Počítačové zpracování češtiny

Závislostní syntaktická analýza

Daniel Zeman

http://ufal.mff.cuni.cz/course/popj1/

Závislostní model

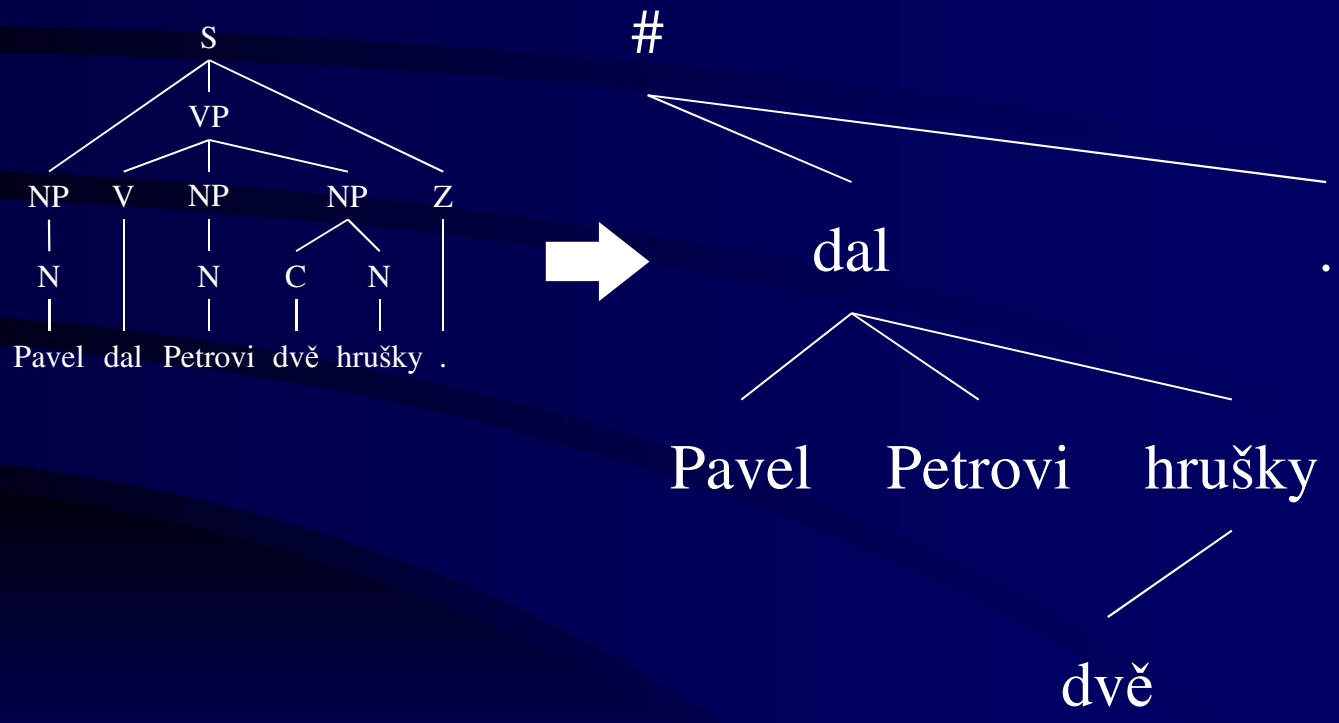
- Shrnutí syntaktických vztahů:
- Členění věty na **fráze** (složky).
  - Hlavní stavební kámen frázového (složkového) modelu.
- Hlava fráze, **závislost** ostatních členů fráze na hlavě.
  - Hlava = řídící větný člen, ostatní členy fráze jsou závislé.
  - Hlavní stavební kámen závislostních stromů.
- I ve složkových stromech lze mluvit o závislostech a naopak.

Příklad závislostního stromu

- [#,0] ([dal,2] ([Pavel,1], [Petrovi,3], [hrušky,5] ([dvě,4])), [.,6])

