nástroj by kromě návrhu patternu také uvedl *míru jistoty* (značíme c podle angl. *confidence*), s jakou daný pattern přiřazuje. To by pomohlo rozdělit množinu automaticky označkovaných konkordancí na ty, u kterých je vysoká pravděpodobnost, že jsou patterny přiřazeny správně (označme jako množinu H) a na ty ostatní (doplněk množiny H budeme značit H^C).

Existence takového nástroje by měla význam pro:

1. Ulehčení práce anotátorovi, protože automaticky označkované konkordance z množiny H by stačilo pouze zkontrolovat a zbývající (H^C) by ručně označkoval sám, případně by mohl uvažovat i navrhovaný pattern(y).

2. Díky zvětšování množiny označkovaných konkordancí bychom měli více dat pro populaci sémantických typů, což by hypoteticky vedlo k dalšímu zlepšování klasifikátoru. V tomto procesu bychom používali pouze konkordance označkované s vysokou mírou jistoty (tedy množinu H) a proces bychom mohli iterovat.

3. Dle předchozího bodu bychom měli co nejlepší nástroj, který by mohl být používán k automatickému přiřazování patternů v libovolných větách, což má význam např. pro strojový překlad.

V této kapitole navrhneme jednoduchý heuristický klasifikátor, který primárně přiřazuje pattern podle skóre vypočítaného na základě shody věty s jednotlivými patterny. V nerozhodných případech se použije apriorní pravděpodobnost odhadnutá z trénovacích dat.

Součástí je také evaluace navrženého klasifikátoru a další možnosti a návrhy pro jeho vylepšení.

6.2 Data

V tuto chvíli je v PDEV zkompilováno necelých 700 sloves pokrývajících cca 500 000 slovesných tokenů, z nichž je cca 200 000 označkovaných patterny. Přibližně polovinu z označkovaných konkordancí tvoří množina referenčních vzorků, což je náhodný výběr slovesných výskytů pro každé zkompilované sloveso o velikosti obvykle 250 výskytů (viz kapitola 2.1.2).

Z tohoto referenčního vzorku jsme odstranili věty se značkou x a také příliš dlouhé věty (> 100 tokenů). Zbylé věty jsme rozdělili na trénovací a testovací data v poměru $2:1$ pro každé sloveso. Trénovací data jsem používali při návrhu a ladění klasifikátoru, testovací data byla použita pro evaluaci klasifikátoru.

Při automatickém rozpoznávání patternů jsou použity populace sémantických typů získaných z celé množiny označkovaných dat (cca 200 000 vět). Není to úplně správné, protože k tomuto účelu by měla být použita trénovací data. Rozhodli jsme se však použít všechna dostupná označkovaná data, abychom získali co nejlepší populaci sémantických typů.

V PDEV je dalších cca 300 000 slovesných výskytů zkompilovaných sloves, které nejsou označkovány. Tuto množinu bychom mohli použít pro iterativní zlepšování populace sémantických typů a klasifikátoru, které bylo naznačené v bodu 2 v kapitole 6.1.

Použitá data jsou ze dne 12. 2. 2010, kdy bylo zkompilovaných 672 sloves. Pro evaluaci jsme vybrali 20 sloves, z nichž většina byla použita již pro měření mezianotátorské shody. Zbylá slovesa jsme vybrali dle jejich četnosti v BNC50 a preferovali jsme ta s vyšším počtem patternů. Velkým omezením bylo, že některé definice patternů nejsou konzistentní a je obtížné je strojově zpracovat (viz kapitola 3.1.1). Pro jednoduchost jsme se rozhodli s těmito slovesy nepracovat. Bohužel jsme tak přišli o velmi frekventovaná slovesa, jako jsou např. *say* a *tell*.

6.3 Heuristický klasifikátor

Úlohou klasifikátoru je správně přiřadit pattern slovesnému tokenu v daném kontextu. V trénovacích a testovacích datech je značkou buď číslo patternu případně doplněné písmenem e, nebo značka u. Doplňující značka e označuje výjimku, kterých je v datech omezené množství (6,41 % v množině všech referenčních vzorků) a značkou u jsou označeny konkordance, kterým nelze přiřadit žádný pattern. V referenčních vzorcích je jich celkem 2,15 %.

Vzhledem k tomu, že těchto značek není příliš mnoho a rozpoznání konkordancí, které by takto měly být označeny, by bylo příliš obtížné, bude klasifikátor přiřazovat pouze čísla patternů. Při testování se pak nebude považovat za chybu, pokud bude správně určeno číslo patternu, ale originální značka bude mít navíc e.

Klasifikátor má tento vstup:

- Populace sémantických typů – seznam dvojic (t, w), kde t je sémantický typ a w je slovo, které realizuje t.

  Uvedený seznam je ve formátu CSV včetně statistických charakteristik (atributů) $F_1, \ldots, F_4, PMI_3$ a PMI_4. Tento seznam může již být pročištěný, ale nemusí.

- Definice patternů – definice jsou ve formátu CSV, kde jedna řádka odpovídá jednomu patternu.

- Slovník pro převod plurálu na singulár – pokud použijeme slovník při sestavování populace sémantických typů (viz kap. 5.1.2), musíme slovník zapojit i v klasifikátoru, protože jinak by se množné tvary slov nenalezly v seznamu dvojic.

- Soubor apriorních pravděpodobností – pravděpodobnosti jsou odhadnuty na základě trénovacích dat.

  Označme v sloveso a r číslo jeho patternu. Pak pravděpodobnost s jakou se pattern r vyskytne u slovesa v odhadneme jako $p(r|v) = c(r, v)/c(v)$, kde $c(r, v)$ je počet výskytů slovesa v v trénovacích datech označených patternem r a $c(v)$ je celkový počet výskytů slovesa v v trénovacích datech, které byly označkovány určitým patternem, tedy ne značkou u.

- Věty s vyznačeným slovesným tokenem. Věty nejprve syntakticky analyzujeme pomocí Stanfordského parseru a ze stanfordských závislostí extrahujeme argumenty označeného slovesa. Použití NE klasifikátoru je volitelné, ale v případě, že byl použit při realizaci sémantických typů, je logické ho použít i při přiřazování patternů.

Výstupem klasifikátoru je číslo patternu, který byl automaticky přiřazen dané větě a míra jistoty, s jakou byl tento pattern přiřazen.

Klasifikátor každou větu porovnává se všemi patterny a pro každý z nich spočítá skóre na základě naplnění kolokačních pozic (agent, objekt, atd.). Celkové skóre patternu je součet součet všech skóre jednotlivých kolokačních pozic. Klasifikátor přiřadí ten pattern, který má nejvyšší skóre. Pokud je takových patternů více, rozhodne se mezi nimi na základě apriorní pravděpodobnosti a vybere se ten, který se v trénovacích datech vyskytoval nejčastěji.

Definujme základní ohodnocovací funkci $o_t(t, w)$, která každé dvojici (t, w) přiřadí skóre podle toho, jestli slovo w může realizovat t, kde t je buď sémantický typ nebo výčet lexikálních jednotek. Pokud je t sémantický typ, pak označme pomocnou proměnnou x výraz

$$x = \frac{F_3(t, w) \cdot PMI_3(t, w)}{\max\limits_{i,j}(F_3(t_i, w_j) \cdot PMI_3(t_i, w_j))}$$

a funkci $o_t(t, w)$ definujme jako

$$o_t(t, w) = \begin{cases} x\,, & x > 0 \\ 0\,, & jinak. \end{cases}$$

V případě, že t je zadán výčtem lexikálních jednotek, pak $o_t(t, w) = 1$, pokud je slovo w obsaženo v t, jinak $o_t(t, w) = 0$.

