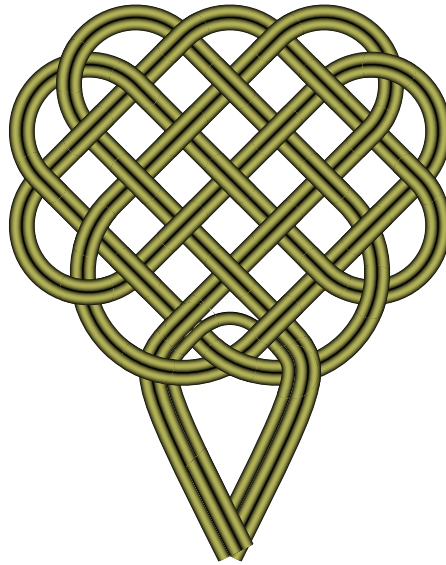


Alpino: accurate, robust, wide coverage computational analysis of Dutch

Gertjan van Noord
University of Groningen



Alpino: accurate, robust, wide coverage parsing of Dutch

Joint work with:

Leonoor van der Beek

Gosse Bouma

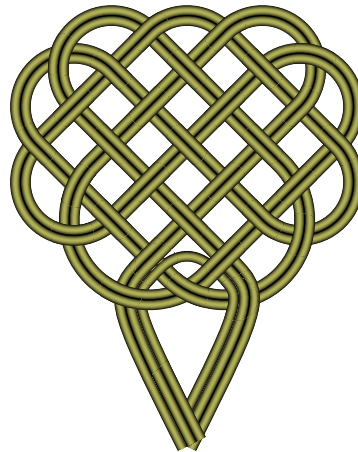
Jan Daciuk

Rob Malouf

Robbert Prins

Begona Villada

...



Alpino

- Sophisticated linguistic analysis (HPSG)
- Care about disambiguation (MaxEnt)
- Care about practical efficiency
- Corpus-based evaluation methodology

Overview

- Background and overview
- Error Mining for linguistic engineering
- Disambiguation 1: parse selection with log-linear model
- Disambiguation 2: incorporating selection restrictions
- Efficiency: Learning Efficient Parsing

Overview

- Background and overview
- Error Mining for linguistic engineering
- Disambiguation 1: parse selection with log-linear model
- Disambiguation 2: incorporating selection restrictions
- Efficiency: Learning Efficient Parsing

Theme: corpora, corpora, corpora

Origin: OVIS

- Spoken Dialogue System for Timetable Information
- 1998: Formal Evaluation NWO
- Comparison with DOP (Scha and Bod)

	WA	CA	cpu
• DOP	76.8	75.2	7011
Grammar-based	84.1	83.0	5524

Origin: OVIS

- Spoken Dialogue System for Timetable Information
- 1998: Formal Evaluation NWO
- Comparison with DOP (Scha and Bod)

	WA	CA	cpu
• DOP	76.8	75.2	7011
Grammar-based	84.1	83.0	5524

FC Groningen - Ajax 1-0

Alpino background

- lexical resources
- grammatical resources
- parser
- dependency structures
- evaluation

The Alpino Lexicon

- Lexical information is crucial
 - ★ subcategorization frames
- The lexicon is a mapping from words to *tags*
 - ★ Compact representation (*perfect hashing* FSA)
- Each *tag* is mapped to (one or more) signs
- This mapping is organised in inheritance network
- Tags combine lexical information and inflection

Lexical resources

- Large full form lexicon
 - ★ 190.000 proper names (persons, organizations, locations, misc)
 - ★ 75.000 nouns
 - ★ 37.000 adjectives
 - ★ 30.000 verbs
 - ★ 11.000 multi-word-units
 - ★ 20.000 misc
- Special rules for named entities:
 - ★ temporal expressions
 - ★ dates
 - ★ numerical expressions and amounts
 - ★
- Large set of heuristics to guess category of unknown words

Unknown word heuristics

- Pita, Peter Jan van Warmerdam
- karma, ancien régime, body mass index
- HELP, usa
- Italie, zó
- boterwetgeving, boter-wetgeving

More unknown word heuristics

- op- en terugbellen, land- en tuinbouw
- regering-Kok, Donald Duck-verhaal, science fiction-schrijver
- nummer 1-hit, artikel 12-status, oer-rock & roll
- wachtenden, verwijze
- animositeit, abusievelijk
- ...

Lexical Analysis

- Lookup each word in (full-form) lexicon
- Treat unknown words
- Filter irrelevant tags
 - ★ Cooccurrence restrictions filter impossible tags
 - ★ HMM-tagger filters unlikely tags

Lexical Type Filter

- Add a verb selecting a PP⟨*prep*⟩ only if *prep* is in the input string as well
- Similarly for verbs with a separable particle (*bel hem op*)
- . . . and various other rules
- Later: train POS-tagger to filter out *unlikely* lexical types in a given *context*

Example

(1) Mercedes zou haar nieuwe model gisteren hebben aangekondigd
Mercedes would her new model yesterday have announced
Mercedes would have announced its new model yesterday

- 395 tags
- 27 tags survive rules
- 13 tags survive HMM-tagger
- 13 tags \longrightarrow 34 signs (vs. 55 or even 2713 signs)

Specific Rules *and* General Constraints

- Linguist: capture generalizations; state general constraints only once
- Parser: specific rules; state as much constraining information as possible
- Combination: Constructionalist HPSG

Grammar Rules

- yesterday: 801 rules
- Grammar rules are instantiations of various *structures*
- ★ hd-compl-struct
- ★ hd-det-struct
- ★ hd-mod-struct
- ★ hd-filler-struct
- ★ hd-extra-struct
- ★ . . .
- *and* include very specific information

Grammar Structures

- Grammar structures are organized in an inheritance network
- Structures are associated with various *principles*
- ★ head-feature-principle
- ★ valence-principle
- ★ filler-principle
- ★ extraposition-principle
- ★