Pojmenování závislostí

Frázové vs. závislostní stromy



Frázové vs. závislostní stromy

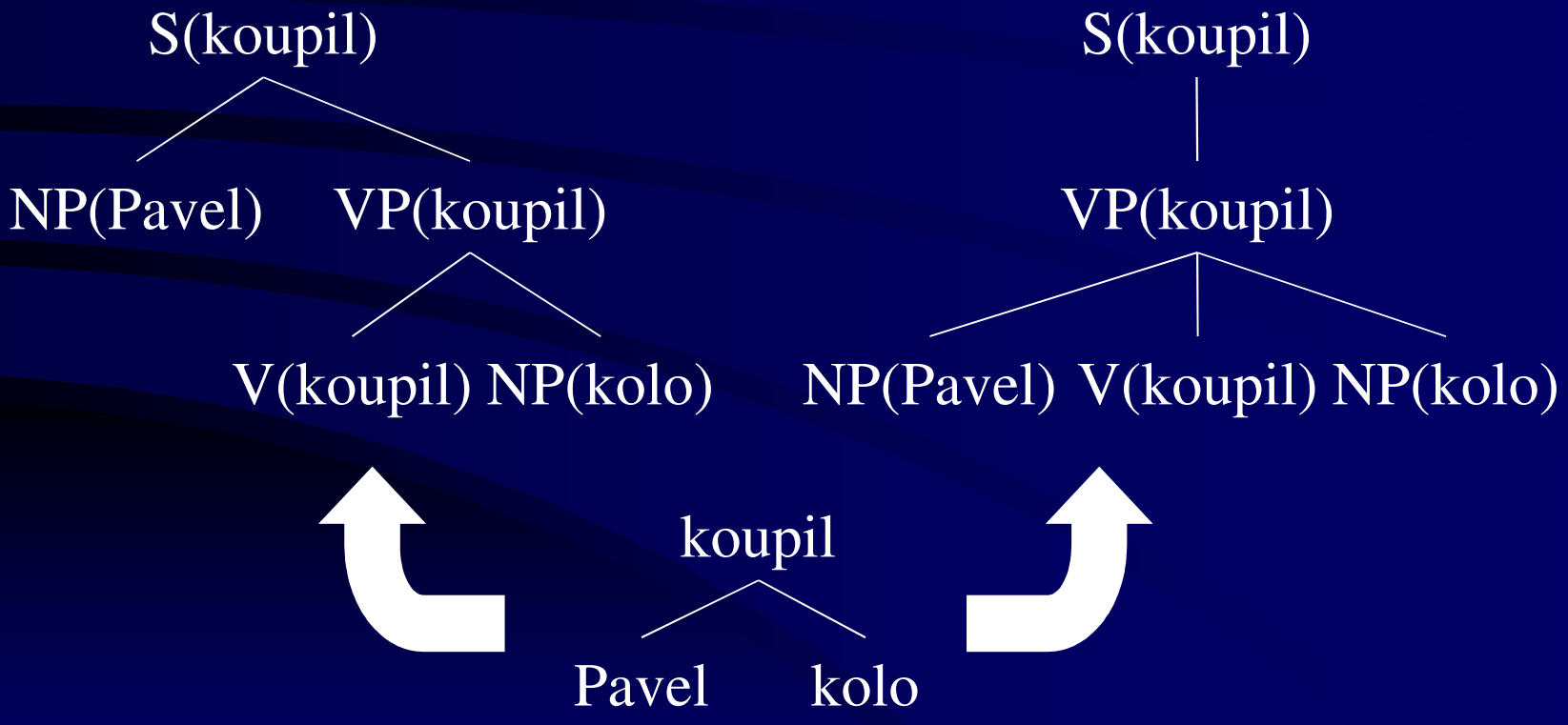
- Frázové (složkové) stromy.
  - Ukazují členění věty na fráze a pojmenovávají je.
  - Nezdůrazňují, co je hlava, které slovo na kterém závisí.
  - Nemusí obsahovat funkci, druh závislosti.
- Závislostní stromy.
  - Ukazují závislosti mezi slovy a pojmenovávají je.
  - Nezachycují podobnost tvoření různých částí věty, rekurzi.
  - Nezachycují průběh budování věty, blízkost závislých členů hlavě.
  - Neobsahují neterminály, druhy frází — ty lze leda odhadovat ze značek hlav.

Rozdíly závislostního a frázového modelu

- Chceme převést frázový strom F na závislostní strom Z nebo obráceně.
- Frázový strom neříká, co je hlava fráze.
  - Pro převod F $\rightarrow$ Z potřebujeme výběrovou funkci, která pro každé pravidlo frázové gramatiky řekne, který symbol na pravé straně je hlava.
- Závislostní strom neukazuje, jak věta vznikla (rekurze), ani nutně nepostihuje celé dělení na fráze.
  - Neříká, co bylo do věty přidáno „dříve“ a co „později“.
  - Více frázových struktur může vést na stejnou závislostní $\Rightarrow$
  - Převod zpět (Z $\rightarrow$ F) je nejednoznačný.

Příklad

- Více frázových stromů vede na tentýž závislostní strom.

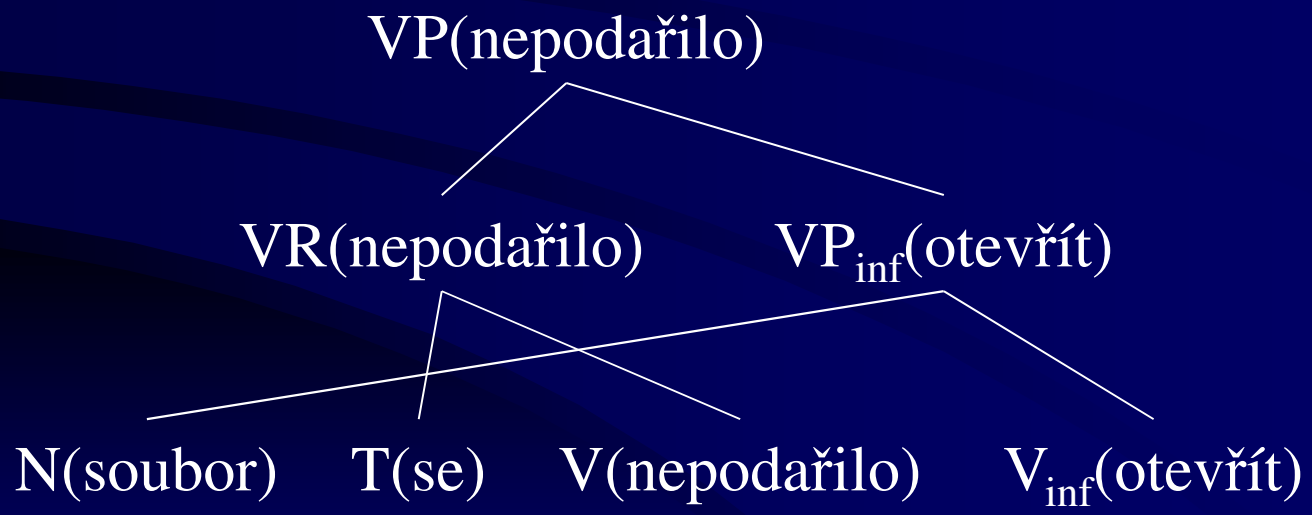


Rozdíly závislostního a frázového modelu

- V závislostním stromě nevíme, jak se jmenují fráze (protože ani nevíme, co jsou fráze, viz předchozí snímek).
  - Potřebujeme funkci, která podle hlavy fráze určí jméno fráze.
  - Potřebujeme? Pro význam je potřeba znát vztahy a **jejich druh**, ale ne vědět, co bylo vygenerováno dříve a co později.
- Ve frázovém stromě neznáme druh vztahu mezi hlavou a ostatními členy — **analytické funkce**. (Ale srov. funkční značky v Penn Treebanku.)
  - Funkce, která pro každou frázi určí typ závislosti členů na hlavě. (Lze říci pro každou frázi stejně jako lze určit hlavu.)
- Podstatný rozdíl: frázové stromy jsou **pevně spjaty se slovosledem!**

