Natuurlijke-Taalverwerking 1

Week 6

Chart parsing; PCFG

Overzicht

- Herhaling: chart parsing
- Probabilistische CFGs
- Evaluatie
- Beperkingen van PCFGs
- Disambiguatie met unificatiegrammatica's
- Disambiguatie in Alpino

Een chart

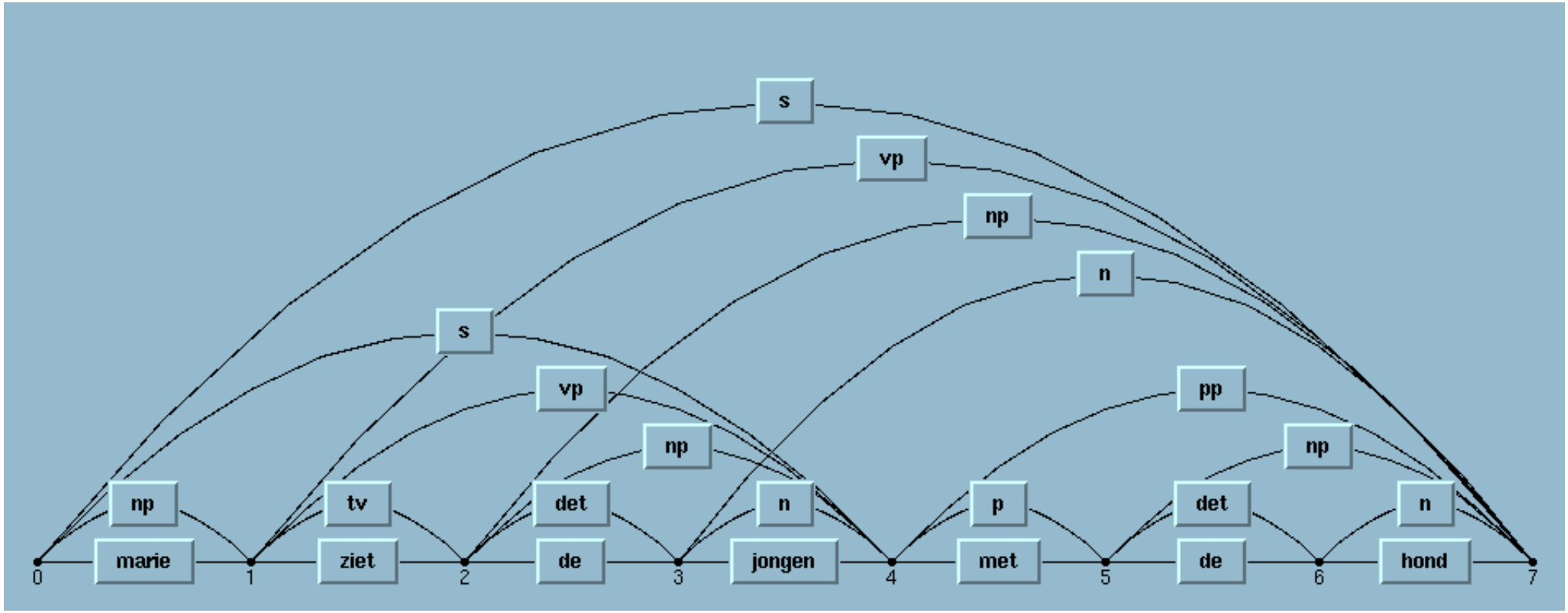

s
vp
np
n
s
vp
pp
np
np
np
tv
det
n
p
det
n
marie
ziet
de
jongen
met
de
hond
0
1
2
3
4
5
6
7

Chart Parser

- Gebruik *chart* met alle gevonden items
- Gebruik *agenda* met alle items die je nog aan chart moet toevoegen
- Voor elk item in de agenda bepaal je of er nieuwe items kunnen worden toegevoegd
- `item(Cat,P0,P,HisList)`

Voorbeeld grammatica

```
rule(s,[np,vp]).      lex(det,Det) :- determiner(Det).     % de het een dat
rule(np,[det,n]).     lex(n,Noun)  :- noun(Noun).          % ei kip stippen
rule(n,[a,n]).        lex(a,Adj)   :- adjective(Adj).      % mooi groene
rule(n,[n,pp]).       lex(v,Verb)  :- verb(Verb).          % ziet bestaat
rule(vp,[v]).         lex(p,Prep)  :- preposition(Prep).   % met van in
rule(vp,[v,np]).      lex(np,NP)   :- name(NP).            % Arie Bob Jan
rule(vp,[vp,pp]).     lex(np,NP)   :- pronoun(NP).         % hij zij het dit
rule(pp,[p,np]).
```

Initialize Agenda: lexical lookup

```
Arie ziet het ei met de stippen

Agenda:                                          Chart:
[item(np,  0,1,[lex('Arie')]),
 item(v,   1,2,[lex(ziet)]),
 item(det, 2,3,[lex(het)]),
 item(np,  2,3,[lex(het)]),
 item(n,   3,4,[lex(ei)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

Treat first item on agenda:

```
add item item(np,  0,1,[lex('Arie')]) to chart
no new items
```

Agenda:

```
[item(v,   1,2,[lex(ziet)]),
 item(det, 2,3,[lex(het)]),
 item(np,  2,3,[lex(het)]),
 item(n,   3,4,[lex(ei)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

Chart:

```
item(np,  0,1,[lex('Arie')]).
```

```
add item(v,1,2,[lex(ziet)]) to chart,
new item item(vp,1,2,[r(v)])
```

```
Agenda:
[item(vp,  1,2,[r(v)]),
 item(det, 2,3,[lex(het)]),
 item(np,  2,3,[lex(het)]),
 item(n,   3,4,[lex(ei)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
```

```
add item(vp,1,2,[r(v)]) to chart,
new item(s,0,2,[r(np,1,vp)])

Agenda:                                  Chart:
[item(s,   0,2,[r(np,1,vp)]),            item(np,  0,1,[lex('Arie')]).
 item(det, 2,3,[lex(het)]),              item(v,   1,2,[lex(ziet)]).
 item(np,  2,3,[lex(het)]),              item(vp,  1,2,[r(v)]).
 item(n,   3,4,[lex(ei)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