Example

```
grammar_rule(vp_arg_v(np),VP, [Arg, V] ) :-
    vp_arg_v_np_struct(VP,Arg,V).
```

```
vp_arg_v_np_struct(VP,Arg,V) :-
    Arg => np,
    vp_arg_v_struct(V,Arg,VP).
```

```
vp_arg_v_struct(V,Arg,VP) :-
    VP => vproj, V => vproj,
    VP:eps1 <=> V:eps1, VP:eps2 => no,
    VP:eps3 <=> V:eps3, V:haspre => yes,
    add_mf(VP,Arg,V), Arg:sel =?> to_left,
    allow_wh_in_situ(Arg,VP),
    hd_comp_struct(V,Arg,VP).
```

```
hd_comp_struct(H,Cmp,M) :-
    H:sc <=> _,
    H:ccat0 <=> Cat,
    projected_hd_struct(H, [Cmp], [], [], [], [], [], [], [], [], [], M, Cat).
```

Example (continued)

```
projected_hd_struct(H,Cmps,Prts,Adjs,Dets,Apps,Fils,Misc,Exs,MExs,Predms,M,Cat) :-  
    struct(H,Cmps,Prts,Adjs,Dets,Apps,Fils,Misc,Exs,MExs,Predms,M),  
    hd_dt_p(Cat,H,M,Adjs,Dets,Apps,Predms).
```

```
struct(H,Cmps,Prts,Adjs,Dets,Apps,Fils,Misc,Exs,MExs,Predms,M) :-  
    head_feature_p(H,M),  
    dip_tags_p(H,M),  
    valence_p(H,Cmps,Prts,M),  
    filler_p(H,Fils,M),  
    extra_p(H,[Cmps,Prts,Adjs,Dets,Apps,MExs,Misc,Predms],Exs,M),  
    m_extra_p(H,[Cmps,Prts,Adjs,Dets,Apps,Exs,Misc,Predms],MExs,M).
```

```

grammar_rule(VP_ARG_V(NP),
  [
    TAGS      D
    SC         E
    EXS       F
    MEXS      G
    VFORM     H
    SLASH     I
    VSLASH    J
    SUBJ      K
    RIGHTX    L
    PRO_DEPS  N
    E_DEPS    O
    EPS1      P
    EPS2      no
    HASWH     Q
    EPS3      R
    MF        (D3 [
      WH      W
      SEL     to_left
      EXS     F1
      MEXS    G1
    ] A2)
    CLEFT     C2
    DT        D2
    CCAT0     E2
    CCAT      F2
    CAPPS     G2
    APPS      H2
    CDETS     I2
    DETS      J2
    CMODS     K2
    MODS      L2
    CPREDMS   M2
    PREDMS    N2
  ],
  (D3,
  [
    HASPRE    yes
    TAGS      D
    SC        (D3|E)
    EXS       Q2
    MEXS      R2
    VFORM     H
    SLASH     I
    VSLASH    J
    SUBJ      K
    RIGHTX    L
    PRO_DEPS  N
    E_DEPS    O
    EPS1      P
    HASWH     Q
    EPS3      R
    MF        A2
    CLEFT     C2
    DT        D2
    CCAT0     E2
    CCAT      F2
    CAPPS     G2
    APPS      H2
    CDETS     I2
    DETS      J2
    CMODS     K2
    MODS      L2
    CPREDMS   M2
    PREDMS    N2
  ])):-
  vproj
wappend0(R2,G1,G,100),
wappend0(Q2,F1,F,100) ,
trig_nondif(W,NWH,V2,V2) ,

```

```

when(V2,?(W,NWH),GRAMMAR :(W = NWH ; W = RYWH(YWH, W2, X2, Y2, Z2), Q = YES)) .