Nespojité fráze

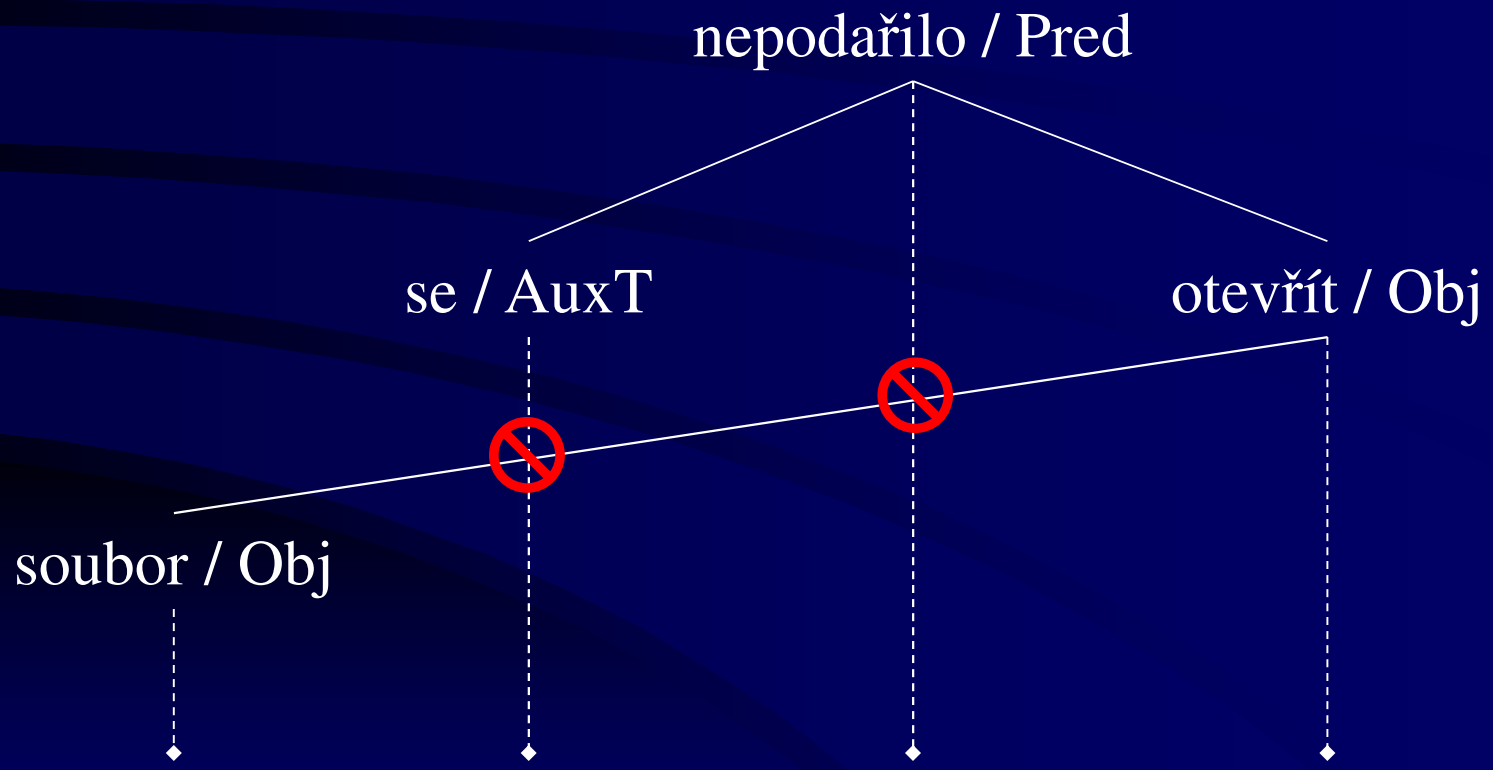
- Nedají se generovat normální gramatikou!
- Nedají se reprezentovat závorkováním.
- *(Soubor (se nepodařilo) otevřít).*