```
add item(s,0,2,[r(np,1,vp)]) to chart,
no new items

Agenda:                                  Chart:
[item(det, 2,3,[lex(het)]),              item(np,  0,1,[lex('Arie')]).
 item(np,  2,3,[lex(het)]),              item(v,   1,2,[lex(ziet)]).
 item(n,   3,4,[lex(ei)]),               item(vp,  1,2,[r(v)]).
 item(p,   4,5,[lex(met)]),              item(s,   0,2,[r(np,1,vp)]).
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

```
add item(det, 2,3,[lex(het)]) to chart,
no new items

Agenda:                                  Chart:
[item(np,  2,3,[lex(het)]),              item(np,  0,1,[lex('Arie')]).
 item(n,   3,4,[lex(ei)]),               item(v,   1,2,[lex(ziet)]).
 item(p,   4,5,[lex(met)]),              item(vp,  1,2,[r(v)]).
 item(det, 5,6,[lex(de)]),               item(s,   0,2,[r(np,1,vp)]).
 item(n,   6,7,[lex(stippen)])]          item(det, 2,3,[lex(het)]).
```

```
add item(np,2,3,[lex(het)])
new item(vp,1,3,[r(v,2,np)])

Agenda:
[item(vp,  1,3,[r(v,2,np)]),
 item(n,   3,4,[lex(ei)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]

Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
```

```
add item(vp,1,3,[r(v,2,np)])
new item(s,0,3,[r(np,1,vp)])

Agenda:
[item(s,   0,3,[r(np,1,vp)]),
 item(n,   3,4,[lex(ei)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]

Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
```

```
add item(s,0,3,[r(np,1,vp)])
no new item

Agenda:
[item(n,   3,4,[lex(ei)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]

Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
```

```
add item(n, 3,4,[lex(ei)])
new item(np,2,4,[r(det,3,n)])

Agenda:
[item(np,  2,4,[r(det,3,n)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
```

```
add item(np,2,4,[r(det,3,n)])
new item(vp,1,4,[r(v,2,np)])

Agenda:
[item(vp,  1,4,[r(v,2,np)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]

Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
```

```
add item(vp,1,4,[r(v,2,np)])
new item(s,0,4,[r(np,1,vp)])
Agenda:
[item(s,   0,4,[r(np,1,vp)]),
 item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
```

```
add item(s,0,4,[r(np,1,vp)])
no new item
Agenda:
[item(p,   4,5,[lex(met)]),
 item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
```

```
add item(p,   4,5,[lex(met)])
no new item
Agenda:
[item(det, 5,6,[lex(de)]),
 item(n,   6,7,[lex(stippen)])]
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
item(p,   4,5,[lex(met)]).
```

```
add item(det, 5,6,[lex(de)]).
no new item
Agenda:
[item(n,   6,7,[lex(stippen)])]
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
item(p,   4,5,[lex(met)]).
item(det, 5,6,[lex(de)]).
```

```
add item(n,   6,7,[lex(stippen)])
new item(np,5,7,[r(det,6,n)])
```

Agenda:

```
[item(np,5,7,[r(det,6,n)])]
```

Chart:

```
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
item(p,   4,5,[lex(met)]).
item(det, 5,6,[lex(de)]).
item(n,   6,7,[lex(stippen)]).
```

```
add item(np,5,7,[r(det,6,n)])
new item(pp,4,7,[r(p,5,np)])

Agenda:
[item(pp,4,7,[r(p,5,np)])]
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
item(p,   4,5,[lex(met)]).
item(det, 5,6,[lex(de)]).
item(n,   6,7,[lex(stippen)]).
item(np,  5,7,[r(det,6,n)]).
```

```
add item(pp,4,7,[r(p,5,np)])
new item(n,3,7,[r(n,4,pp)])  and item(vp,1,7,[r(vp,4,pp)])

Agenda:                                  Chart:
[item(n,3,7,[r(n,4,pp)]),                item(np,  0,1,[lex('Arie')]).
 item(vp,1,7,[r(vp,4,pp)])],             item(v,   1,2,[lex(ziet)]).
                                         item(vp,  1,2,[r(v)]).
                                         item(s,   0,2,[r(np,1,vp)]).
                                         item(det, 2,3,[lex(het)]).
                                         item(np,  2,3,[lex(het)]).
                                         item(vp,  1,3,[r(v,2,np)]).
                                         item(s,   0,3,[r(np,1,vp)]).
                                         item(n,   3,4,[lex(ei)]).
                                         item(np,  2,4,[r(det,3,n)]).
       ---                               item(vp,  1,4,[r(v,2,np)]).
item(pp,  4,7,[r(p,5,np)]).              item(s,   0,4,[r(np,1,vp)]).
                                         item(p,   4,5,[lex(met)]).
                                         item(det, 5,6,[lex(de)]).
                                         item(n,   6,7,[lex(stippen)]).
                                         item(np,  5,7,[r(det,6,n)]).
```

```
add item(n,3,7,[r(n,4,pp)])
new item(np,2,7,[r(det,3,n)])

Agenda:
[item(np,2,7,[r(det,3,n)]),
 item(vp,1,7,[r(vp,4,pp)])],

       ---
item(pp,  4,7,[r(p,5,np)]).
item(n,   3,7,[r(n,4,pp)]).
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
item(p,   4,5,[lex(met)]).
item(det, 5,6,[lex(de)]).
item(n,   6,7,[lex(stippen)]).
item(np,  5,7,[r(det,6,n)]).
```

```
add item(np,2,7,[r(det,3,n)])
new item(vp,1,7,[r(v,2,np)])
Agenda:
[item(vp,1,7,[r(v,2,np)]),
 item(vp,1,7,[r(vp,4,pp)])],
```

```
       ---
item(pp,  4,7,[r(p,5,np)]).
item(n,   3,7,[r(n,4,pp)]).
item(np,  2,7,[r(det,3,n)]).
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
item(p,   4,5,[lex(met)]).
item(det, 5,6,[lex(de)]).
item(n,   6,7,[lex(stippen)]).
item(np,  5,7,[r(det,6,n)]).
```

```
add item(vp,1,7,[r(v,2,np)])
new item(s, 0,7,[r(np,1,vp)])
Agenda:
[item(s, 0,7,[r(np,1,vp)]),
 item(vp,1,7,[r(vp,4,pp)])],
```

```
      ---
item(pp,  4,7,[r(p,5,np)]).
item(n,   3,7,[r(n,4,pp)]).
item(np,  2,7,[r(det,3,n)]).
item(vp,  1,7,[r(v,2,np)]).
```

```
Chart:
item(np,  0,1,[lex('Arie')]).
item(v,   1,2,[lex(ziet)]).
item(vp,  1,2,[r(v)]).
item(s,   0,2,[r(np,1,vp)]).
item(det, 2,3,[lex(het)]).
item(np,  2,3,[lex(het)]).
item(vp,  1,3,[r(v,2,np)]).
item(s,   0,3,[r(np,1,vp)]).
item(n,   3,4,[lex(ei)]).
item(np,  2,4,[r(det,3,n)]).
item(vp,  1,4,[r(v,2,np)]).
item(s,   0,4,[r(np,1,vp)]).
item(p,   4,5,[lex(met)]).
item(det, 5,6,[lex(de)]).
item(n,   6,7,[lex(stippen)]).
item(np,  5,7,[r(det,6,n)]).
```

```
add item(s,0,7,[r(np,1,vp)]); no new item
Agenda:                                          Chart:
[item(vp,1,7,[r(vp,4,pp)])]                      item(np,  0,1,[lex('Arie')]).
                                                 item(v,   1,2,[lex(ziet)]).
                                                 item(vp,  1,2,[r(v)]).
                                                 item(s,   0,2,[r(np,1,vp)]).
                                                 item(det, 2,3,[lex(het)]).
                                                 item(np,  2,3,[lex(het)]).
                                                 item(vp,  1,3,[r(v,2,np)]).
                                                 item(s,   0,3,[r(np,1,vp)]).
                                                 item(n,   3,4,[lex(ei)]).
                                                 item(np,  2,4,[r(det,3,n)]).
      ---                                        item(vp,  1,4,[r(v,2,np)]).
item(pp,  4,7,[r(p,5,np)]).                      item(s,   0,4,[r(np,1,vp)]).
item(n,   3,7,[r(n,4,pp)]).                      item(p,   4,5,[lex(met)]).
item(np,  2,7,[r(det,3,n)]).                     item(det, 5,6,[lex(de)]).
item(vp,  1,7,[r(v,2,np)]).                      item(n,   6,7,[lex(stippen)]).
item(s,   0,7,[r(np,1,vp)]).                     item(np,  5,7,[r(det,6,n)]).
```

```
add item(vp,1,7,[r(vp,4,pp)]) ALREADY EXISTS, add history!
Agenda:                                                  Chart:
[]                                                       item(np,  0,1,[lex('Arie')]).
                                                         item(v,   1,2,[lex(ziet)]).
                                                         item(vp,  1,2,[r(v)]).
                                                         item(s,   0,2,[r(np,1,vp)]).
                                                         item(det, 2,3,[lex(het)]).
                                                         item(np,  2,3,[lex(het)]).
                                                         item(vp,  1,3,[r(v,2,np)]).
                                                         item(s,   0,3,[r(np,1,vp)]).
                                                         item(n,   3,4,[lex(ei)]).
                                                         item(np,  2,4,[r(det,3,n)]).
      ---                                                item(vp,  1,4,[r(v,2,np)]).
item(pp,  4,7,[r(p,5,np)]).                              item(s,   0,4,[r(np,1,vp)]).
item(n,   3,7,[r(n,4,pp)]).                              item(p,   4,5,[lex(met)]).
item(np,  2,7,[r(det,3,n)]).                             item(det, 5,6,[lex(de)]).
item(vp,  1,7,[r(vp,4,pp),r(v,2,np)]).                   item(n,   6,7,[lex(stippen)]).
item(s,   0,7,[r(np,1,vp)]).                             item(np,  5,7,[r(det,6,n)]).
```