```

```

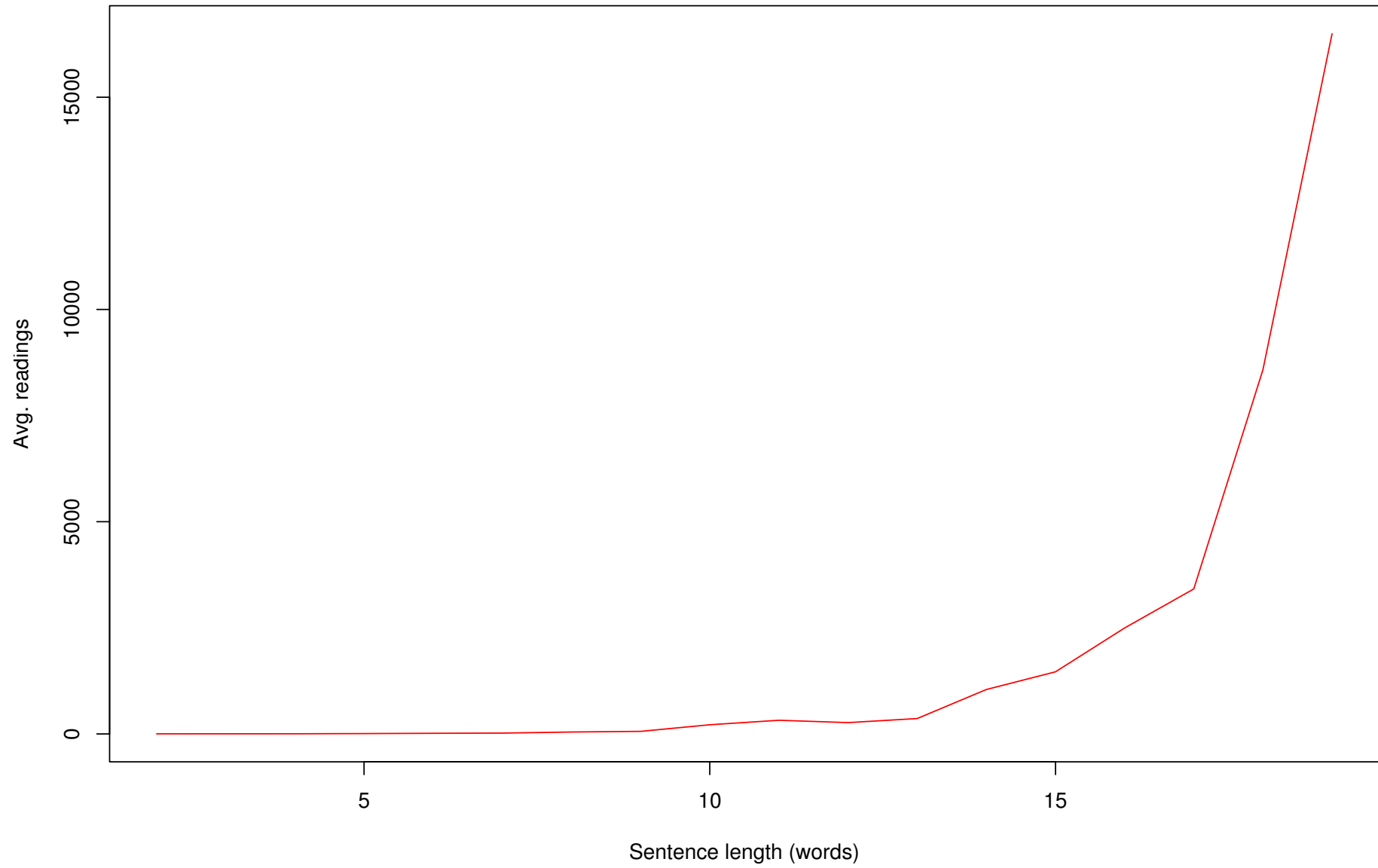
grammar_rule(vp_arg_v(np),
             vproj(_42784,_42785,_42786,_42787,_42788,_42780,_42756,_42791,
                 _42792,_42793,_42794,_42795,_42796,_42797,_42798,_42799,no,_42801,_42802,
                 [np(_42698,_42699,_42700,_42701,_42694,_42703,_42704,_42705,_42706,_42707,
                     _42708,_42709,_42710,sel(0,1,1,1),_42761,_42737,_42714,_42715,_42716,_42717,
                     _42718,_42719,_42720,_42721,_42722,_42723,_42724,_42725,_42726,_42727,_42728,
                     _42729,_42730,_42731,_42732)|_42734] ,_42804,_42805,_42806,_42807,_42808,_42809,
                     _42810,_42811,_42812,_42813,_42814,_42815,_42816) ,
                 [np(_42698,_42699,_42700,
                     _42701,_42694,_42703,_42704,_42705,_42706,_42707,_42708,_42709,_42710,
                     sel(0,1,1,1),_42761,_42737,_42714,_42715,_42716,_42717,_42718,_42719,
                     _42720,_42721,_42722,_42723,_42724,_42725,_42726,_42727,_42728,_42729,
                     _42730,_42731,_42732) ,
                     vproj(_42558,yes,_42560,_42787,[np(_42698,_42699,
                         _42700,_42701,_42694,_42703,_42704,_42705,_42706,_42707,_42708,_42709,
                         _42710,sel(0,1,1,1),_42761,_42737,_42714,_42715,_42716,_42717,_42718,
                         _42719,_42720,_42721,_42722,_42723,_42724,_42725,_42726,_42727,_42728,
                         _42729,_42730,_42731,_42732)|_42788] ,_42766,_42742,_42791,_42792,_42793,
                         _42794,_42795,_42570,_42797,_42798,_42799,_42574,_42801,_42802,_42734,
                         _42578,_42805,_42806,_42807,_42808,_42809,_42810,_42811,_42812,_42813,
                         _42814,_42815,_42816)  ])).

```

Parser

- Left-corner Parser with Memoization and Goal Weakening
- Delayed evaluation for recursive constraints
- Parse Forest: compact representation of all parses
- Disambiguation: select best parse from parse forest
- Robustness: do something useful if no full parse is available

Number of Parses



If no full parse is available

- Top category: maximal projection (NP, VP, S, PP, AP . . .)
- Often: not a full parse
 - ★ fragmentary input, ungrammatical input, . . .
 - ★ omissions in the grammar, dictionary, . . .

Partial parse results

- Parser finds all instances of top category *anywhere* in input
- Find best sequence of non-overlapping parses

Partial parse results

- Parser finds all instances of top category *anywhere* in input
- Find best sequence of non-overlapping parses

Soms zes plastic bekers tegelijk , in een kartonnen dragertje

Sometimes six plastic cups at the same time, in a cardboard retainer

[Soms zes plastic bekers] [tegelijk] [,] [in een kartonnen dragertje]

Is this useful?

- It depends . . .
- Often: **yes**
- ★ partial parse is correct (fragmentary input)
- ★ OVIS: important to recognize temporals and locatives
- ★ information extraction does not need full parses
- ★ . . .

Examples

- Fantastisch dus . *Fantastic, therefore.*
- Iedereen toch tevreden . *Everybody happy nonetheless.*
- Tijd is schaars . iedereen heeft haast . *Time is scarce. everybody is in a hurry.*
- 14 Hoe lang duurde de oorlog tussen Irak en Iran ? *14 How long took the war between Iraq and Iran?*
- SKOPJE Een buitenwijk van de Macedonische hoofdstad Skopje wordt onder de voet gelopen door miljoenen miljoenpoten . *SKOPJE Part of the Macedonian capital Skopje is being run over by millions of . . .*

Examples

- SKOPJE Een buitenwijk van [de Macedonische hoofdstad Skopje] wordt onder de voet gelopen door miljoenen miljoenpoten .
- Raymann is laat Tante Esselien ontvangt [Boris Dittrich , fractievoorzitter van D66] .
- [Voetballer Alexi Lalas] wordt genoemd (te veel aan testosteron) , alsmede [tennisster Mary Joe Fernandez] .
- Deelnemers onder anderen [burgemeester Meijer van Zwolle] , projectontwikkelaar Peter Ruig rok ...
- [CNV-voorzitter Doekle Terpstra] spreekt van het ' meest

Dependency Structures

- Provide a *grammar independent* level of representation
- Annotation format from *Corpus of Spoken Dutch project (CGN)*
- Alpino Treebank url: <http://www.let.rug.nl/%7Evannoord/trees>
- Evaluation
- Detailed Annotation Manual:
<http://www.let.rug.nl/%7Evannoord/Lassy/>
- Demo:
<http://www.let.rug.nl/%7Evannoord/bin/alpino>

Evaluation

- Compare dependency structure found by the parser to a *gold standard* dependency structure, verified by linguist
- Standard test-set: Alpino treebank
 - ★ 7153 sentences from cdbl-part of Eindhoven corpus
 - ★ manually verified dependency structures