Neprojektivita

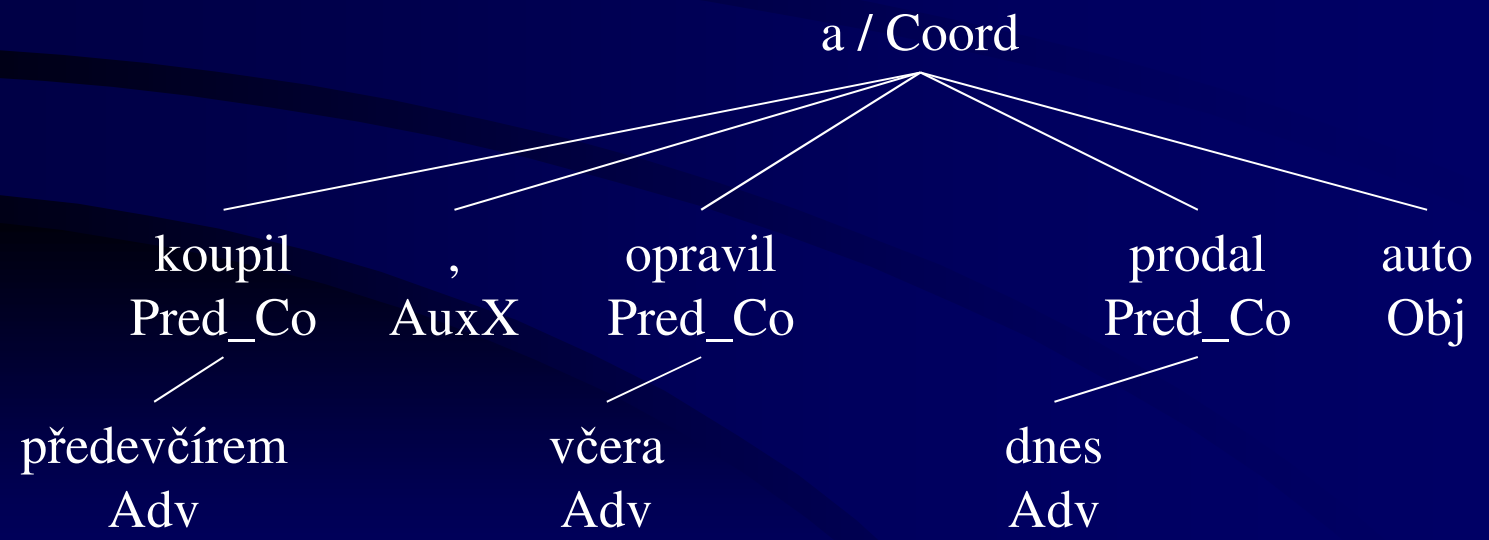
- Závislostní strom včetně slovosledu (vodorovná souřadnice uzlu).
- Projekce na podstavu: kolmice z uzlu protíná nějakou hranu (**neprojektivní hrana**).
- Formálně:
  - Závislost ([ř,$x_{ř}$],[z,x_z]). x_w je pořadí slova w ve větě.
  - Existuje uzel [u,x_u], že $x_{ř} < x_u < x_z$ nebo $x_z < x_u < x_{ř}$ a [u,x_u] neleží v podstromu, jehož kořenem je [ř,$x_{ř}$].
- Neformálně: řetězec odpovídající podstromu řídícího uzlu není souvislý, jsou v něm **díry**.

Neprojektivita: závislostní strom ji umí!