Item dat al bestaat

- Probeer toe te voegen:

  `item(vp,1,7,[r(vp,4,pp)])`

- Chart bevat:

  `item(vp,1,7,[r(v,2,np)])`

- Resultaat:
  - ⋆ Geen toevoging van een nieuw item;
  - ⋆ combineer geschiedenissen in bestaand item.

Prolog

```
parse(Sent) :-
    add_words_to_agenda(Sent,0,P,Agenda),
    process_agenda(Agenda),
    extract_tree(s,0,P,Tree).

process_agenda([]).

process_agenda([Item|Agenda]) :-
    chart_member_check(Item),!,
    process_agenda(Agenda).

process_agenda([Item|Agenda0]) :-
    chart_add(Item),
    findall(NewItem, completion(Item,NewItem), Agenda1),
    append(Agenda1, Agenda0, Agenda),
    process_agenda(Agenda).
```

Completion

- Alleen regels met 1 of 2 dochters

```
completion(item(D2,P1,P,_),item(Cat,P0,P,[r(D1,P1,D2)])) :-
   rule(Cat,[D1,D2]),
   chart_member(item(D1,P0,P1,_)).

completion(item(D1,P0,P1,_),item(Cat,P0,P,[r(D1,P1,D2)])) :-
   rule(Cat,[D1,D2]),
   chart_member(item(D2,P1,P,_)).

completion(item(D,P0,P,_), item(Cat,P0,P,[r(D)])) :-
   rule(Cat,[D]).
```

Terugvinden van parsebomen

- Samenpakken van geschiedenis van items

```
chart_member_check(item(Cat,P0,P,His0)) :-
    chart_delete(item(Cat,P0,P,His1)),
    append(His0,His1,His),
    chart_add(item(Cat,P0,P,His)).
```

Terugvinden van parsebomen

- Terugvinden van parseboom:

```
extract_tree(Cat,P0,P,b(Cat,Ds)) :-
    chart_member(Cat,P0,P,His),
    member(Single,His), % kies eentje
    extract_trees(Single,P0,P,Ds).

extract_trees(r(D),P0,P,[Tree]) :-
    extract_tree(D,P0,P,Tree).

extract_trees(r(D1,P1,D2),P0,P,[Tree1,Tree2]):-
    extract_tree(D1,P0,P1,Tree1),
    extract_tree(D2,P1,P, Tree2).

extract_trees(lex(Word),_,_,[w(Word)]).
```

Disambiguatie

- Ambiguïteit: meerdere mogelijke analyses voor een zin
- Tot nu toe: geaccepteerd dat het bestaat, maar verder niets mee gedaan
- Dit college: kiezen van de beste ontleding
- Centraal thema: beste ontleding is de meest waarschijnlijke ontleding

PCFG

- Probabilistic Context-Free Grammar;
- een CFG waarbij elke regel voorzien is van een waarschijnlijkheid.