Oborem hodnot funkce $o_t(t, w)$ je $\langle 0, 1 \rangle$.

Jmenné entity mají zpravidla nízkou frekvenci a je velká pravděpodobnost, že při čištění seznamu sémantických typů budou odfiltrovány. Některá slova však mohla být rozpoznána NE klasifikátorem (pokud byl použit). V tom případě k takovému slovu w připojena značka NEP, NEO nebo NEL (viz kapitola 5.1.1). V případě, že je

- značka NEP a t je *Human*, pak $o_t(t, w) = 1$,
- značka NEO a t je *Institution*, pak $o_t(t, w) = 1$,
- značka NEL a t je *Location*, pak $o_t(t, w) = 1$.

Protože v seznamu slov, která realizují jednotlivé sémantické typy jsou pouze substantiva, přidali jsme navíc pravidla pro zájmena (mají POS tag PRP) a pro sémantické typy *Human* a *Self*. Pokud je t *Human* a $w \in \{$*I, he, she, you, we*$\}$, pak $o_t(t, w) = 1$. Podobně pokud slovo w je zakončeno *-self* nebo *-selves* a t je *Self*, pak $o_t(t, w) = 1$.

Tato funkce je implementována v proceduře `is_word_semtype` ve scriptu `rule_pattern_recognizer.pl`.

V jedné kolokační pozici může být více přípustných sémantických typů či výčtů lexikálních jednotek, stejně tak může být tato pozice obsazena více argumenty $\{w_1, \ldots, w_n\}$, které mají mezi sebou vztah koordinace či apozice.

Výpočet celkového skóre pro pozici s provedeme následovně:

$$o_s(s) = \frac{\sum_{i=1}^{n} \max_j o_t(t_j, w_i)}{n} .$$

Celkové skóre patternu r získáme jako součet všech skóre jednotlivých kolokačních pozic patternu r, tedy

$$o_p(r) = \sum_{i=1}^{n} o_s(s_i) ,$$

kde n je počet kolokačních pozic (slotů).

Po spočtení skóre všech patternů vybereme ten s nejvyšším ohodnocením. Pokud je takových patternů více, rozhodujeme se na základě apriorní pravděpodobnosti $p(r|v)$ a vybereme z těchto patternů ten, který byl v trénovacích datech označkován nejčastěji.

Určení míry jistoty při výběru patternu

Pokud je výběr patternů jednoznačný na základě výpočtu skóre, tak je míra jistoty (confidence) $c = 1$. Pokud nastane situace, že klasifikátor vybírá pattern na základě apriorní pravděpodobnosti z k kandidátů, spočítáme míru jistoty následovně:

$$c = \frac{\max_{i=1,\ldots,k} p(r_i|v)}{\sum_{i=1}^{k} p(r_i|v)} ,$$

kde $p(r_i|v)$ je apriorní pravděpodobnost patternu r_i pro sloveso v.

Míru jistoty bychom chtěli použít především pro rozdělení množiny automaticky označkovaných konkordancí na dvě disjunktní podmnožiny. U podmnožiny s vysokou mírou jistoty (označujeme jako H) by stačilo při ruční anotaci pouze zkontrolovat, jestli jsou patterny přiřazeny správně. Tato podmnožina by také mohla být použita pro následné iterativní zlepšování populace sémantických typů, což by vedlo k dalšímu zlepšování klasifikátoru.

6.4 Evaluace úspěšnosti přiřazení patternů

6.4.1 Metoda evaluace

Pro experimentální ověření účinnosti jsme vybrali 20 sloves. Některá z nich byla použita již při měření mezianotátorské shody. Další slovesa byla vybrána náhodně, preferována byla slovesa s vyšší četností v BNC50 a s vyšším počtem patternů. Bohužel nemohla být použita slovesa s vysokým počtem patternů (např. *call* – 34 nebo *tell* – 21), protože tato slovesa (a některá další) jsme museli vyloučit na základě toho, že v definicích patternů jsou nekonzistence, které znemožňují strojově zpracovat slovesné patterny.

Úspěšnost automatického přiřazování patternů určíme jako poměr správně přiřazených patternů ku velikosti testovací sady (zde nemá smysl mluvit o úplnosti, protože klasifikátor přiřadí pattern každému slovesnému tokenu). Množinu testovacích dat pro i-té sloveso označme D_i, množinu konkordancí, kterým byl správně přiřazen pattern, označme jako Y_i. Pak přesnost A_i (accuracy) přiřazení patternů slovesa i spočítáme jako

$$A_i = \frac{|Y_i|}{|D_i|}\,.$$

Průměrnou úspěšnost $\bar{A}$ spočítáme jako vážený průměr přesností A_i:

$$\bar{A} = \frac{\sum_{i=1}^{n} |D_i| \cdot A_i}{\sum_{i=1}^{n} |D_i|}$$

6.4.2 Průběh a výsledky evaluace

Jako baseline (základní triviální návrh) jsme použili přiřazování patternů pouze na základě apriorních pravděpodobností odhadnutých z trénovacích dat. Tedy jsme vždy přiřadili pattern, který se v trénovacích datech vyskytoval nejčastěji. Úspěšnost přiřazení správného patternu je uvedena ve sloupci A_B v tabulce 6.1. Průměrná úspěšnost této metody byla 61,54 %.

Klasifikátor jsme testovali nejprve s použitím nástroje na rozpoznání jmenných entit (NER), který byl použit také při vytváření seznamů sémantických typů – tedy seznamů dvojic (t, w), kde t je sémantický typ a w je slovo, které ho realizuje. Tento seznam jsme použili pročištěný s prahovými hodnotami $F_3 \geq 1$ a $PMI_3 \geq 0,5$.

sloveso	N	P	f	$\|D\|$	A_B	A	$\|Y\|$	$\bar{c}$	$\|H\|$	$\|H_Y\|$	A_H	A_1	A_2	A_3
abandon	8	4,22	2 813	77	54,55	76,62	59	82,61	54	49	90,74	76,62	76,62	77,92
abstain	3	2,47	232	74	58,11	71,62	53	76,88	26	21	80,77	71,62	71,62	74,32
address	6	3,85	3 628	80	63,75	72,50	58	83,40	49	39	79,59	77,50	73,75	73,75
alter	2	1,28	2 489	64	90,62	90,62	58	99,01	64	58	90,63	90,62	90,62	90,62
announce	4	2,68	8 739	82	48,78	80,49	66	86,18	59	52	88,14	80,49	80,49	80,49
argue	7	1,73	11 362	82	92,68	92,68	76	95,59	67	62	92,54	92,68	92,68	92,68
devote	6	4,65	1 530	80	47,50	63,75	51	80,95	45	38	84,44	66,25	65,00	62,50
engage	9	4,82	2 985	79	37,97	44,30	35	62,90	20	9	45,00	44,30	44,30	40,51
fire	15	7,96	1 488	76	28,95	40,79	31	61,95	27	15	55,56	42,11	42,11	40,79
handle	6	3,35	2 796	80	56,25	66,25	53	87,49	55	41	74,55	66,25	66,25	70,00
land	12	7,02	1 496	66	40,91	59,09	39	67,09	27	18	66,67	63,64	65,15	60,61
need	5	4,12	28 352	65	35,38	53,85	35	76,37	31	21	67,74	56,92	56,92	53,85
plan	4	3,08	7 294	59	54,24	81,36	48	77,55	35	31	88,57	81,36	81,36	81,36
propose	8	3,80	4 987	81	65,43	76,54	62	81,43	50	38	76,00	77,78	76,54	75,31
rush	9	4,99	984	277	53,07	65,34	181	71,14	113	69	61,06	66,43	66,79	63,54
signal	6	3,76	892	73	57,53	63,01	46	83,87	39	24	61,54	63,01	63,01	61,64
sleep	11	2,44	1 575	414	77,78	84,78	351	83,05	90	79	87,78	85,27	85,27	83,82
treat	4	2,60	6 455	81	71,60	81,48	66	82,27	35	25	71,43	83,95	83,95	81,48
urge	6	2,84	2 603	83	72,29	81,93	68	96,37	75	65	86,67	81,93	81,93	85,54
visit	3	2,16	5 889	76	68,42	75,00	57	88,22	55	48	87,27	73,68	71,05	73,68
celkem				2049	61,54	72,86	1493		1016	802	78,94	73,79	73,55	72,62