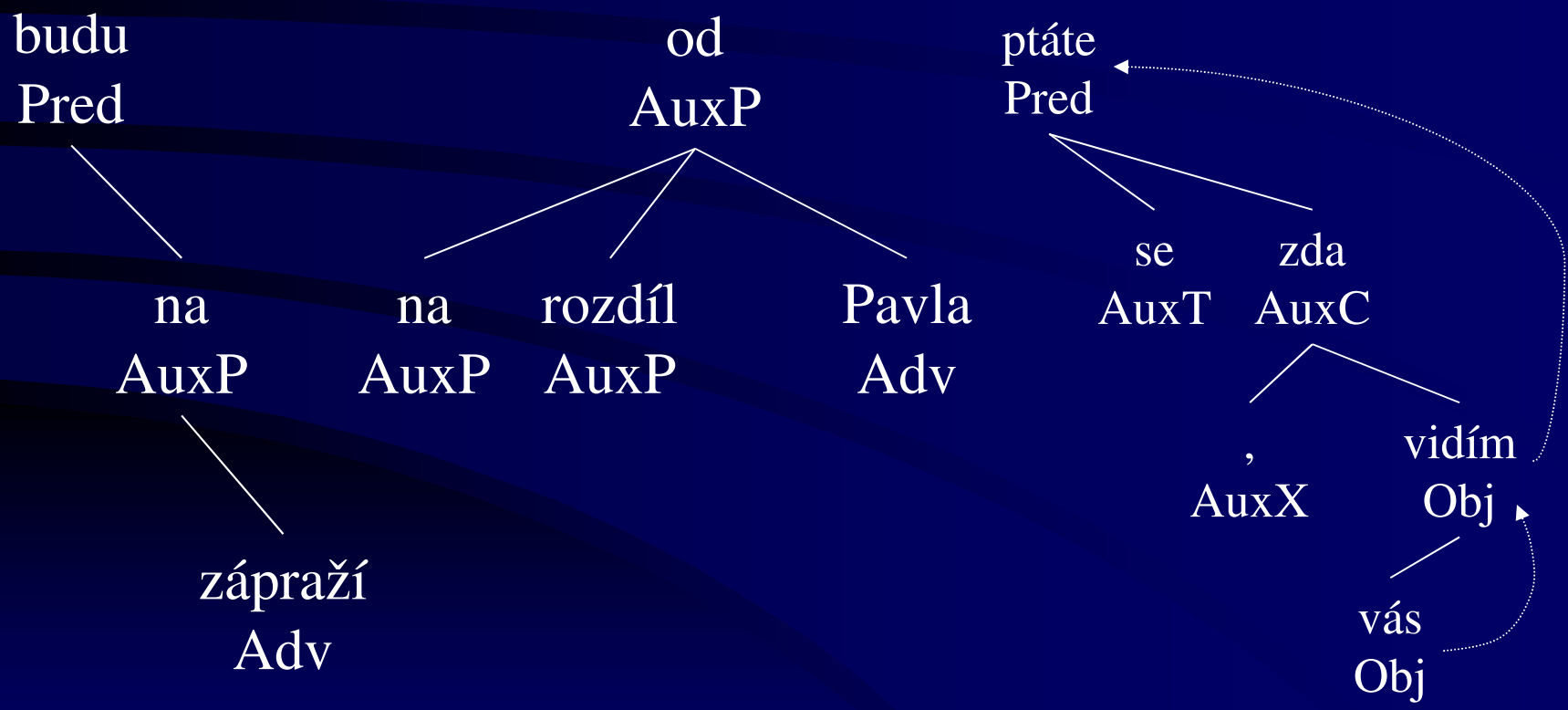


Problém: ne všechno je závislost

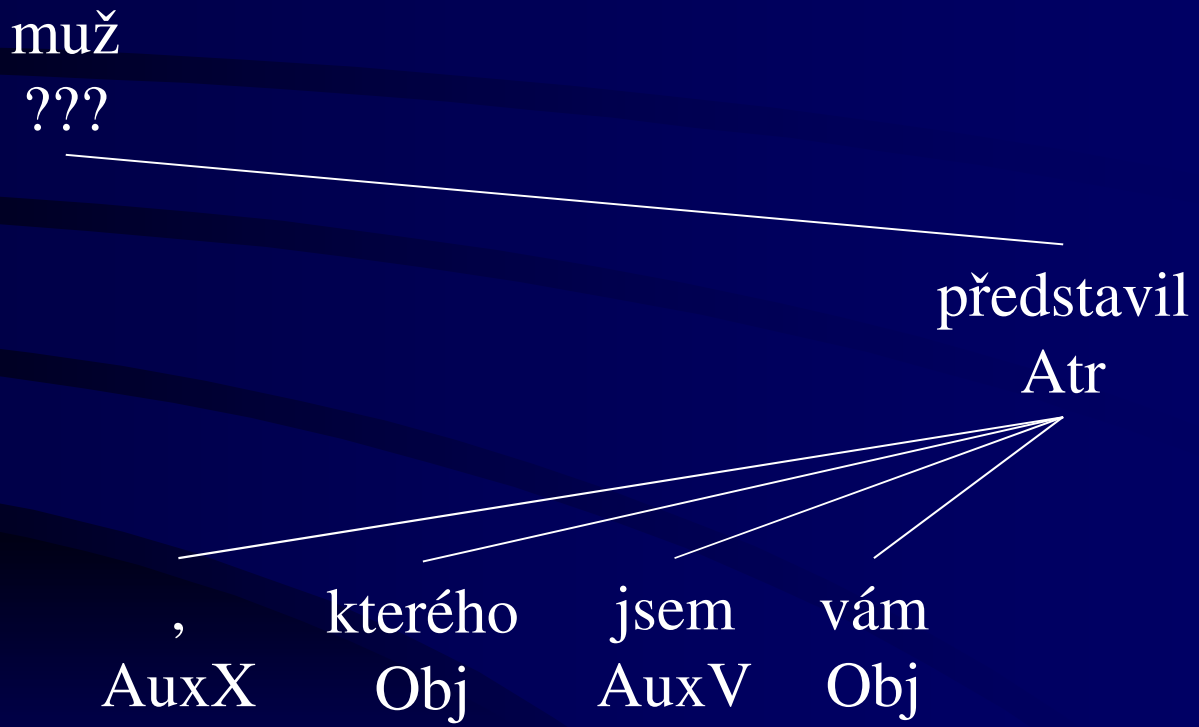
- Koordinace a apozice.
  - Společné rozvití celé koordinace × členy koordinace.
  - Pomocné uzly (interpunkce apod.).



Předložkové fráze, vnořené klauze spojkové



Vnořené klauze vztažné



Fráze, závislosti a jiné modely

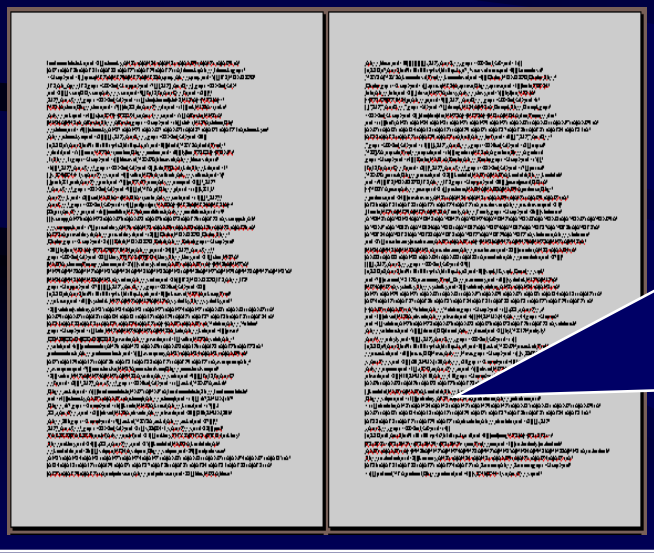
- Frázový (složkový, bezprostředně-složkový).
  - Nejrozšířenější ve světě, vhodný pro angličtinu.
  - Frázové (bezkontextové) gramatiky.
- Závislostní.
  - Oblíbený v některých zemích, např. u nás.
  - Zvláště vhodný pro jazyky s volným slovosledem.
  - Závislostní gramatiky, gramatiky závislostních stromů.
- Kategoriální gramatiky.
- Tree-adjoining grammars (TAGs).
- A řada dalších…

Závislostní gramatika

- Na rozdíl od frázového modelu je spojení s gramatikou umělé („závislostní strom neodráží způsob svého vzniku“).
- Pro češtinu asi neexistuje implementace.
- Bezkontextová gramatika + výběrová funkce (pouze projektivní konstrukce).
- Gramatika, přepisující neterminál na celý podstrom *(gramatika závislostních stromů)*.
- Souvislost s link grammars, tree-adjoining grammars, kategoriálními gramatikami.
- HPSG, unifikace.