Waarschijnlijkheid

- random variable X
- $P(X = x)$: de kans dat X waarde x heeft
- X: de eerstvolgende letter in een Nederlandse tekst
- $P(X = e) > P(X = u)$

Waarschijnlijkheid

- $$\sum_i P(X = x_i) = 1$$

- Voor grote n:

$$P(X = x_i) = \frac{C(X = x_i)}{n}$$

Voorwaardelijke waarschijnlijkheid

- $P(X = x|\text{extra informatie})$
- $P(X = u)$
- $P(X = u|\text{de vorige letter was een q})$

Voorwaardelijke kansen

- $P(x)$ als verkorte notatie voor $P(X = x)$
-
$$P(x|y) = \frac{P(x,y)}{P(y)}$$
-
$$p(x|y) = \frac{C(x,y)}{C(y)}$$

PCFG regels

- Waarschijnlijkheid van een regel $X \rightarrow D_1 \ldots D_n$, gegeven dat het een regel voor X moet zijn
- $P(X \rightarrow D_1 \ldots D_n | X)$
- dus waarschijnlijkheid van regels met zelfde moederknoop moet samen 1 zijn

Bijvoorbeeld

```
rule(s,[np,vp],  0.9).  rule(s,[vp], 0.1).

rule(np,[det,n1],0.8).  rule(np,[n1], 0.2).

rule(n1,[n],     0.7).  rule(n1,[n1,pp], 0.3).

rule(vp,[vp,pp], 0.4).  rule(vp,[v], 0.4).  rule(vp,[v,np],  0.2).

rule(pp,[p,np],  1.0).

lex(det,de, 0.5).   lex(det,het, 0.5).

lex(p,van,  0.9).   lex(p,met, 0.1).

lex(n,man,  0.2).   lex(n,vrouw, 0.4).  lex(n,mens, 0.4).

lex(v,slaapt,0.2). lex(v,slaat,0.3). lex(v,kust,0.2). lex(v,kus,0.3).
```

PCFG parseboom

- Waarschijnlijkheid van een parse: product van de waarschijnlijkheden van de regels

PCFG zin

- Som van de waarschijnlijkheden van de parses

PCFG

- Onder bepaalde voorwaarden:
  - ⋆

$$\sum_{l \in L} P(l) = 1$$

Motivatie voor PCFG

- Disambiguatie: kies de parseboom met hoogste waarschijnlijkheid
- Taalmodel: geeft waarschijnlijkheid aan zinnen (bijvoorbeeld voor spraakherkenning, OCR)
- Robuustheid: regels met lage waarschijnlijkheid zodat ook aan onverwachte (ongrammaticale) zinnen een structuur kan worden toegekend
- Leren: PCFG kunnen worden afgeleid uit corpora (unsupervised) en treebanks (supervised)

Parseren met PCFG

- Aanpak 1: voeg waarschijnlijkheid toe bij reconstrueren parseboom, kies beste uit alle parsebomen

- Aanpak 2: neem waarschijnlijkheid mee in de parser, en bouw alleen de beste parse

Efficiënt vinden van de beste parse

- `item(Cat,P0,P,His,Prob)`
- `His` is slechts één geschiedenis
- `Prob` is de waarschijnlijkheid
- `Prob` tot nu toe beste manier om dit item te bouwen

Completion

```
completion(item(D2,P1,P,_,Prob1),item(Cat,P0,P,r(D1,P1,D2),Prob)) :-
   rule(Cat,[D1,D2],Prob2),
   chart_member(item(D1,P0,P1,_,Prob3)),
   Prob is Prob1 * Prob2 * Prob3.
```

Samenpakken van geschiedenis van items

- Samenpakken van geschiedenis van items

```
chart_member_check(item(Cat,P0,P,His,Prob)) :-
    item(Cat,P0,P,_,Prob1),
    (   Prob1 >= Prob    % dus de nieuwe is slechter/even goed
    ->  true
    ;                    % dus de nieuwe is beter
        chart_delete(item(Cat,P0,P,_,Prob1),
        fail             % beschouw nieuwe als nieuw!
    ).
```

Truc met log

- Vermenigvuldigen van steeds kleinere breuken levert problemen op
- Bijv: 2.22507e-308, kleiner is niet te representeren met een 64-bit double
- Daarom: representeer breuken met -log
- $-log(2.22507e^{-308}) = 708.39642$
- Let op met vermenigvuldigen (wordt optellen)

Waar komen de waarschijnlijkheden vandaan?

- Leren
- Mogelijkheid 1: alleen op basis van tekst
- Mogelijkheid 2: op basis van treebank

Treebank

- Verzameling parsebomen
- Voor representatieve verzameling zinnen
- Elke parseboom wordt geacht de juiste te zijn

Treebank

- $P(X \rightarrow C_1 \ldots C_n | X)$ wordt geschat als:

$$\frac{C(X \rightarrow C_1 \ldots C_n)}{C(X)}$$

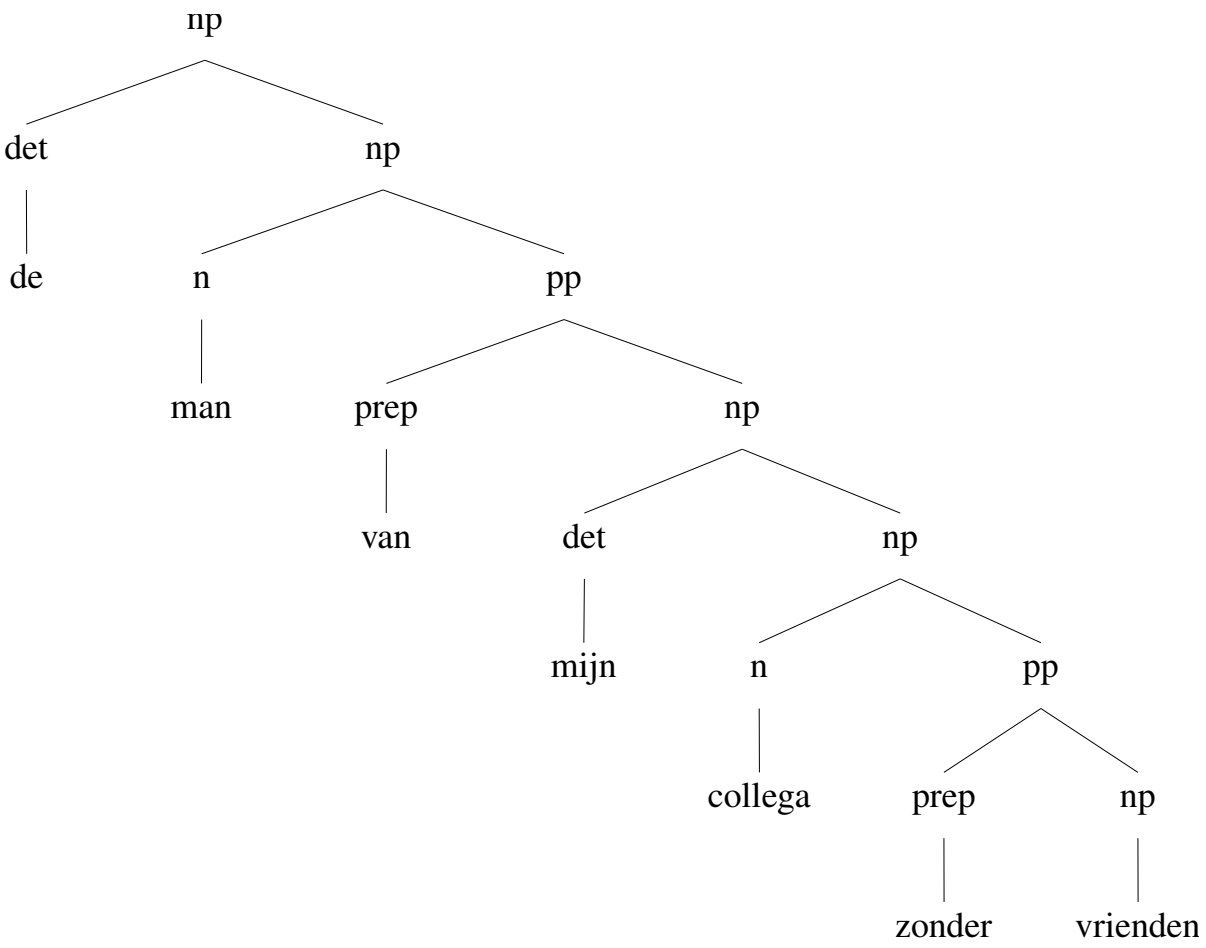
Evaluatie (1)