Current results

- version-1
 - ★ 90.06 % CA: proportion of correct labeled dependencies
 - ★ 43.11 % exact: proportion of sentences with correct parse
 - ★ 20 seconds per sentence
- version-2
 - ★ 89.26 % CA: proportion of correct labeled dependencies
 - ★ 41.83 % exact: proportion of sentences with correct parse
 - ★ 3.8 seconds per sentence

Overview

- Background and overview
- Error Mining for linguistic engineering
- Disambiguation 1: parse selection with log-linear model
- Disambiguation 2: incorporating selection restrictions
- Efficiency: Learning Efficient Parsing

Error Mining for Linguistic Engineering

- Goal: improve grammar and dictionary
- Test suites (sets of hand-crafted examples) \implies problems must be anticipated
- Treebanks \implies much too small
- Instead: use unannotated material

Goal: improve Grammar and Dictionary

- Apply the system to large set of sentences
- Analyse sentences with *missing* parses
- Find words and word sequences that occur (much) more often in these sentences

Error Mining

- Error Mining Metric
- Results:
 - ★ Linguistic Examples
 - ★ Increase of Coverage

Corpora

- Various newspapers 1994-2002 (Trouw, NRC, AD, Volkskrant, Parool)
- Other material: Wikipedia, Mediargus
- Sentences up to 20 (30) words (with time-out)
- $\geq 2M$ sentences, $\geq 40M$ words, $\geq 200M$ chars
- Exploits Linux cluster of RuG HPC

Metric (1)

- *full* parse: a parse spanning the whole sentence
- $C(w)$: frequency of word w
- $C(w|\text{fail})$: frequency of word w in sentences which fail to parse
- compute *suspicion* for all words w :

$$S(w) = \frac{C(w|\text{fail})}{C(w)}$$

Coverage

- For this material: 91–96%
- An suspicion significantly above 9 percent is interesting

1.000	7	l'd
1.000	9	aangroei
1.000	9	aanzoek
1.000	7	adoreert
1.000	8	afkeur
1.000	21	afroep
1.000	7	après
1.000	7	berge
1.000	7	einmal

Metric (2)

- Often, words are problematic only in certain contexts
- $C(w_i \dots w_j)$: frequency
- $C(w_i \dots w_j | \text{fail})$: frequency in failed parses
- $S(w_i \dots w_j) = \frac{C(w_i \dots w_j | \text{fail})}{C(w_i \dots w_j)}$

0.100	716	via	<i>via</i>
0.843	15	via via	<i>indirectly</i>
0.084	165	waard	<i>worth</i>
1.000	10	de waard	<i>the host</i>

Metric (3)

- Consider longer sequences only if more suspicious than corresponding shorter ones:
 - ★ $S(w_h w_i \dots w_j w_k) > S(w_h w_i \dots w_j)$ *and*
 - ★ $S(w_h w_i \dots w_j w_k) > S(w_i \dots w_j w_k)$

Sort results

- Prefer most suspicious forms
- Prefer most frequent forms
- Here: sort by suspicion

Browse most suspicious forms

1.000	82	! enz.	chess
1.000	8	! gevolgd	chess
1.000	7	, zo 12-17u	announcement
1.000	15	, zo 13-17u	
1.000	316	- fl.	new books
1.000	12	; 127 blz.	
1.000	10	; 142 blz.	
1.000	14	; 143 blz.	
1.000	19	16x27	checkers
1.000	7	2Klaver pas	bridge
1.000	8	4 t/m 12 jaar	announcement (theater, ..)
1.000	17	I have	foreign language

Browse most suspicious forms (2)

1.000	7	de huisraad	
1.000	7	Maar eerlijk is eerlijk	
1.000	9	en noem maar	
1.000	18	is daar een voorbeeld	
1.000	7	par excellence	
1.000	7	In vroeger tijden	
1.000	7	dan ten hele	
1.000	7	hele gedwaald	
1.000	7	het libido	
1.000	9	kinds af	
1.000	8	tenzij .	<i>unless .</i>

List problematic examples

@ Vroeger was het nee , tenzij .

@ Nou ja , het is een Nee , tenzij . . .

erlandse wetgever staat een ' nee , tenzij .

@ Orgaandonatie tenzij ik de nagel van mijn ree

@ Officieel is het : ja , tenzij .

@ Anderen : nee , tenzij .

g gebied tussen ja , mits en nee , tenzij .

@ Geen jacht tenzij .

@ U zult niet doden , tenzij .

Sagot & de la Clergerie (2006)

- Unproblematic forms are blamed if they co-occur with problematic forms
- Try to shift blame to forms which are suspicious in other sentences
- Initially, suspicion of an observation of a form in given sentence of length n : $1/n$
- Suspicion of a form is the mean of the suspicion of all its observations
- Suspicion of an observation of a form is the suspicion of its form, normalized by the sum of the suspicions of all forms that occur in the sentence

de Kok (CLIN 2009)



- Provide evaluation framework
- Compare scoring methods
- Ignore low suspicion forms
- Add larger N-gram if significantly more suspicious than its shorter variants
- Provide graphical interface

Results from Mediargus

- Telkens hij [Everytime he]
- (had er AMOUNT) voor veil [(had AMOUNT) for sale]
- (om de muren) van op te lopen [to get terribly annoyed by]
- Ik durf zeggen dat [I dare to say that]
- op punt stellen [to fix/correct something]
- de daver (op het lijf) [shocked]
- (op) de tippen (van zijn tenen) [being very careful]
- ben fier dat [am proud of]
- Nog voor halfweg [still before halfway]
- (om duimen en vingers) van af te likken [delicious]

Mining results viewer

File Edit

Forms

Scoring: Suspicion * f(uniqSuspSents)

Regex: []te[]

Score	Form
98.2725	nog aan te duiden .
56.2463	spelers aan te duiden
48.9198	speler aan te duiden
38.8484	Kaarten te koop
33.8489	Tickets te koop
32.962	@ te brengen
32.8109	Af te raden .
32.1785	wegens te oud .
32.153	En te zeggen
32.1209	van te winnen .
31.7385	op punt te houden
31.6868	tegen te pruttelen
31.6272	op te voetballen
31.4516	zullen te zien krijgen
30.9767	@ te lijf
30.4327	pan uit te swingen
29.6742	Nog te zien in
29.4642	in te scannen .
28.6062	kijk en te keur
28.5908	geruststelling te weten
28.2901	Zoals te verwachten ,
27.7959	(te verkrijgen
27.6099	bij te maken dat
26.9288	in te zeggen .
26.7904	van te benutten