Závislostní analýza s pomocí statistického modelu

Korpus ručně anotovaných textů



p(hrana([**ve**], [**dveřích**])) = p_1
p(hrana([**v**], [**dveřích**])) = p_2
p(hrana([**ve**], [**dveře**])) = p_3
p(hrana([**ve**], [**dveřím**])) = p_4

kde asi:

$p_1 > p_2 > p_3$ a $p_4 \not> 0$

Základní metoda: hledáme nejpravděpodobnější strom

- Hledáme strom M, který je s největší pravděpodobností zápisem dané věty S.
- Formálně: $\arg\max_M p(M|S)$
  - Podle Bayesova vzorce můžeme vyjádřit: $p(M|S) = \frac{p(S|M) \cdot p(M)}{p(S)}$
  - p(S|M) je pravděpodobnost, že věta, jejímž zápisem je strom M, je právě S.
  - p(M) je pravděpodobnost výskytu (existence) stromu M.
  - p(S) je pravděpodobnost výskytu věty S. Z hlediska hledání nejpravděpodobnějšího stromu je ovšem pouhou konstantou, která výsledek neovlivní:

Základní metoda: hledáme nejpravděpodobnější strom

- p(S) je pravděpodobnost výskytu věty S. Z hlediska hledání nejpravděpodobnějšího stromu je ovšem pouhou konstantou, která výsledek neovlivní:

$$\arg\max_M \frac{p(S|M)\cdot p(M)}{p(S)} = \arg\max_M (p(S|M)\cdot p(M))$$

- Úkolem je tedy odhadnout pravděpodobnosti p(S|M) a p(M).
- p(S|M) maximalizujeme tak, že strom M konstruujeme přímo ze slov věty S.

Pravděpodobnost stromu

- Opět předpoklad: hrany jsou na sobě nezávislé (silné a chybné — lepší model by byl opatrnější).
- Součin pravděpodobností hran.

$$p(M) = \prod_{i=1}^{n} p(h_i)$$

- Jak najít nejpravděpodobnější strom? Podobně jako u značkování: Viterbiho algoritmem! (Střední cesta mezi hladovým algoritmem a backtrackingem.)

MST Parser

- McDonald et al., HLT-EMNLP 2005
- http://sourceforge.net/projects/mstparser/
- MST = maximum spanning tree = nejlépe ohodnocená kostra (orientovaného) grafu
- Začne se úplným grafem.
  - Tj. předpokládáme možnou závislost mezi libovolnými dvěma slovy ve větě.
- Postupně odstraňovat špatně hodnocené závislosti.
- Hodnocení závislostí zajistí statistický algoritmus.
  - Ten se učí na rysech hrany.
  - Rys je např. lemma, slovní druh, pád… řídícího / závislého uzlu.

MST Parser

- Říct, na které rysy se má parser zaměřit, lze úpravou zdrojáku (Java).
- Není snadné zahrnout tzv. *rysy 2. řádu*
  - Tj. ohodnocení závislosti podmínit např. i morfologickou značkou prarodiče.
- Parser lze pustit v neprojektivním režimu.
- Trénování na celém PDT údajně trvá asi 30 hodin.
  - Je nutné procházet kombinace rysů a hledat, které z nich jsou nejužitečnější.
- Vlastní parsing je ve srovnání s tím rychlý.

Malt Parser

- Nivre et al., *Natural Language Engineering*, 2007
- http://maltparser.org/
- Založeno na přechodech *(transitions)* z jedné konfigurace do druhé.
- Konfigurace:
  - Vstupní buffer (slova ve větě zleva doprava)
  - Zásobník
  - Výstupní strom (slova, závislosti a značky závislostí)
- Přechody:
  - Shift: přesune slovo z bufferu na zásobník
  - Larc: levá závislost mezi horními dvěma slovy v zásobníku
  - Rarc: pravá závislost mezi horními dvěma slovy v zásobníku

Malt Parser

- Parser řídí tzv. orákulum, které na základě aktuální konfigurace vybere přechodovou operaci.
- Trénování: strom z trénovacích dat rozložit na posloupnost konfigurací a přechodů
  - Někdy existuje více možností
    - Různé trénovací algoritmy: např. hladově co nejdřív tvořit závislosti.
- Orákulum se učí na základě rysů konfigurace.
  - Např. slovo, lemma, slovní druh, rod, pád…
    - *n*-tého slova od vrcholu zásobníku
    - *k*-tého slova zbývajícího v bufferu
    - konkrétního uzlu v již vytvořené části výstupního stromu

Malt Parser

- Na trénování je opět použit statistický algoritmus, v tomto případě tzv. *Support Vector Machines (SVM).*
  - Klasifikátor. Na vstupu vektory: hodnoty všech rysů aktuální konfigurace.
  - Při trénování k tomu navíc výstupní hodnota, tj. název operace (Shift / Larc / Rarc).
  - Při ostrém běhu výstupní hodnotu řekne natrénované orákulum (SVM).
- Trénování nad celým PDT může trvat celé týdny.
  - Složitost je $O(n^2)$, kde n je počet trénovacích příkladů.
  - Z PDT lze vytáhnout přes 3 milióny trénovacích příkladů.
- Vlastní parsing už je rychlejší (řádově věta za vteřinu).