Tabulka 6.1: Úspěšnost automatického přiřazování patternů: N – počet patternů slovesa, P – perplexita patternů slovesa, f – frekvence slovesa v BNC50, $|D|$ – velikost testovaného vzorku, A_B – úspěšnost baseline návrhu, A – úspěšnost klasifikátoru, $|Y|$ – počet správně označených vět, $\bar{c}$ – průměrná míra jistoty, $|H|$ – počet patternů přiřazených s mírou jistoty $\geq 0,9$, $|H_Y|$ – počet správně přiřazených patternů s velkou mírou jistoty , A_H – $A_H = |H_Y|/|H|$, A_1 – úspěšnost experimentu 1 , A_2 – úspěšnost experimentu 2, A_3 – úspěšnost experimentu 3

Ve sloupci A tabulky 6.1 je uvedena účinnost popsaného klasifikátoru (viz kap. 6.3). U většiny sloves se úspěšnost oproti baseline (sloupec A_B) zvýšila, pouze u sloves *alter* a *argue* jsou hodnoty shodné, u žádného slovesa nenastalo zhoršení. Průměrná úspěšnost klasifikátoru je 72,86 %.

Ve sloupci $|Y|$ je počet správně přiřazených patternů v testovacích datech, ve sloupci $\bar{c}$ je aritmetický průměr míry konfidence, s jakou byly určeny jednotlivé patterny. V dalším sloupci ($|H|$) je počet konkordancí označených s velkou mírou jistoty (zde byl nastaven práh $c \geq 0,9$). Ve sloupci $|H_Y|$ je počet správně přiřazených patternů, u kterých byla velká míra jistoty přiřazení. Ve sloupci A_H je uvedena úspěšnost na podmnožině H množiny testovacích dat.

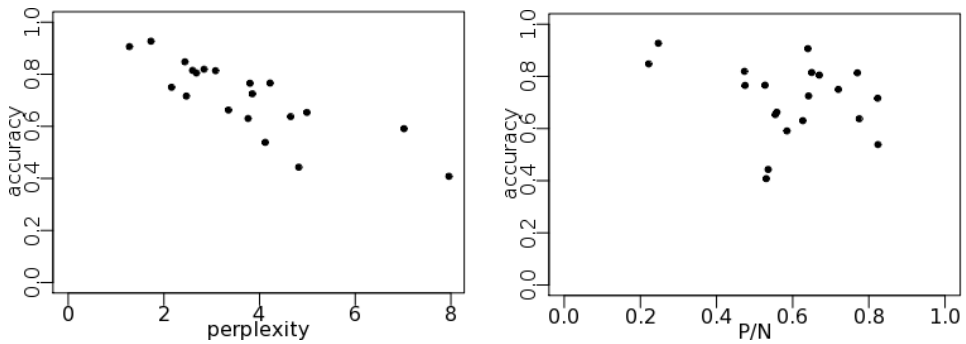
V následujících sloupcích A_1, A_2 a A_3 jsou uvedeny výsledky tří experimentů:

1. NER nebyl použit ani v klasifikátoru pro přiřazení patternu, ani při vytváření populace sémantických typů.

2. NER jsme použili pouze při vytváření populace sémantických typů, v klasifikátoru pro přiřazení patternu použit nebyl.

3. Místo pročištěného seznamu dvojic (t, w) jsme použili úplný seznam těchto dvojic.

Zkoumali jsme také, zda úspěšnost klasifikátoru nějakým způsobem souvisí s perplexitou patternů slovesa, s počtem patternů či s mezianotátorskou shodou. Nejvýznamnější závislost se ukázala mezi úspěšností klasifikátoru a perplexitou patternů. Vyšší úspěšnost je u sloves, u kterých je perplexita patternů nižší. Korelace je znázorněna na obrázku 6.1 a.



Obrázek 6.1: korelace úspěšnosti (accuracy) automatického přiřazení patternu slovesnému tokenu a a) perplexity b) poměru perplexity ku počtu patternů (P/N)

Perplexitu jsme spočítali na základě celého referenčního vzorku, tedy sjednocení trénovacích i testovacích dat. Maximální hodnota perplexity slovesa může být rovna počtu jeho patternů. To nastává právě tehdy, když patterny mají rovnoměrné rozdělení v datech. Poměr perplexity slovesa ku počtu jeho patternů (označme $d = P/N$) je hodnota z intervalu $(0, 1\rangle$. Tato hodnota by teoreticky

mohla pro každé sloveso udávat *obtížnost* přiřazování patternů. Pak by se dalo očekávat, že úspěšnost klasifikátoru bude menší na „obtížnějších“ slovesech. Korelace úspěšnosti a hodnoty d (obr. 6.1 b) je však slabší (Pearsonův korelační koeficient $k = -0.27$) než korelace úspěšnosti a perplexity ($k = -0.82$).

6.5 Další vývoj

Problematika automatického rozpoznávání patternů nabízí mnohá vylepšení. Navržený pravidlový klasifikátor by mohl být vylepšen v mnoha směrech.

- efektivnější využití pomocných nástrojů (parsing, NER)
- populace sémantických typů
- lepší využití dostupných informací
- vylepšení samotného klasifikátoru

Efektivnější využití pomocných nástrojů

Před samotnou úlohou přiřazování patternů bylo třeba data syntakticky zanalyzovat a následně zpracovat výstup použitého parseru. Nabízí se zde možnost použití úspěšnějšího parseru nebo vylepšení následného postprocesingu (viz kapitola 4.4).

Další oblastí možného zlepšení je použití lepšího slovníku popř. lemmatizéru, jak již bylo zmíněno v kapitole 5.1.2.

Použití NE klasifikátoru s modelem pro rozpoznávání více typů jmenných entit by mohlo pomoci automaticky rozpoznat více sémantických typů.

Populace sémantických typů

Při evaluaci byl použitý seznam sémantických typů pročištěný s prahovou hodnotou proměnné $F_3 \geq 1$ a $PMI_3 \geq 0,5$. Při dokonalejším čištění seznamů sémantických typů nebo při lepším propojení lexikálních jednotek se sémantickými typy, které realizují, by se dalo dosáhnout větší úspěšnosti.

Lepší využití dostupných informací

Z definice patternu nejsou využívány všechny dostupné informace. Např. by mohlo být užitečné pracovat s formou slovesa uvedenou v definici patternu, protože některé patterny vyžadují sloveso v pasivní formě, čímž by se vyloučila možnost přiřadit tento pattern pro aktivní věty. Tato informace nebyla v současné verzi využita, protože kódování v patternech nebylo konzistentní.