Remove form

Form information

Suspicion: 0.697713

Frequency: 41

Unparsable frequency: 41

Unique sentences: 41

Unparsable sentences

Match all unparsable sentences

Regex:

@ BAAIGEM Zeshonderd stamboekpaarden te kijk en te keur

@ Daarnaast staan zowat honderd auto's van allerhande merken te kijk en te keur " , zegt Seba

@ De achthoekige glazen zuil van twee meter hoog en een doorsnede van een meter staat heer

@ De knalgele beestjes stonden samen met honderden andere gevleugelde collega's drie dagen

@ De sportieve , trendy vrijetijdskledij staat weer te kijk en te keur op Boo.com .

@ De werken hingen te kijk en te keur in het wijkcentrum Aan 't Spoor .

@ Een foto van dit comité hangt te kijk en te keur op de LVZ-stand .

@ Een wc-bril voor het hele gezin , pralines met foie gras , en een welriekende toiletpot : het zijn

@ Er staan dit jaar zo'n 650 stalletjes te kijk en te keur .

@ Gedekte tafels staan er te kijk en te keur : de klassieke feesttafel , een ontbijttafel , zomer- er

@ Geitjes te kijk en te keur

@ Geitjes te kijk en te keur in 't Roodhuis

@ Hafingers en andere paarden staan er te kijk en te keur .

@ Het kruim van het Vlaamse melkvee , de mooiste kalveren en de beste fokvarkens staan op 1

@ In de verkeersvrije straten was het laveren tussen de kraampjes waar de Oostenrijkse culinair

@ In Galerie Thomas , Opperstraat 60 , zijn tekeningen van Anny Roggeman , Alena Straetmans

@ Interessant zijn de restaurants , waar karkassen van gehalveerde geiten te kijk en te keur han

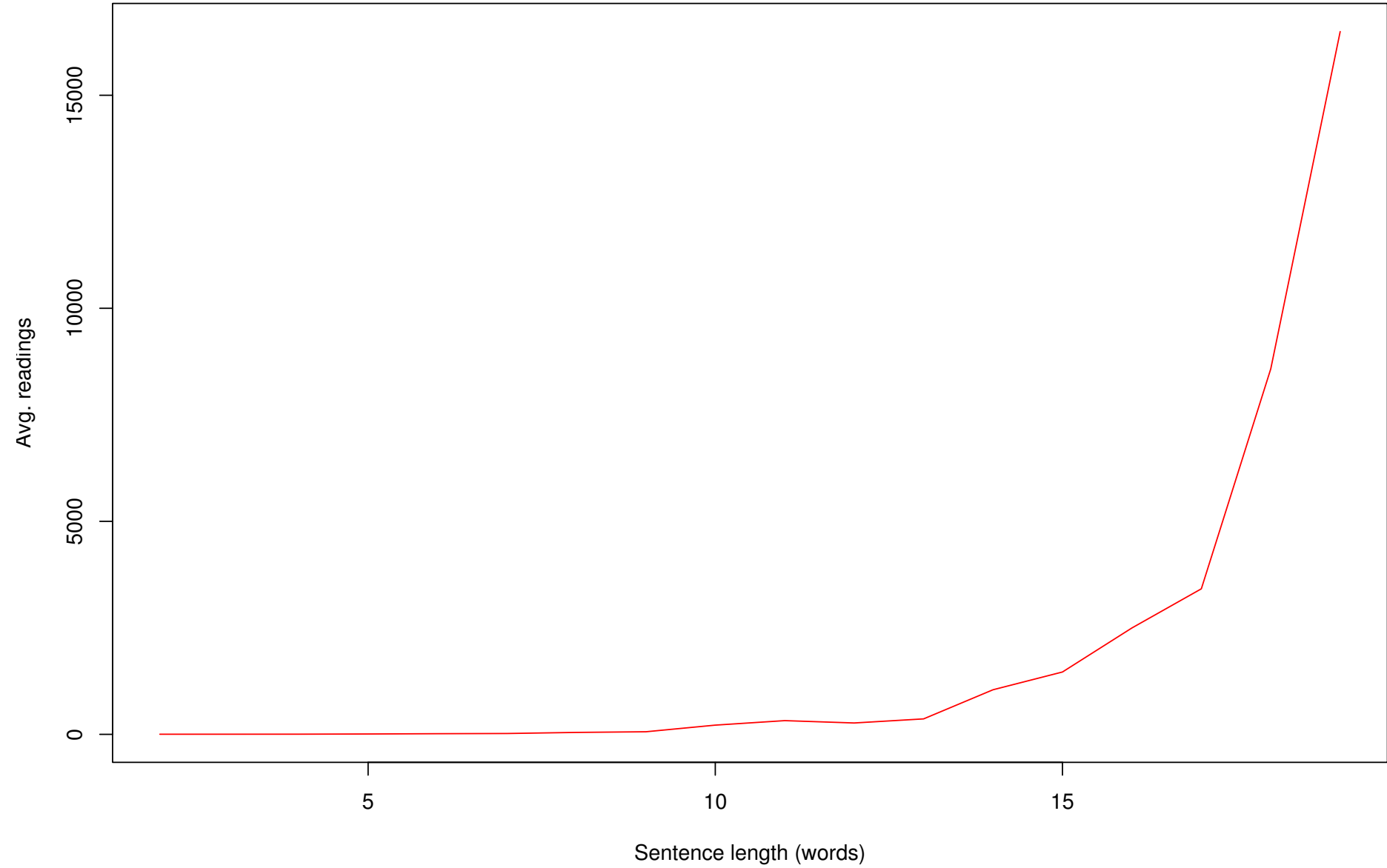
@ Kanaries te kijk en te keur

@ Kleinvee te kijk en te keur

Overview

- Background and overview
- Error Mining for linguistic engineering
- Disambiguation 1: parse selection with log-linear model
- Disambiguation 2: incorporating selection restrictions
- Efficiency: Learning Efficient Parsing