- Verdeling treebank:
  - ⋆ deel voor training (bijvoorbeeld 90%)
  - ⋆ deel voor testen (bijvoorbeeld 10%)
- Nooit testen op *bekend* materiaal

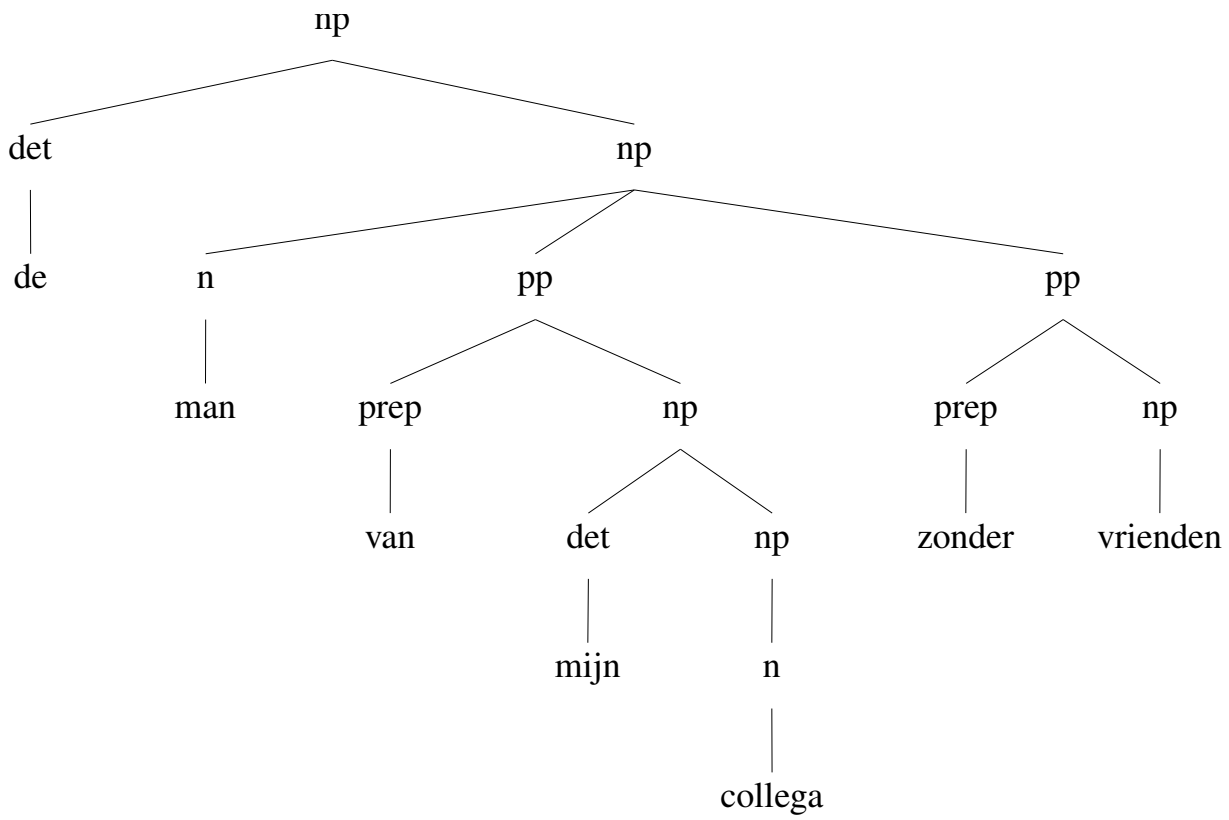
Evaluatie (2)

- Exact Match: voor hoeveel zinnen levert de parser de juiste parse op?
- Fijner instrument: maak onderscheid tussen slechtere en betere parses
- Hoeveelheid constituenten die correct zijn

Vergelijken van bomen



Vergelijken van bomen



Vergelijken van bomen

- np 0 7 np 4 7
  np 1 7 pp 5 7
  pp 2 7 np 3 7

- np 0 7 pp 5 7
  np 1 7 np 3 5
  pp 2 5

- Overlap: 3

- 3/5 of 3/6???

Precision, Recall

- Precisie: van wat je gevonden hebt, hoeveel is correct?
- Recall: van wat je moest vinden, hoeveel is correct?
- Voorbeeld

F-score

- Gewogen gemiddelde van precision (P) en recall (R)
- Betere f-score als precision en recall dichter bij elkaar
- F-score:

$$2 * \frac{P * R}{P + R}$$

PCFG parseboom

- Waarschijnlijkheid van een parse: Product van de waarschijnlijkheden van de regels
- Waarschijnlijkheid van een zin: Som van de waarschijnlijkheden van de parses