Věříme, že vznik podrobné dokumentace k vytváření a kódovaní patternů (viz kap. 3.4) bude mít za důsledek konzistentní zápis patternů. Současně se má změnit také XML struktura, ke které rovněž vznikne podrobná dokumentace. Díky tomu by se dalo snadněji využít více informací zakódovaných v patternu.

Vylepšení samotného klasifikátoru

Vylepšit lze i samotný klasifikátor. Současná verze přiřazuje právě jeden pattern, který má nejvyšší skóre. Může se však stát, že jiný pattern bude mít skoro stejně vysoké skóre. Pak by měl klasifikátor navrhnout oba patterny resp. normalizovaný vektor $q_1, , \ldots, q_n$, kde n je počet patternů slovesa a hodnoty q_i jsou skóre pro patterny slovesa.

Při výběru patternu nejsou uvažovány všechny informace, které jsou k dispozici. Např. se zatím vůbec nepracuje se sémantickými rolemi nebo s množinou lexikálních jednotek, pokud je zároveň uveden sémantický typ. V této verzi se nepracuje ani s hierarchií sémantických typů, což by rovněž mohlo zvýšit celkovou úspěšnost.

Apriorní pravděpodobnost by se měla využít vždy a nejen pouze při rozhodování se mezi patterny se shodným skóre. Klasifikátor by také mohl využívat metody strojového učení.

Současný klasifikátor by mohl být použit k automatickému přiřazení patternů dosud neoznačkovaným konkordancím zkompilovaných sloves, kterých je cca 300 000. Po nastavení vhodné prahové hodnoty pro míru jistoty odhadnuté na základě testovacích dat bychom oddělili ty konkordance, kde je míra jistoty vyšší než prahová hodnota a z těchto vět bychom získali další slova realizující sémantické typy (viz kapitola 5), což by vedlo k vylepšení populace sémantických typů. Díky zlepšení populace sémantických typů bychom dosáhli následného vylepšení klasifikátoru. Po několika opakováních tohoto procesu by se úspěšnost klasifikátoru ustálila.

6.6 Shrnutí

V této kapitole jsme navrhli a otestovali nástroj pro automatické přiřazování patternů. Úspěšnost tohoto nástroje – klasifikátoru je 72,86 %. Na jeho vylepšení se bude dále pracovat. Klasifikátor by měl nejen přispět k rozšiřování množiny označkovaných konkordancí, ale díky němu je možné přiřadit pattern slovesu v jakékoliv anglické větě, což může být užitečné zejména ve strojovém překladu [17].

Kapitola 7

Závěr

V předložené práci jsme se zabývali projektem PDEV a jeho možném využití v NLP aplikacích. Projekt jsme popsali po myšlenkové i technické stránce a srovnali ho s podobnými existujícími projekty pro angličtinu i pro češtinu (viz kapitola 2).

Dále jsme popsali současný stav projektu, včetně výsledků experimentu měření mezianotátorské shody (kapitola 3.3), jejíž dobrý výsledek je jedním z hlavních ukazatelů možnosti automatického využití PDEV v NLP aplikacích.

Podrobněji jsme se zaměřili na automatickou syntaktickou anlýzu angličtiny a provedli jsme experimentální ověření účinnosti použitého Stanfordského parseru včetně srovnání s dalšími parsery a metodami (viz kapitola 4).

V kapitole 5 jsme se zabývali vytvořením seznamů lexikálních jednotek realizujících jednotlivé sémantické typy na základě dostupných označkovaných konkordancí. Výsledné pročištěné seznamy byly použity v kapitole 6 při návrhu klasifikátoru pro přiřazení patternu. V této kapitole je rovněž uvedeno experimentální ověření úspěšnosti navrženého klasifikátoru. Klasifikátor by měl přispět k rozšíření množiny označkovaných konkordancí.

Můžeme říci, že tato práce v nějaké míře splňuje všechny body zadání. Podrobněji je zpracována kapitola 4, mnohem více však mohly být propracovány kapitoly 5 a 6. V těchto kapitolách je alespoň naznačen směr dalšího možného vývoje.

Některé výsledky resp. části této práce byly využity pro přípravu publikací: [31] (publikováno jako kapitola v knize), [16] (přijato k publikaci ve sborníku konference TSD 2010) a [30] (zasláno jako příspěvek do konference EMNLP 2010).

Věříme, že práce bude mít přínos pro další vývoj projektu PDEV a také bude užitečnou zpětnou vazbou pro autora PDEV. Během řešení práce jsme se potýkali s několika problémy jako je např. chybějící dokumentace k definicím patternů a nekonzistence jejich kódování. Díky těmto pozorováním jsme mohli dát podmět k jejich vylepšení či odstranění.

Dodatek A

Seznam použitých zkratek

BNC – British National Corpus

CPA – Corpus Pattern Analysis

FGD – Funkční generativní popis

PDEV – Pattern Dictionary of English Verbs

PEDT – Prague English Dependency Treebank

PDT – Prague Dependency Treebank

PTB – Penn Treebank

WSJ – Wall Street Journal

Dodatek B

Definice patternů

Popis polí v CSV formátu, oddělovačem je středník:

1. *indikátor, zda je pattern dobře definován* – pomocná položka; pokud je zde 1, je pattern v pořádku a může být strojově zpracován; v opačném případě je zde 0
2. *počet slotů pro sémantické typy* – pomocná položka obsahující počet pozic, které mohou být obsazeny sémantickým typem, tedy kolik dvojic (typ, slovo) lze v ideálním případě z věty získat
3. *slovesné lemma* – základní forma slovesa, podle které lze vyhledávat v Entry Manageru (sloupec Entry)
4. *počet patternů slovesa*
5. *slovesná forma* – většinou je stejná jako lemma, ale může být zde pasivum či negace
6. *číslo patternu*
7. *agent* – má tuto strukturu:
   `Sémantický_typ[Sémantická_role]:množina_lexikálních_jednotek`
   Prvky v množině lexikálních jednotek jsou od sebe odděleny čárkami stejně jako jednotlivé sémantické role. Celá struktura se může v tomto poli několikrát opakovat, oddělovačem je zde znak |.
8. *objekt* – stejně jako agent
9. *nepřímý objekt* – stejně jako agent a objekt
10. *komplement*
11. *objektové klauze* – přípustné klauze jsou označeny zkratkou, např. *that* pro that-clause, *to* pro konstrukci to+inf
12. *klauze* – stejné jako *objektové klauze*
13. *příslovečná určení (adverbial)* – pokud se jedná o předložkovou frázi, je nejprve definována množina předložek (označeno jako „hw“) a předložkový objekt je definován stejně jako agent

```
<pattern id="3" num="2">
   <template adverbial_class="" object_none="" adverbial_none="" auxiliary="">
     <verb_form>devote</verb_form>
     <subject>
       <argspec none="" headword="" type="">
         <BSO_type name="Human"/>
         <subspec>
           <Role name="Author"/>
         </subspec>
       </argspec>
       <alternation>
         <argspec headword="">
           <BSO_type name="Institution"/>
           <subspec>
             <Role name="Newspaper | Journal"/>
           </subspec>
         </argspec>
       </alternation>
     </subject>
     <object>
       <argspec none="" headword="" type="">
         <BSO_type name="Document Part"/>
         <subspec>
           <Lexset name="">
             <item>page| chapter| space</item>
           </Lexset>
         </subspec>
       </argspec>
       <alternation>
         <argspec headword="">
           <BSO_type name="Document"/>
           <subspec/>
         </argspec>
       </alternation>
     </object>
     <adverbial optional="false" func="">
       <alternation type="">
         <ptag headword="to" value="PP">
           <argspec>
             <adverbs/>
             <BSO_type name="Anything"/>
             <subspec>
               <Role name="Topic"/>
             </subspec>
           </argspec>
         </ptag>
       </alternation>
     </adverbial>
     <clausal/>
     <oclausals ing="" to="" wh="" quote="" that=""/>
     <clausals ing="" optional="" to="" wh="" quote="" that=""/>
     <comment type=""/>
   </template>
   <primary_implicature idiom="" phrasal="">
     [[{Human = Author} | {Institution = Newspaper | Journal} | Document]]
     gives priority to discussion or elaboration of [[Anything = Topic]]
     in [[Document Part | Document]]
   </primary_implicature>
   <framenet>NOT YET IN. 12 January 2010.</framenet>
   <domain/>
   <register/>
   <comment type=""/>
   <exploitation/>
 </pattern>
```