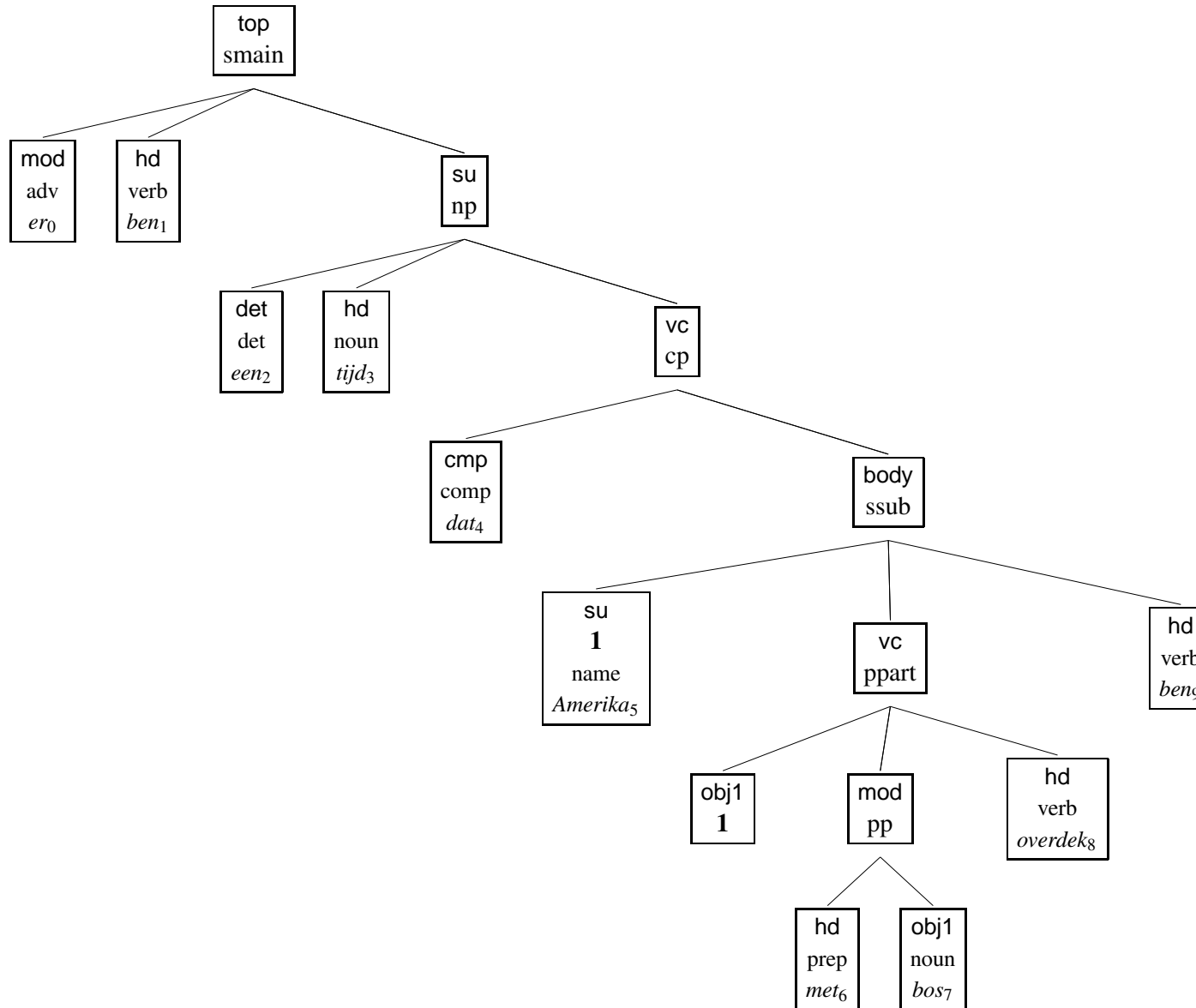
Ambiguity in Alpino



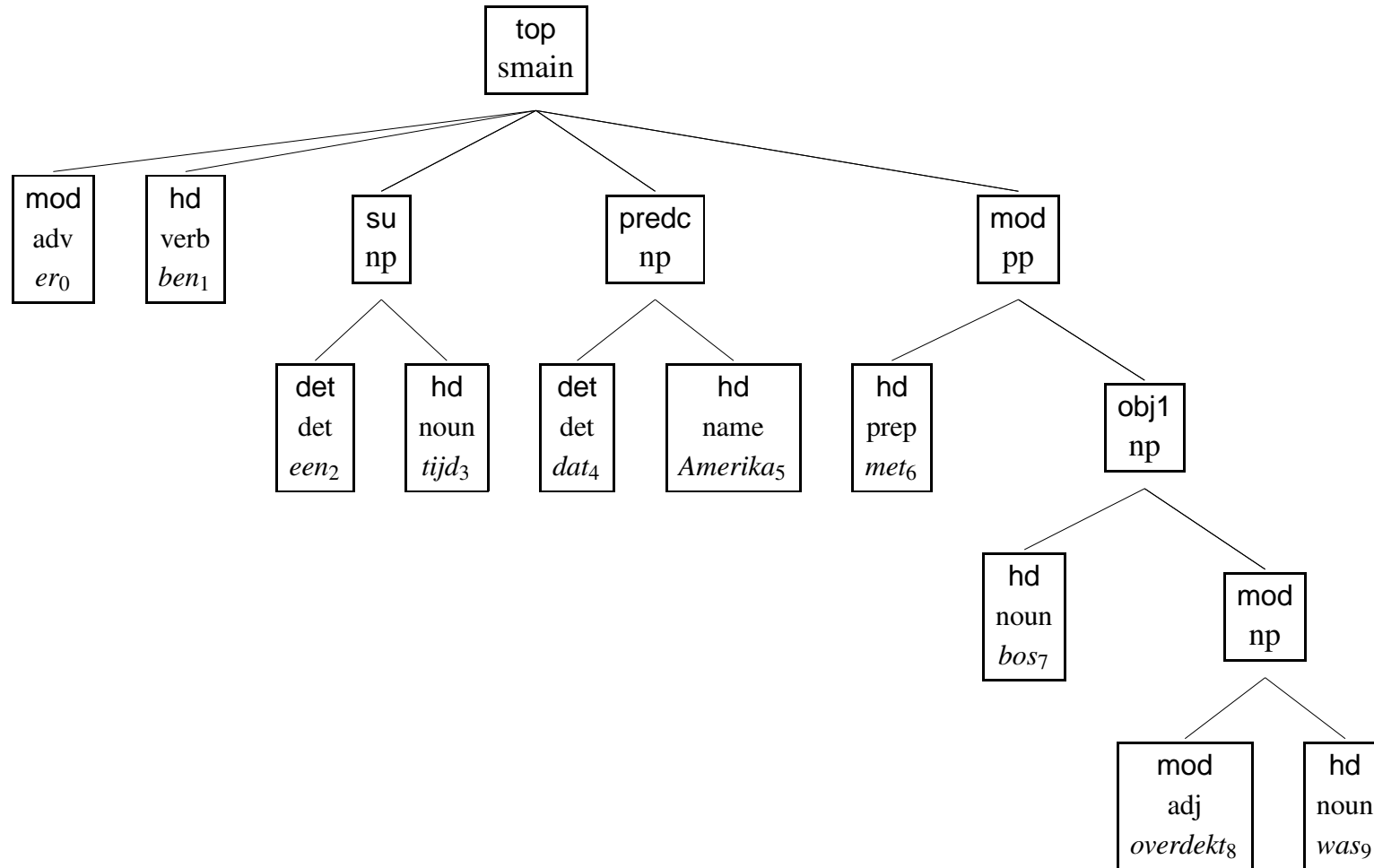
Ambiguity

- the expected lexical and structural ambiguities
- many, many, many unexpected, absurd, ambiguities