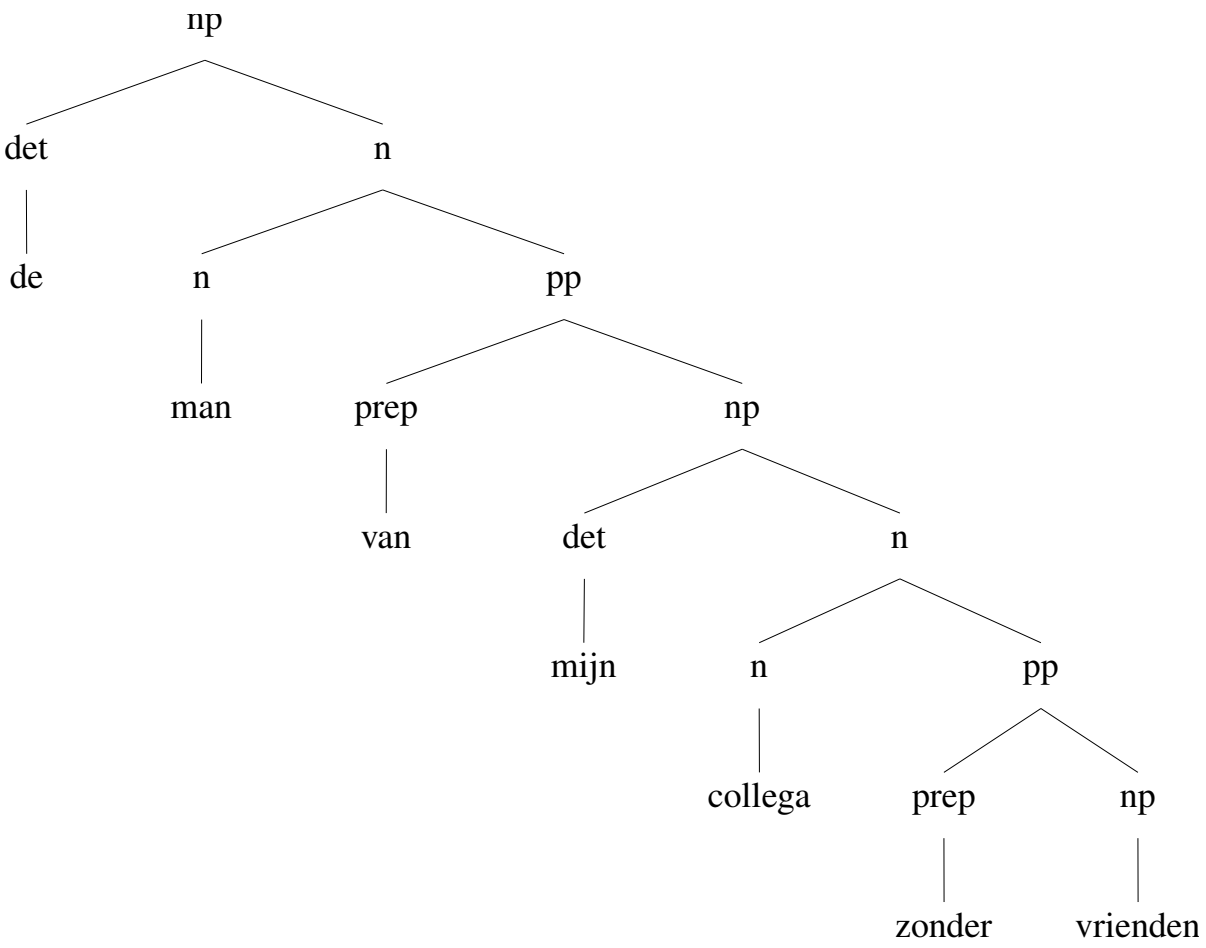
Motivatie voor PCFG

- Disambiguatie: kies de parseboom met hoogste waarschijnlijkheid
- Taalmodel: geeft waarschijnlijkheid aan zinnen (bijvoorbeeld voor spraakherkenning, OCR)
- Robuustheid: regels met lage waarschijnlijkheid zodat ook aan onverwachte (ongrammaticale) zinnen een structuur kan worden toegekend
- Leren: PCFG kunnen worden afgeleid uit corpora (unsupervised) en treebanks (supervised)

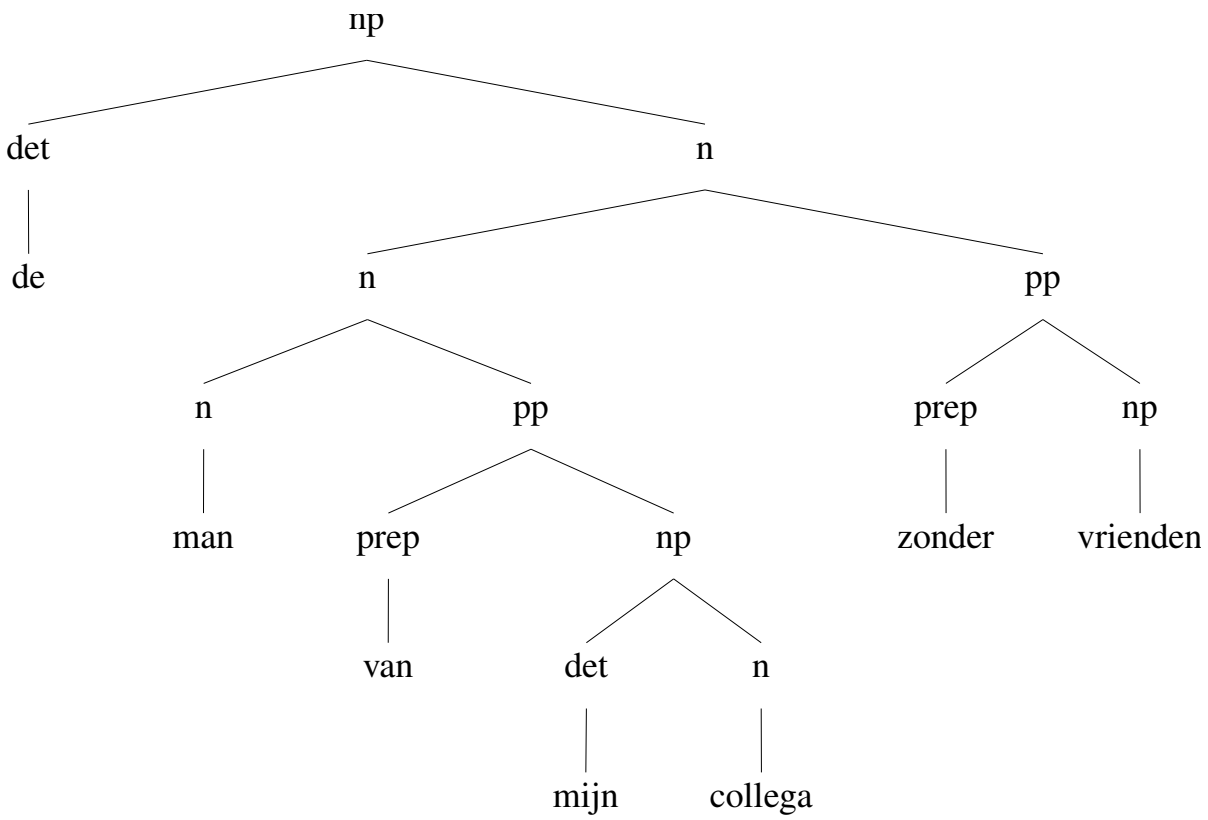
Beperkingen van PCFG

- Beperkingen van CFG . . .
- Sommige systematische amiguïteiten kunnen niet worden onderscheiden
- Voorkeuren: alleen *lokaal* tussen regels, geen globale voorkeuren