Obrázek B.1: Definice patternu č. 2 slovesa *devote* v XML souboru

```
$VAR1 = {
   'verb_form' => 'devote',
   'object' => [
      {
         'roles' => [],
         'lexset' => [ 'page', 'chapter', 'space' ],
         'bso_type_orig' => 'Document Part',
         'bso_type' => 'Document Part'
      },
      {
         'roles' => [],
         'lexset' => [],
         'bso_type_orig' => 'Document',
         'bso_type' => 'Document'
      }
   ],
   'subject' => [
      {
         'roles' => [ 'Author' ],
         'lexset' => [],
         'bso_type_orig' => 'Human',
         'bso_type' => 'Human'
      },
      {
         'roles' => [ 'Newspaper', 'Journal' ],
         'lexset' => [],
         'bso_type_orig' => 'Institution',
         'bso_type' => 'Institution'
      }
   ],
   'clausals' => [],
   'oclausals' => [],
   'is_ok' => '1',
   'pattern_num' => '2',
   'verb_stem' => 'devote',
   'adverbials' => [
      {
         'alternations' => [
         {
            'type' => 'ptag',
            'pobject' => {
               'roles' => [ 'Topic' ],
               'lexset' => [],
               'bso_type_orig' => 'Anything',
               'bso_type' => 'Anything'
            },
            'headword' => [ 'to' ]
         }
         ],
         'opt' => 'false'
      }
   ],
   'comp' => '',
   'patterns_count' => '6',
   'iobject' => [],
   'semtypes_count' => '3'
};
```

Obrázek B.2: Vnitřní reprezentace patternu č. 2 slovesa *devote*

```
1;2;abandon;8;abandon;1;Human|Institution;Activity|Plan;;;;;
1;2;abandon;8;abandon;2;Human|Institution;Attitude;;;;;
1;2;abandon;8;abandon;3;Human|Human Group[Military];Location;;;;;
1;2;abandon;8;abandon;4;Human;Artifact;;;;;
1;1;abandon;8;abandon;5;Human;EMPTY:ship;;;;;
1;3;abandon;8;abandon;6;Human 1|Animal 1;Human 2|Animal 2;;;;;
   adv,opt=true,hw(to)Anything[Bad];
1;3;abandon;8;abandon;7;Human;Self;;;;;adv,opt=false,hw(to)Activity|
   hw(to)Attitude;
1;3;abandon;8;abandon;8;Human;Self;;;;;adv,opt=false,hw(to)Deity;

1;2;abstain;3;abstain;1;Human;;;;;;adv,opt=true,hw(from)Activity;
1;2;abstain;3;abstain;2;Human;;;;;;adv,opt=true,hw(from)Alcoholic
   Drink|hw(from)Drug|hw(from)Food;
1;1;abstain;3;abstain;3;Human[Delegate]|Institution;;;;;;adv,opt=true,
   hw(from)EMPTY:vote|hw(in)EMPTY:vote;

1;3;devote;6;devote;1;Human|Institution;Resource:time,money,effort;;;;;
   adv,opt=false,hw(to)Activity;
1;3;devote;6;devote;2;Human[Author]|Institution[Newspaper,Journal];
   Document Part:page,chapter,space|Document;;;;;
   adv,opt=false,hw(to)Anything[Topic];
1;3;devote;6;devote;3;Human;Thought;;;;;adv,opt=false,hw(to)Activity|
   hw(to)Entity;
1;3;devote;6;devote;4;Human|Institution;Activity;;;;;adv,opt=false,
   hw(to)Anything;
1;3;devote;6;devote;5;Human|Institution;Building|Building Part|Container
   ;;;;;adv,opt=false,hw(to)Anything;
1;3;devote;6;devote;6;Human;Self;;;;;adv,opt=false,hw(to)Activity;

1;2;propose;8;propose;1;Human|Institution;Action|Plan|Proposition;;;;;
1;2;propose;8;propose;2;Human|Institution;Entity;;;;;
1;1;propose;8;propose;3;Human;;;;;that;
1;1;propose;8;propose;4;Human|Institution;;;;;to;
1;3;propose;8;propose;5;Human 1;Human 2;;;;;adv,opt=false,hw(for,as)Role;
1;2;propose;8;propose;6;Human[Male];;;;;;adv,opt=false,hw(to)Human 2[Female];
1;1;propose;8;propose;7;Human 1;EMPTY:marriage;;;;;
1;1;propose;8;propose;8;Human 1|Institution 1;EMPTY:toast,health;;;;;

1;2;urge;6;urge;1;Human 1|Institution 1;Human 2|Institution 2;;;;to;
1;2;urge;6;urge;2;Human|Institution;Action|Attitude;;;;;
1;1;urge;6;urge;3;Human|Institution;;;;;that;
1;2;urge;6;urge;4;Human 1;Human 2;;;;quote;
1;2;urge;6;urge;5;Human 1;Human 2;;;;;adv,opt=false,hw(on)EMPTY;
1;2;urge;6;urge;6;Human[Rider];Horse;;;;;

1;2;visit;3;visit;1;Human|Human Group;Location;;;;;
1;2;visit;3;visit;2;Human 1;Human 2;;;;;
1;2;visit;3;visit;3;Human;Location[Educational]:exhibition,conference;;;;;
```

Obrázek B.3: Definice patternů vybraných sloves v CSV souboru

Dodatek C

Populace sémantických typů

noun	semtype	F_1	F_2	F_3	PMI_3	F_4	PMI_4
aircraft	Plane	36,00	20,00	27,83	8,28	34,43	8,76
plane	Plane	21,00	12,00	16,50	8,68	20,31	9,15
helicopter	Plane	21,00	2,00	11,25	8,09	14,67	8,64
P-40	Plane	3,00	3,00	3,00	9,59	3,00	9,76
aeroplane	Plane	3,00	2,00	2,50	8,59	3,00	9,03
caravan	Plane	5,00	0,00	2,50	8,33	2,50	8,50
van	Plane	5,00	0,00	2,50	6,52	2,50	6,69
vehicle	Plane	5,00	0,00	2,50	6,06	2,50	6,23
Hurricanes	Plane	2,00	2,00	2,00	7,42	2,00	7,59
Sugar	Plane	2,00	2,00	2,00	8,59	2,00	8,76
glider	Plane	2,00	2,00	2,00	9,59	2,00	9,76
jet	Plane	3,00	1,00	2,00	7,42	3,00	8,18
night	Plane	3,00	1,00	2,00	2,77	3,00	3,53
number	Plane	3,00	1,00	2,00	2,11	1,51	1,87
prototype	Plane	2,00	2,00	2,00	8,01	2,00	8,18
spacecraft	Plane	2,00	2,00	2,00	9,01	2,00	9,18