- many *don't care* ambiguities
- longer sentences have millions of parses

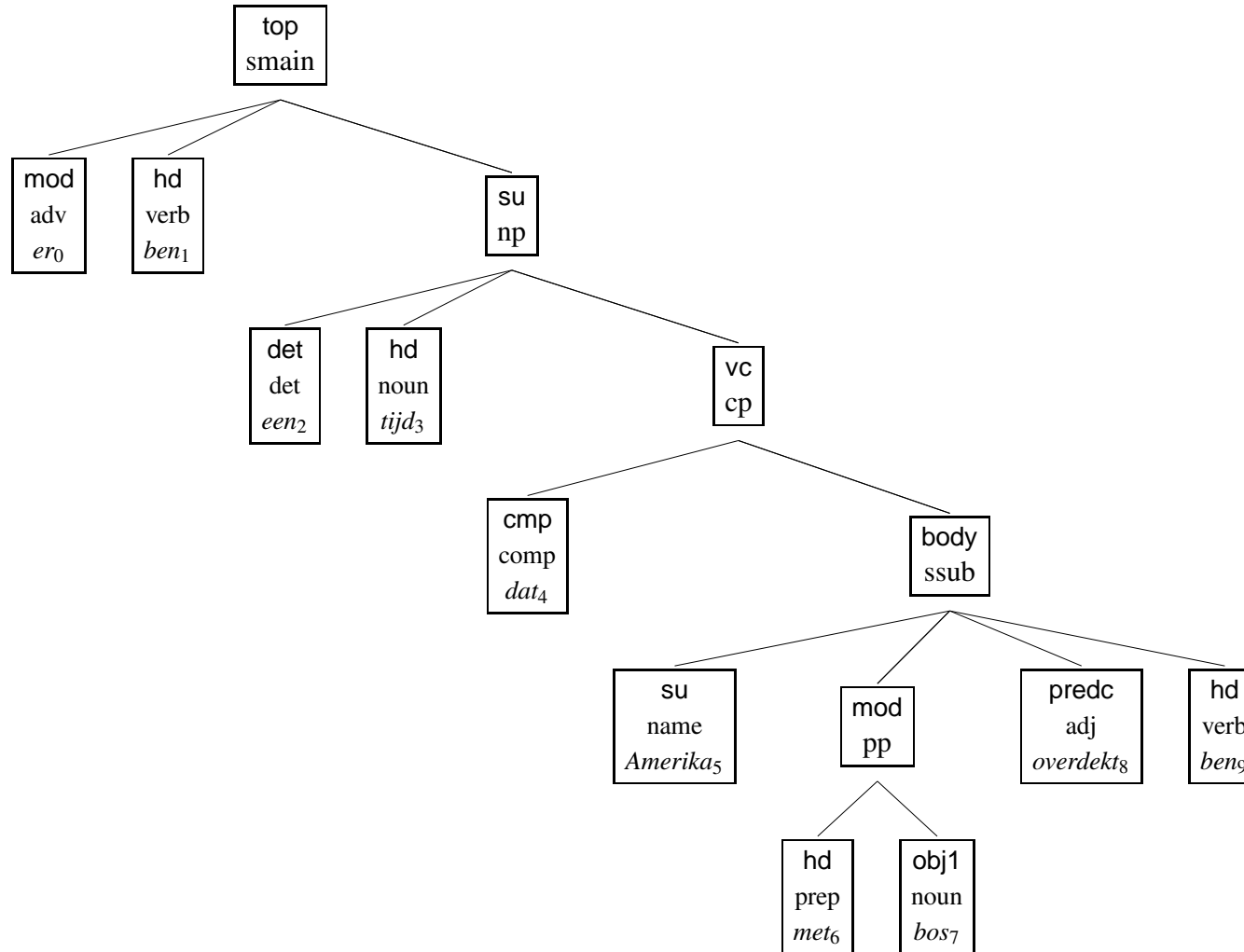
Er was een tijd dat Amerika met bossen overdekt was