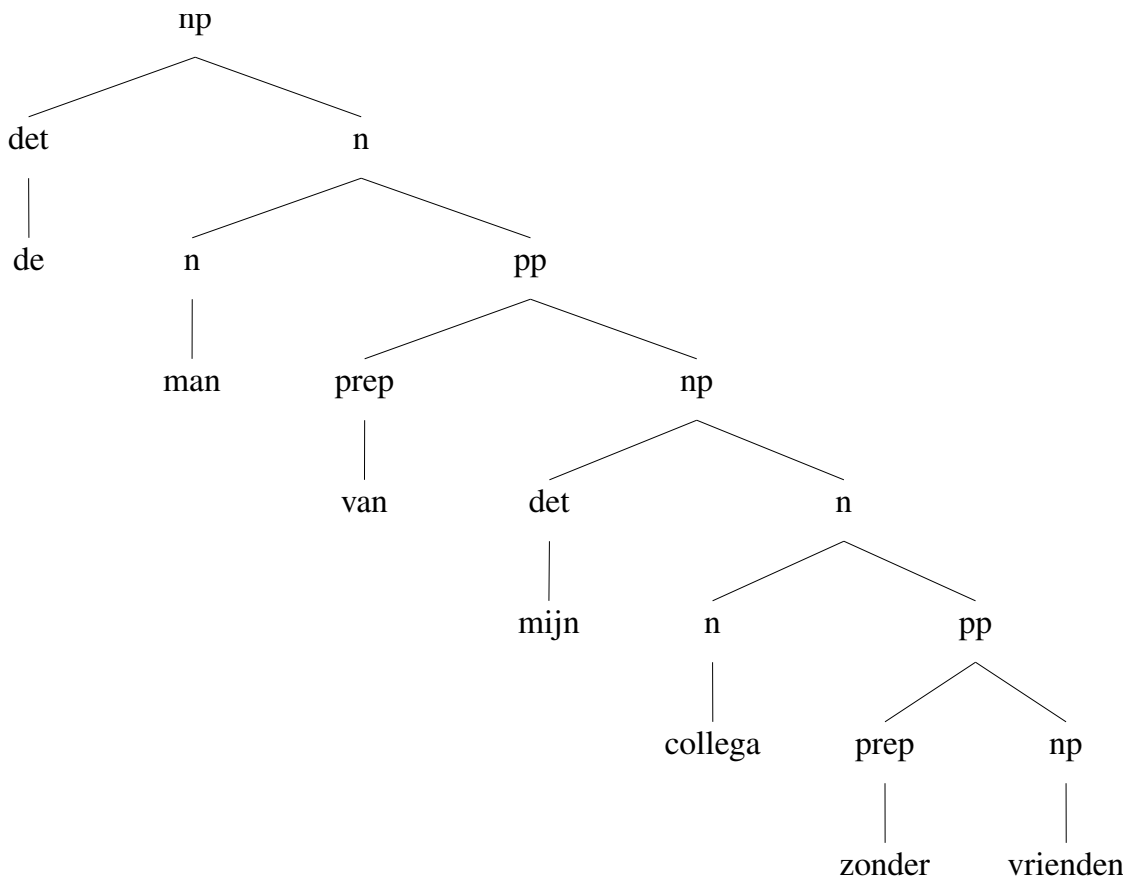
Beperkingen van PCFG



Beperkingen van PCFG

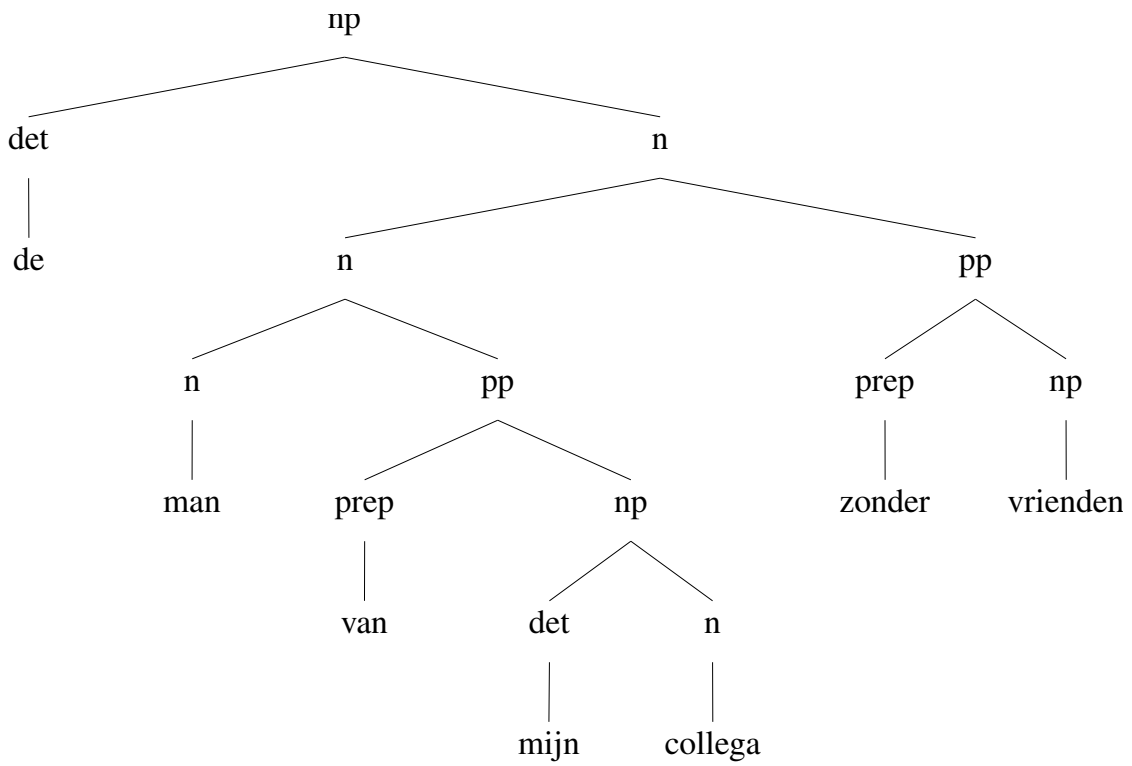


Tel regels parse 1




```
np --> det n          2
n  --> n pp           2
pp --> prep np        2
```