Tabulka C.1: Populace sémantického typu *Plane*; prahové hodnoty: $F_3 \geq 2$, $PMI_3 \geq 0,5$, $PMI_4 \geq 0$

noun	semtype	F_1	F_2	F_3	PMI_3	F_4	PMI_4
animal	Animal	75,00	2,00	28,42	4,03	16,60	4,84
dog	Animal	40,00	1,00	18,25	3,79	5,03	3,51
cat	Animal	25,00	2,00	12,58	4,51	9,67	5,71
bird	Animal	24,00	0,00	9,67	3,11	0,50	0,42
horse	Animal	13,00	5,00	8,58	3,95	9,00	5,60
fish	Animal	19,00	2,00	8,33	3,22	8,00	4,74
predator	Animal	16,00	1,00	6,67	4,46	6,67	6,04
cattle	Animal	15,00	0,00	6,17	4,35	3,00	4,89
owner	Animal	11,00	0,00	4,08	1,52	0,50	0,07
bull	Animal	6,00	2,00	4,00	4,24	4,20	5,90
male	Animal	10,00	0,00	4,00	3,54	0,50	2,13
blood	Animal	6,00	2,00	3,67	2,60	5,33	4,72
condition	Animal	14,00	0,00	3,50	1,12	3,50	2,70
sheep	Animal	8,00	0,00	3,42	3,93	1,00	3,74
species	Animal	9,00	0,00	3,42	3,11	0,50	1,92
beetle	Animal	7,00	0,00	3,00	4,83	3,00	6,41
mouse	Animal	5,00	1,00	3,00	3,74	2,33	4,96
prey	Animal	6,00	1,00	3,00	3,24	1,71	4,02
owl	Animal	8,00	0,00	2,83	4,58	2,83	6,16
monkey	Animal	6,00	1,00	2,75	4,53	3,50	6,46
type	Animal	8,00	0,00	2,75	1,16	0,50	0,29
plant	Animal	6,00	0,00	2,50	1,21	0,50	0,47
rat	Animal	4,00	1,00	2,50	4,57	4,00	6,83
shark	Animal	6,00	0,00	2,33	4,88	2,33	6,46
train	Animal	5,00	1,00	2,25	2,56	5,00	5,29
thought	Animal	5,00	1,00	2,17	1,69	1,73	2,95
chick	Animal	4,00	1,00	2,08	3,06	2,50	4,90
North_Sea	Animal	2,00	2,00	2,00	5,66	2,00	7,24
bear	Animal	3,00	1,00	2,00	4,24	3,00	6,41
beast	Animal	3,00	1,00	2,00	3,92	1,50	5,09
camel	Animal	2,00	2,00	2,00	5,24	2,00	6,83
goat	Animal	4,00	0,00	2,00	4,66	0,50	4,24
gull	Animal	2,00	2,00	2,00	4,66	2,00	6,24
lizard	Animal	4,00	0,00	2,00	4,92	2,00	6,50
oxen	Animal	2,00	2,00	2,00	4,44	2,00	6,02
while	Animal	4,00	0,00	2,00	2,72	0,50	2,30

Tabulka C.2: Populace sémantického typu *Animal*; prahové hodnoty: $F_3 \geq 2$, $PMI_3 \geq 0,5$, $PMI_4 \geq 0$

noun	semtype	F_1	F_2	F_3	PMI_3	F_4	PMI_4
weapon	Weapon	23,00	23,00	23,00	7,99	23,00	8,00
gun	Weapon	18,00	18,00	18,00	7,15	18,00	7,16
knife	Weapon	17,00	17,00	17,00	8,83	17,00	8,84
spear	Weapon	11,00	11,00	11,00	9,20	11,00	9,21
rifle	Weapon	9,00	9,00	9,00	7,91	9,00	7,92
sword	Weapon	9,00	9,00	9,00	9,17	9,00	9,18
missile	Weapon	9,00	8,00	8,50	6,32	9,00	6,42
club	Weapon	7,00	7,00	7,00	6,67	7,00	6,68
pistol	Weapon	7,00	7,00	7,00	8,04	7,00	8,05
handgun	Weapon	5,00	5,00	5,00	8,84	5,00	8,85
stone	Weapon	5,00	5,00	5,00	4,88	5,00	4,89
truncheon	Weapon	5,00	5,00	5,00	9,32	5,00	9,33
article	Weapon	4,00	4,00	4,00	5,30	4,00	5,31
bomb	Weapon	4,00	4,00	4,00	5,65	4,00	5,66
bow	Weapon	4,00	4,00	4,00	9,32	4,00	9,33
shield	Weapon	4,00	4,00	4,00	8,52	4,00	8,53
shotgun	Weapon	4,00	4,00	4,00	8,15	4,00	8,16
stick	Weapon	4,00	4,00	4,00	7,62	4,00	7,63
artillery	Weapon	3,00	3,00	3,00	7,59	3,00	7,60
bat	Weapon	3,00	3,00	3,00	8,10	3,00	8,11
cannon	Weapon	3,00	3,00	3,00	8,10	3,00	8,11
firearm	Weapon	3,00	3,00	3,00	9,32	3,00	9,33
grenade	Weapon	3,00	3,00	3,00	6,74	3,00	6,75
machete	Weapon	3,00	3,00	3,00	9,32	3,00	9,33
screwdriver	Weapon	3,00	3,00	3,00	8,59	3,00	8,60
sticks	Weapon	3,00	3,00	3,00	8,32	3,00	8,33
axis	Weapon	2,00	2,00	2,00	8,32	2,00	8,33
baton	Weapon	2,00	2,00	2,00	8,32	2,00	8,33
belt	Weapon	2,00	2,00	2,00	7,74	2,00	7,75
explosive	Weapon	2,00	2,00	2,00	8,74	2,00	8,75
musket	Weapon	2,00	2,00	2,00	8,74	2,00	8,75
razor	Weapon	2,00	2,00	2,00	8,32	2,00	8,33
rock	Weapon	2,00	2,00	2,00	5,04	2,00	5,05
rocket	Weapon	2,00	2,00	2,00	5,47	2,00	5,48
scythe	Weapon	2,00	2,00	2,00	9,32	2,00	9,33

Tabulka C.3: Populace sémantického typu *Weapon*; prahové hodnoty: $F_3 \geq 2$, $PMI_3 \geq 0,5$, $PMI_4 \geq 0$