Příklad analýzy Malt Parserem

- stack = #
- buffer = Pavel dal Petrovi dvě hrušky .
- tree =

SHIFT

- stack = # Pavel
- buffer = dal Petrovi dvě hrušky .
- tree =

Příklad analýzy Malt Parserem

- stack = # Pavel
- buffer = dal Petrovi dvě hrušky .
- tree =

SHIFT

- stack = # Pavel dal
- buffer = Petrovi dvě hrušky .
- tree =

Příklad analýzy Malt Parserem

- stack = # Pavel dal
- buffer = Petrovi dvě hrušky .
- tree =

LARC

- stack = # dal
- buffer = Petrovi dvě hrušky .
- tree = dal(Pavel)

Příklad analýzy Malt Parserem

- stack = # dal
- buffer = Petrovi dvě hrušky .
- tree = dal(Pavel)

SHIFT

- stack = # dal Petrovi
- buffer = dvě hrušky .
- tree = dal(Pavel)

Příklad analýzy Malt Parserem

- stack = # dal Petrovi
- buffer = dvě hrušky .
- tree = dal(Pavel)

RARC

- stack = # dal
- buffer = dvě hrušky .
- tree = dal(Pavel,Petrovi)

Příklad analýzy Malt Parserem

- stack = # dal
- buffer = dvě hrušky .
- tree = dal(Pavel,Petrovi)

SHIFT

- stack = # dal dvě
- buffer = hrušky .
- tree = dal(Pavel,Petrovi)

Příklad analýzy Malt Parserem

- stack = # dal dvě
- buffer = hrušky .
- tree = dal(Pavel,Petrovi)

SHIFT

- stack = # dal dvě hrušky
- buffer = .
- tree = dal(Pavel,Petrovi)

Příklad analýzy Malt Parserem

- stack = # dal dvě hrušky
- buffer = .
- tree = dal(Pavel,Petrovi)

LARC

- stack = # dal hrušky
- buffer = .
- tree = dal(Pavel,Petrovi),hrušky(dvě)

Příklad analýzy Malt Parserem

- stack = # dal hrušky
- buffer = .
- tree = dal(Pavel,Petrovi),hrušky(dvě)

RARC

- stack = # dal
- buffer = .
- tree = dal(Pavel,Petrovi,hrušky(dvě))

Příklad analýzy Malt Parserem

- stack = # dal
- buffer = .
- tree = dal(Pavel,Petrovi,hrušky(dvě))

RARC

- stack = #
- buffer = .
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)))

Příklad analýzy Malt Parserem

- stack = #
- buffer = .
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)))

SHIFT

- stack = # .
- buffer =
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)))

Příklad analýzy Malt Parserem

- stack = # .
- buffer =
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)))

RARC

- stack = #
- buffer =
- tree = #(dal(Pavel,Petrovi,hrušky(dvě)),.)

Neprojektivní režim Maltu

- Lze dokázat, že uvedený přechodový systém je
  - korektní
    - výsledkem je vždy strom, tj. souvislý graf bez cyklů
  - úplný pro množinu projektivních stromů
    - každý projektivní strom lze vyjádřit jako posloupnost přechodů
- Jak přidat neprojektivní závislosti?
  - Nová přechodová operace **`SWAP`**:
  - Vyjmout druhé slovo od povrchu zásobníku a vrátit ho do bufferu. Tím se prohodí pořadí slov na vstupu.
  - Tuto operaci dovolit jen u slov, která ještě nebyla prohozena (jejich pořadí na zásobníku odpovídá jejich původnímu pořadí ve větě).

Příklad neprojektivní analýzy

- stack = #
- buffer = Soubor se nepodařilo otevřít .
- tree =

SHIFT

- stack = # Soubor
- buffer = se nepodařilo otevřít .
- tree =

Příklad neprojektivní analýzy

- stack = # Soubor
- buffer = se nepodařilo otevřít .
- tree =

SHIFT

- stack = # Soubor se
- buffer = nepodařilo otevřít .
- tree =

Příklad neprojektivní analýzy

- stack = # Soubor se
- buffer = nepodařilo otevřít .
- tree =

SHIFT

- stack = # Soubor se nepodařilo
- buffer = otevřít .
- tree =

Příklad neprojektivní analýzy

- stack = # Soubor se nepodařilo
- buffer = otevřít .
- tree =

LARC

- stack = # Soubor nepodařilo
- buffer = otevřít .
- tree = nepodařilo(se)

Příklad neprojektivní analýzy

- stack = # Soubor nepodařilo
- buffer = otevřít .
- tree = nepodařilo(se)

SHIFT

- stack = # Soubor nepodařilo otevřít
- buffer = .
- tree = nepodařilo(se)

Příklad neprojektivní analýzy

- stack = # Soubor nepodařilo otevřít
- buffer = .
- tree = nepodařilo(se)

SWAP

- stack = # Soubor otevřít
- buffer = nepodařilo .
- tree = nepodařilo(se)

Příklad neprojektivní analýzy

- stack = # Soubor otevřít
- buffer = nepodařilo .
- tree = nepodařilo(se)

LARC

- stack = # otevřít
- buffer = nepodařilo .
- tree = nepodařilo(se),otevřít(Soubor)

Příklad neprojektivní analýzy

- stack = # otevřít
- buffer = nepodařilo .
- tree = nepodařilo(se),otevřít(Soubor)

SHIFT

- stack = # otevřít nepodařilo
- buffer = .
- tree = nepodařilo(se),otevřít(Soubor)

Příklad neprojektivní analýzy

- stack = # otevřít nepodařilo
- buffer = .
- tree = nepodařilo(se),otevřít(Soubor)

LARC

- stack = # nepodařilo
- buffer = .
- tree = nepodařilo(se,otevřít(Soubor))

Příklad neprojektivní analýzy

- stack = # nepodařilo
- buffer = .
- tree = nepodařilo(se,otevřít(Soubor))

RARC

- stack = #
- buffer = .
- tree = #(nepodařilo(se,otevřít(Soubor)))

Příklad neprojektivní analýzy

- stack = #
- buffer = .
- tree = #(nepodařilo(se,otevřít(Soubor)))

SHIFT

- stack = # .
- buffer =
- tree = #(nepodařilo(se,otevřít(Soubor)))

Příklad neprojektivní analýzy

- stack = # .
- buffer =
- tree = #(nepodařilo(se,otevřít(Soubor)))

RARC

- stack = #
- buffer =
- tree = #(nepodařilo(se,otevřít(Soubor)),.)