Tel regels parse 2




```
np --> det n          2
n  --> n pp           2
pp --> prep np        2
```

Alleen lokale voorkeuren kunnen worden uitgedrukt

- *PP met als hoofd* `van` *treedt vaker op bij* `n` *dan bij* `v`

- Soms kunnen zulke voorkeuren alsnog worden uitgedrukt, door de benodigde context informatie in de regels te coderen

```
pp(van)  --> p(van)  np      p(van)  --> van
pp(nvan) --> p(nvan) np      p(nvan) --> met        p(nvan) --> op ...
n  --> n pp(van)             n   --> n pp(nvan)
vp --> vp pp(nvan)           vp  --> vp pp(van)
```

- Dit kun je automatisch doen (*head annotation*)

Vermenigvuldigen van de regels

. . . om relevante informatie in de context lokaal beschikbaar te maken

- head annotation
- vergelijkbare andere trucs
- werkt behoorlijk goed voor Engels (f-score 86%)
- gespecialiseerde, rijkere, modellen werken nog iets beter

Beperkingen van treebank om PCFG te leren

- Regels die niet in treebank voorkomen
- Alternatieve technieken om waarschijnlijkheden te schatten (*smoothing*)

PCFG $\rightarrow$ PDCG?

- Kunnen we dezelfde techniek voor DCG gebruiken?
- Elke DCG regel associëren met waarschijnlijkheid?
- Korte antwoord: nee

Problemen

- Waarschijnlijkheid 'verdwijnt' als unificatie faalt
- Leren van waarschijnlijkheden op basis van een treebank levert niet de optimale waarschijnlijkheden
- Andere techniek: *random fields*; *log-linear*; *maximum entropy*
- Mist ook de andere nadelen van PCFG!
- Resulterende formalisme: Stochastic Attribute Value Grammar

Log-linear model

- Geef *features* voor disambiguatie (eigenschappen van parses)
- Training: ken een *gewicht* toe aan elk feature
  - ⋆ verhoog gewicht van features die in correcte parse vookomen
  - ⋆ verlaag gewicht van features die in incorrecte parse voorkomen
- Toepassen van het model:
  - ⋆ Voor elke parse, tel hoevaak ieder feature voorkomt
  - ⋆ Vermenigvuldig ieder aantal met corresponderend gewicht en sommeer
  - ⋆ Selecteer parse met grootste som

Model

$$\phi(t) = \sum_i \lambda_i f_i(t)$$

- t is een parse-boom
- $f_i(t)$ is de frequentie van feature i in t
- λ_i is het gewicht van feature i

Model

$$P(t|s) = \frac{\exp(\phi(t))}{Z^s}$$

$$Z^s = \sum_{u \in T_s} \exp(\phi(u))$$

- T_s: alle parsebomen voor zin s
- waarschijnlijkheid van x door te normalizeren ten opzichte van alle andere parses *voor dezelfde zin*
- om de beste te kiezen hoeven we alleen ϕ uit te rekenen

Trainen

- Verzin goede features i
- Geen beperkingen!
- Er bestaan algoritmes die vervolgens de beste λ_i bepalen, gegeven training data

Features

- r(Regel)
- f(Woord,Cat)
- r2(Regel,N,RegelN)
- r3(Regel,N,WoordN)

Features

- subj_topic, non_subj_topic
- subj_rel, non_subj_rel
- extraction_from_a_pp
- long_distance_dep, non_long_distance_dep
- mf(Cat1,Cat2)
- parallel

Features

- dep(HdCat,Rel,ArgCat)
- dep(HdCat,Rel,ArgCat,ArgWord)
- dep(HdCat,HdWord,Rel,ArgCat,ArgWord)

Voorbeelden Alpino

- Alpino Treebank:
  - ⋆ krantentekstgedeelte (cdbl) van Eindhoven corpus
  - ⋆ 145.000 words
  - ⋆ handmatig gecorrigeerde syntactische annotaties
  - ⋆ annotaties volgens CGN (Corpus Gesproken Nederlands)

Features met laagste gewichten

```
-0.0707213 h1(long)
-0.0585366 f2(was,noun)
-0.0507852 f2(tot,vg)
-0.0497879 h1(decap(not_begin))
-0.0494901 s1(extra_from_topic)
-0.0411195 r3(np_det_n,2,was)
-0.0410466 f2(op,prep)
-0.0372584 f2(kan,noun)
-0.0337606 h1(skip)
```

Features met hoogste gewichten

```
0.0741717 f2(en,vg)
0.064064  dep35(en,vg,hd/obj1,prep,tussen)
0.0549897 f2(word,verb(passive))
0.0461192 r2(non_wh_topicalization(np),1,np_pron_weak)
0.039418  s1(subj_topic)
0.0387447 dep23(pron(wkpro,nwh),hd/su,verb)
```

Resultaten disambiguatie

- Training: `cdbl`-deel van de Alpino treebank (145,000 woorden met dependency structures)
- Test: deel van Trouw 2001
- Model moet de beste parse kiezen uit maximaal 2000 concurrerende parses
- Accuratesse: proportie van correcte genoemde dependency relations

Results Parse Selection

	accuracy %
baseline	75.88
oracle	94.86
model	90.93

Samenvatting tot dusver

- Automatische analyse van natuurlijke taal; computer natuurlijke taal laten begrijpen
- Syntactische structuur noodzakelijk belangrijk onderdeel
- CFG, DCG definieren grammaticale regels
- Parser berekent voor gegeven zin en grammatica vervolgens mogelijke syntactische structuren
- Ambiguïteit
- PCFG geeft mogelijkheid om beste parse te vinden
- Stochastic Attribute Value Grammar om waarschijnlijkheden te introduceren in bijvoorbeeld DCG ($\longrightarrow$ Alpino)

Omgekeerde probleem: bepalen van grammaticale zin voor een gegeven betekenis

- Generator berekent voor gegeven grammatica en gegeven betekenis de mogelijke syntactische structuren
- Meerdere manieren om dezelfde betekenis te verwoorden
- Statistisch *fluency* model: kies de 'vloeiendste' uiting
- . . .