noun	semtype	F_1	F_2	F_3	PMI_3	F_4	PMI_4
fear	Emotion	87,00	72,00	79,08	7,00	86,93	6,94
interest	Emotion	109,00	0,00	52,50	5,25	50,00	4,99
suspicion	Emotion	67,00	13,00	40,00	6,77	67,00	7,32
feeling	Emotion	41,00	11,00	25,58	5,60	40,67	6,07
concern	Emotion	35,00	16,00	25,25	5,62	34,84	5,89
opposition	Emotion	46,00	1,00	23,17	5,44	44,29	6,18
anger	Emotion	27,00	17,00	22,00	6,89	27,00	6,99
anxiety	Emotion	27,00	11,00	19,00	6,68	27,00	6,99
conscience	Emotion	18,00	14,00	15,67	6,78	17,65	6,76
hostility	Emotion	29,00	2,00	15,50	6,27	29,00	6,99
controversy	Emotion	25,00	1,00	13,00	5,92	25,00	6,67
emotion	Emotion	23,00	2,00	12,00	5,94	23,00	6,69
criticism	Emotion	16,00	6,00	11,00	5,50	16,00	5,85
spleen	Emotion	11,00	11,00	11,00	7,71	11,00	7,52
pride	Emotion	10,00	10,00	10,00	6,79	10,00	6,60
guilt	Emotion	13,00	7,00	9,83	6,92	12,88	7,12
curiosity	Emotion	20,00	0,00	9,50	6,32	9,50	6,13
frustration	Emotion	8,00	8,00	8,00	6,13	8,00	5,94
doubt	Emotion	9,00	5,00	7,00	5,43	9,00	5,60
passion	Emotion	14,00	0,00	6,83	5,84	6,83	5,65
resentment	Emotion	12,00	1,00	6,50	6,09	6,50	5,90
rage	Emotion	6,00	6,00	6,00	7,49	6,00	7,30
deal	Emotion	12,00	0,00	5,75	3,70	5,75	3,51
wrath	Emotion	10,00	2,00	5,67	6,76	10,00	7,38
indignation	Emotion	11,00	0,00	5,50	6,27	5,50	6,07
protest	Emotion	11,00	0,00	5,50	4,85	5,50	4,66
sense	Emotion	9,00	2,00	5,08	3,68	6,84	3,92
enthusiasm	Emotion	10,00	0,00	5,00	5,08	5,00	4,89
hope	Emotion	8,00	2,00	5,00	4,03	2,46	2,82
expectation	Emotion	10,00	0,00	4,75	5,15	4,00	4,72
opinion	Emotion	6,00	3,00	4,17	4,13	5,50	4,34
fury	Emotion	6,00	2,00	4,00	6,71	4,67	6,74
sympathy	Emotion	8,00	0,00	4,00	5,25	4,00	5,06
desire	Emotion	8,00	0,00	3,75	4,13	3,25	3,73
response	Emotion	8,00	0,00	3,75	3,45	3,50	3,16
hatred	Emotion	5,00	2,00	3,50	4,76	5,00	5,09
tension	Emotion	5,00	2,00	3,50	4,88	5,00	5,20
trouble	Emotion	4,00	3,00	3,50	5,20	3,75	5,11

Tabulka C.4: Populace sémantického typu *Emotion*; prahové hodnoty: $F_3 \geq 3,5$, $PMI_3 \geq 0,5$, $PMI_4 \geq 0$

noun	semtype	F_1	F_2	F_3	PMI_3	F_4	PMI_4
coffee	Beverage	44,00	11,00	27,50	8,37	38,92	8,86
tea	Beverage	46,00	8,00	27,00	8,15	29,94	8,29
beer	Beverage	31,00	1,00	16,00	8,06	11,00	7,51
wine	Beverage	24,00	3,00	13,50	8,01	13,50	8,00
pint	Beverage	22,00	3,00	12,50	8,13	17,25	8,59
alcohol	Beverage	23,00	0,00	11,50	7,70	11,50	7,69
cup	Beverage	20,00	2,00	11,00	7,95	20,00	8,80
glass	Beverage	17,00	4,00	10,50	7,01	14,60	7,48
champagne	Beverage	9,00	4,00	6,50	8,35	8,20	8,67
gin	Beverage	6,00	5,00	5,50	8,81	5,83	8,88
drink	Beverage	6,00	4,00	5,00	6,97	6,00	7,22
brandy	Beverage	7,00	1,00	4,00	8,54	7,00	9,34
juice	Beverage	4,00	4,00	4,00	8,54	4,00	8,53
bottle	Beverage	4,00	3,00	3,50	5,94	3,60	5,97
whisky	Beverage	4,00	2,00	3,00	7,93	3,33	8,07
cocktail	Beverage	3,00	2,00	2,50	8,67	2,67	8,75
lager	Beverage	4,00	1,00	2,50	8,35	4,00	9,01
soup	Beverage	4,00	1,00	2,50	7,86	4,00	8,53
soda	Beverage	2,00	2,00	2,00	8,76	2,00	8,75

Tabulka C.5: Populace sémantického typu *Beverage*; prahové hodnoty: $F_3 \geq 2$, $PMI_3 \geq 0,5$, $PMI_4 \geq 0$

Dodatek D

Obsah přiloženého CD

Data – data

Dokumentace – popis dat a programů

Scripts – použité programy

Text – text diplomové práce v PDF

Dodatek E

Popis programů na CD

Tento dodatek obsahuje stručný popis programů připojených na CD, kde lze nalézt také podrobnější dokumentaci. Zde je popsána pouze nápověda k uživatelským nástrojům.

Uživatelské nástroje

view_semtypes_lists.sh – program pro filtrování, třídění a prohlížení dvojic (sémantický typ, slovo) včetně jejich statistických charakteristik

Popis parametrů:

`-h | -help` ... vystiskne nápovědu

`-file <filename>` ... *filename* je vstupní soubor ve formátu CSV obsahující sloupce *slovo*, *sémantický typ*, F_1, F_2, F_3, PMI_3, F_4 a PMI_4, může být i zkomprimovaný pomocí gzip

`(-f1 | -f2 | -f3 | -f4) <value>` ... nastaví dolní prahovou hodnotu pro atributy $F_1, \ldots, F_4$

`(-pmi3 | -pmi4) <value>` ... nastaví dolní prahovou hodnotu pro atributy PMI_3 a PMI_4

`(-uf1 | -uf2 | -uf3 | -uf4) <value>` ... nastaví horní prahovou hodnotu pro atributy $F_1, \ldots, F_4$

`(-upmi3 | -upmi4) <value>` ... nastaví horní prahovou hodnotu pro atributy PMI_3 a PMI_4

`-sort <column1> ... <columnN>` ... nastaví, podle jakých sloupců se má třídit – výstup bude primárně setříděn podle *column1*, pak podle *column2*, ... a nakonec podle *columnN*. Přípustné hodnoty sloupců jsou: `noun, semtype, f1, f2, f3, pmi3, f4, pmi4`.

Příklad použití:

```
view_semtypes_lists.sh -file semtypes_lists.txt -f3 10
   -pmi3 1.5 -sort semtype f3 pmi3
```

Zobrazí pouze řádky, kde F_3 je větší nebo rovno 10 a PMI_3 je větší nebo rovno 1,5. Výstup bude setříděn podle sémantického typu, pak podle hodnoty F_3 a nakonec podle hodnoty PMI_3.

view_orig_sentences.sh – program na dohledání původních vět, kde se vyskytlo slovo s daným sémantickým typem. Současně se ke každé větě zobrazí i sloveso, číslo patternu a funkce argumentu.

Popis parametrů:

`<columns>` ... nastaví šířku obrazovky

`<sémantický typ>` ... hledaný sémantický typ

`<slovo>` ... hledané slovo

Příklad použití:

```
view_orig_sentences.sh $COLUMNS Plane aircraft
```

Nalezne konkordance ve kterých se vyskytlo slovo *aitcraft* a v patternu byl na stejné pozici sémantický typ *Plane*.