Úspěšnost Maltu a MST

- Na češtině (PDT):
  - MST Parser přes 85 %
  - Malt Parser 86 % (ale časově náročné)
    - Úspěšnost na větách asi 35 %, to je hodně!
  - Daly by se také zkombinovat, protože pracují dost odlišně.
- Na dalších jazycích (soutěž CoNLL)
  - Většinou je MST o něco lepší.
  - Absolutní čísla nejsou srovnatelná napříč jazyky, hodně závisí na konkrétním korpusu.

Rysy jsou základ úspěchu

- Společný rys MST a Maltu:
  - Jsou schopny vzít v úvahu velké množství rysů *(features)* vstupního textu.
  - Netriviální algoritmus strojového učení zařídí, že z těchto rysů dokážeme vybrat ty důležité.
  - Algoritmy strojového učení jsou obecné klasifikátory.
    - Většinou už existuje nějaká knihovna, která se stáhne a použije.
    - Konkrétní problém (zde budování stromu) je potřeba převést na posloupnost klasifikačních rozhodnutí, tj. vektorů (hodnoty rysů + odpověď).

Valence

- Sloveso je centrum věty.
- Větné členy závislé na slovesu jsou jeho doplnění.
- Dva druhy doplnění:
  - vnitřní = aktanty = participanty
  - vnější = volná

Druhy slovesných doplnění: vnitřní × volné

a) Nejvýše jedno doplnění daného typu může rozvíjet tentýž slovesný výskyt … *vnitřní doplnění*.

b) Doplnění téhož typu může jeden slovesný výskyt rozvíjet vícekrát … *volné doplnění*.

a´) Toto doplnění může stát pouze u jisté skupiny sloves … *vnitřní doplnění kromě konatele*.

b´) Toto doplnění může stát téměř u každého slovesa *volné doplnění nebo konatel*.

Druhy slovesných doplnění: (sémanticky) povinné × volitelné

- I povinné doplnění může ve větě chybět, pak ale je skryto nebo známo z kontextu.
- Pokud doplnění chybí, ale autor věty ho nemůže neznat, jde o *povinné doplnění*.
- Příklad 1: Moji přátelé přijeli. Kam? *Nevím. Odkud? Nevím.
- Příklad 2: Moji přátelé odjeli. Odkud? *Nevím. Kam? Nevím.

Druhy slovesných doplnění: argument × adjunct

- Jde v podstatě o opozici povinné × volitelné, ale posuzuje se povrchově, ne sémanticky.
- *„Odjeli odněkud.“* „Odněkud“ sémanticky povinné, ale povrchově (spíše) volitelné.
- *„Položil to na stůl.“* „Na stůl“ je asi povinné i povrchově, ale méně než v angličtině.
- Nadále budeme termínem *povinné* × *volitelné* rozumět povrchovou povinnost a volitelnost, tj. *argument* × *adjunct*.

Druhy vnitřních doplnění

- Konatel (agens, actor, bearer, původce)
- Trpitel (paciens, zasažený předmět)
- Adresát (addressee, nepřímý předmět)
- Původ (origo, origin)
- Výsledek (efekt, effect)
- Do hloubkového rámce patří všechna vnitřní doplnění a sémanticky povinná volná doplnění.

Příklad věty se všemi vnitřními doplněními

- Doplnění v závorkách jsou volitelná, ostatní povinná. (Téměř) všechna jsou však vnitřní (viz poznámku dole)!

  Matka$_{\texttt{Act}}$ předělala (dětem$_{\texttt{Adr}}$) loutku$_{\texttt{Pat}}$ (z kašpárka$_{\texttt{Orig}}$) (na čerta$_{\texttt{Ef}}$).

  - Poznámka: Tento prastarý příklad se opisuje z místa na místo, ale není přesný. „dětem“ je spíš benefaktor než adresát, tj. volné, nikoli vnitřní doplnění. Takovýhle dativ jde totiž přidat skoro všude. Skutečný adresát by to byl např. v „Matka dala dětem loutku.“

Základní třídy rámců

- Slovesa bez aktantů *(prší)*
- Nepřechodná slovesa *(Pavel padá)*
- Přechodná slovesa *(Pavel bije Petra)*
- Dvojpřechodná slovesa *(Pavel dává Petrovi knihu)*
- A řada dalších: při povrchovém členění asi 180 tříd?

Rámce lze automaticky odfiltrovat z korpusu

Jan viděl Marii.

vs.

Jan viděl Marii včera kolem páté na nádraží.

Typy povinných doplnění — příklady

- Jmenné fráze: N4, N3, N2, N7, N1(Pnom)
- Předložkové fráze: R2(bez), R3(k), R4(na), R6(na), R7(s)…
- Zvratná zájmena „se“, „si“: PR4, PR3.
- Vedlejší věty: S, JS(že), JS(zda)…
- Infinitivy (VINF), příčestí trpná (VPAS), příslovce (DB)…