`$COLUMNS` je proměnná prostředí obsahující počet sloupců terminálu

Konverze definic patternů

convert_xml_into_csv.pl – program pro konverzi XML souboru s definicemi patternů na formát CSV

Patterns.pm – modul obsahující procedury pro práci s definicemi patternů v CSV formátu

Mezianotátorská shoda

iaa.pl – program pro výpočet mezianotátorské shody

prepare_annot_data_bnc.pl – program pro přípravu označkovaných dat z BNC k měření mezianotátorské shody

prepare_annot_data_bnc_hanks.pl – program pro zpracování dat v BNC označkovaných Patrickem Hanksem

prepare_annot_data_pedt.pl – příprava dat pro měření mezianotátorské shody na označkovaných datech v PEDT

Automatická extrakce slovesných argumentů a evaluace

dep_tree_to_dot_lang.pl – převod stanfordských závislostí do zdrojového kódu pro program Graphviz

Graphviz.pm – modul pro generování zdrojového kódu pro Graphviz

get_phrase_tree.btred – btred script, který vypíše složkové stromy z PEDT

get_sentences.btred – btred script, který vypíše tokenizované věty z PEDT

get_verb_arguments.btred – btred script, který nalezne slovesné argumenty v PEDT

check_and_split_parsing.pl – program pro technické zpracování výstupu Stanfordského parseru, rozdělí výstup na složkové stromy a stanfordské závislosti

find_and_map_verb_args.pl – program pro zpracování stanfordských závislostí a naleznutí argumentů označkovaného slovesa

STDR.pm – modul pro zpracování stanfordských závislostí

Penn_trees.pm – modul pro zpracování PTB stromů

NER.pm – modul pro zpracování výstupu z klasifikátoru jmenných entit

eval_pretty_print.pl – program pro přípravu evaluace

eval_parsing.pl – program pro samotnou evaluaci

run_evaluation.sh – script pro spuštění evaluace

Populace sémantických typů

get_instances_for_semtypes.pl – program pro propojení slovesného argumentu se sémantickými typy nacházejícími se na stejné kolokační pozici

make_lists.sh – vytvoří populace sémantických typů

make_lists_for_semtypes.pl – program pro počítání statistických charakteristik použitých při vytváření populace sémantických typů

count_doubles.pl – pomocný program, který posčítá hodnoty shodných dvojic

create_dictionary.pl – program pro vytvoření jednoduchého slovníku

entropy.pl – program pro výpočet entropie a vzájemné informace

Automatické přiřazování patternů

get_apriori_stats.pl – program ze vstupních dat zjistí rozložení patternů jednotlivých sloves

rule_pattern_recognizer.pl – program pro přiřazení patternů a následnou evaluaci

Nezařazené

Common.pm – pomocné funkce, které jsou využívány v dalších scriptech

process_data.pl – zpracování konkordancí, vstupem může být buď referenční vzorek nebo celá množina označkovaných konkordancí, podle vstupních parametrů se buď věty rozdělí do souborů podle sloves nebo se spočítají statistické charakteristiky referenčního vzorku

Literatura

[1] Webové stránky projektu CPA
`http://nlp.fi.muni.cz/projekty/cpa/`

[2] Patrick Hanks, James Pustejovsky: *A Pattern Dictionary for Natural Language Processing*, In Revue Francaise de linguistique appliquée, 2005.

[3] Patrick Hanks, Karel Pala, Pavel Rychlý: *Towards an empirically well-founded semantic ontology for NLP* In Proceedings of the 4th International Workshop on Generative Approaches to the Lexicon, Paris, 2007.

[4] Agirre, E. and Edmonds, P. (eds.): *Word Sense Disambiguation: Algorithms and Applications*. Springer, 2007. (vybrané kapitoly)

[5] Patrick Hanks: *Lexical Analysis: Norms and Exploitations*. Připravováno. MIT Press.

[6] John Sinclair: The lexical item. In: Hanks, P. (ed.), Lexicology: Critical Concepts in Linguistics. 6 volumes. Routledge. Prvně publikováno v Weigand, E. (ed.) Contrastive Lexical Semantics Amsterdam: John Benjamins, pp. 1–24. (1998, 2008).

[7] Zdeněk Žabokrtský: *Valency Lexicon of Czech Verbs*, MFF UK, 2005.

[8] Dana Hlaváčková: *Databáze slovesných valenčních rámců VerbaLex*, dizertační práce FF MUNI, Brno, 2007.

[9] Webové stránky projektu FrameNet
`http://framenet.icsi.berkeley.edu/`

[10] Webové stránky projektu PropBank
`http://verbs.colorado.edu/m̃palmer/projects/ace.html`

[11] Webové stránky projektu VerbNet
`http://verbs.colorado.edu/m̃palmer/projects/verbnet.html`

[12] Markéta Lopatková, Zdeněk Žabokrtský, Václava Kettnerová a kol.: *Valenční slovník českých sloves*, Univerzita Karlova v Praze, Nakladatelství Karolinum, Praha, 2008.

[13] Webové stránky projektu VALLEX
`http://ufal.mff.cuni.cz/vallex/`

[14] Webové stránky Centra zpracování přirozeného jazyka
http://nlp.fi.muni.cz/

[15] Webové stránky projektu Trang
http://www.thaiopensource.com/relaxng/trang.html

[16] Silvie Cinková, Martin Holub, Pavel Rychlý, Lenka Smejkalová, Jana Šindlerová: *Can Corpus Pattern Analysis Be Used in NLP?*, v tisku TSD 2010, Brno, 2010.

[17] Jan Popelka: *O klasifikaci anglických sloves dle PDEV a české překladové ekvivalenci*, seminární práce MFF UK, Praha, 2010.

[18] Webové stránky projektu Sketch Engine
http://www.sketchengine.co.uk/

[19] Marie-Catherine de Marneffe, Bill MacCartney, Christopher D. Manning: *Generating Typed Dependency Parses from Phrase Structure Parses*, LREC 2006.

[20] Marie-Catherine de Marneffe, Christopher D. Manning: *The Stanford typed dependencies representation*, In COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation. 2008.

[21] Marie-Catherine de Marneffe, Christopher D. Manning: *The Stanford typed dependencies manual*
http://nlp.stanford.edu/software/dependencies_manual.pdf

[22] Webová stránka FAQ Stanfordského parseru
http://nlp.stanford.edu/software/parser-faq.shtml

[23] Webová stránka software Graphviz
http://www.graphviz.org/

[24] Webové stránky Charniak-Johnosonova parseru
ftp://ftp.cs.brown.edu/pub/nlparser/

[25] Webové stránky McDonaldova parseru
http://sourceforge.net/projects/mstparser/

[26] Definice CoNLL formátu:
http://nextens.uvt.nl/depparse-wiki/DataFormat

[27] Webové stránky projektu TectoMT
http://ufal.mff.cuni.cz/tectomt/

[28] Webové stránky projektu Morče
http://ufal.mff.cuni.cz/morce/

[29] Adwait Ratnaparkhi. *A Maximum Entropy Part-Of-Speech Tagger*. In Proceedings of the Empirical Methods in Natural Language Processing Conference, May 17-18, 1996. University of Pennsylvania.

[30] Martin Holub, Lenka Smejkalová, Pavel Rychlý, Silvie Cinková: *Evaluation of Verb Arguments Extraction Using Automatic Parsing*, zasláno na konferenci EMNLP 2010.

[31] Silvie Cinková, Martin Holub, Lenka Smejkalová: *The Lexical Population of Semantic Types in Hanks's PDEV*, A Way with Words: Recent Advances in Lexical Theory and Analysis. A Festschrift for Patrick Hanks, Kampala, 2010.

[32] Webové stránky projektu Stanford NE Recognizer
`http://nlp.stanford.edu/software/CRF-NER.shtml`

[33] Damian Conway: *An Algorithmic Approach to English Pluralization*, webová stránka
`http://www.csse.monash.edu.au/d̃amian/papers/HTML/Plurals.html`