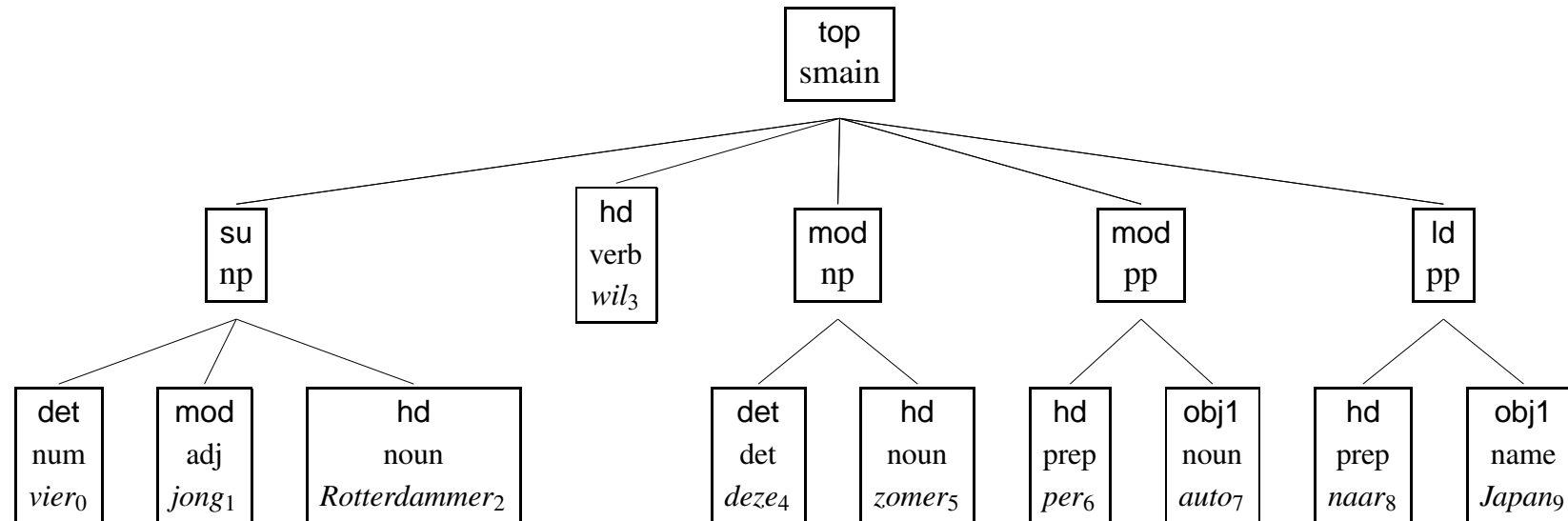
Er was een tijd dat Amerika met bossen overdekt was



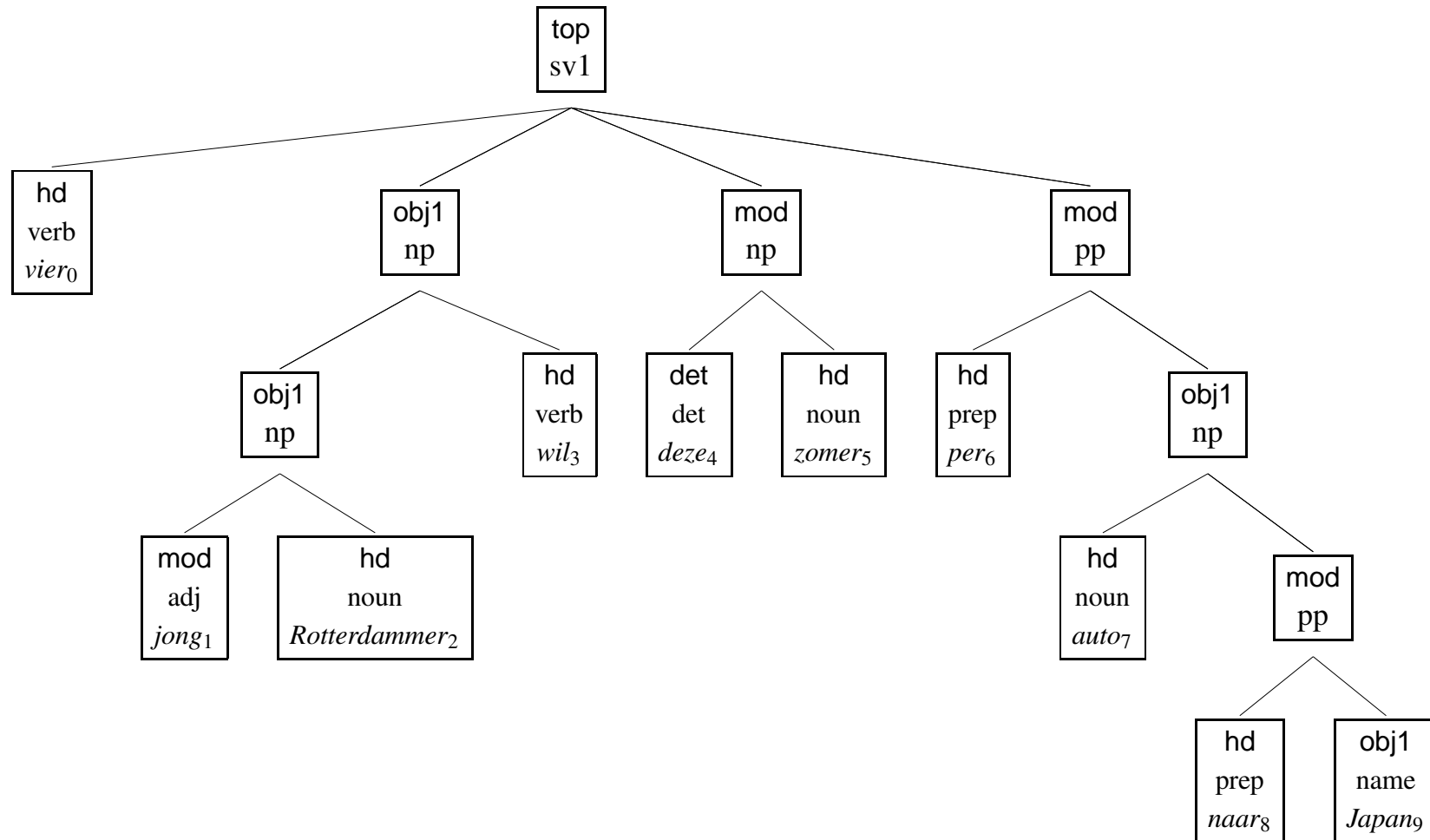
Er was een tijd dat Amerika met bossen overdekt was



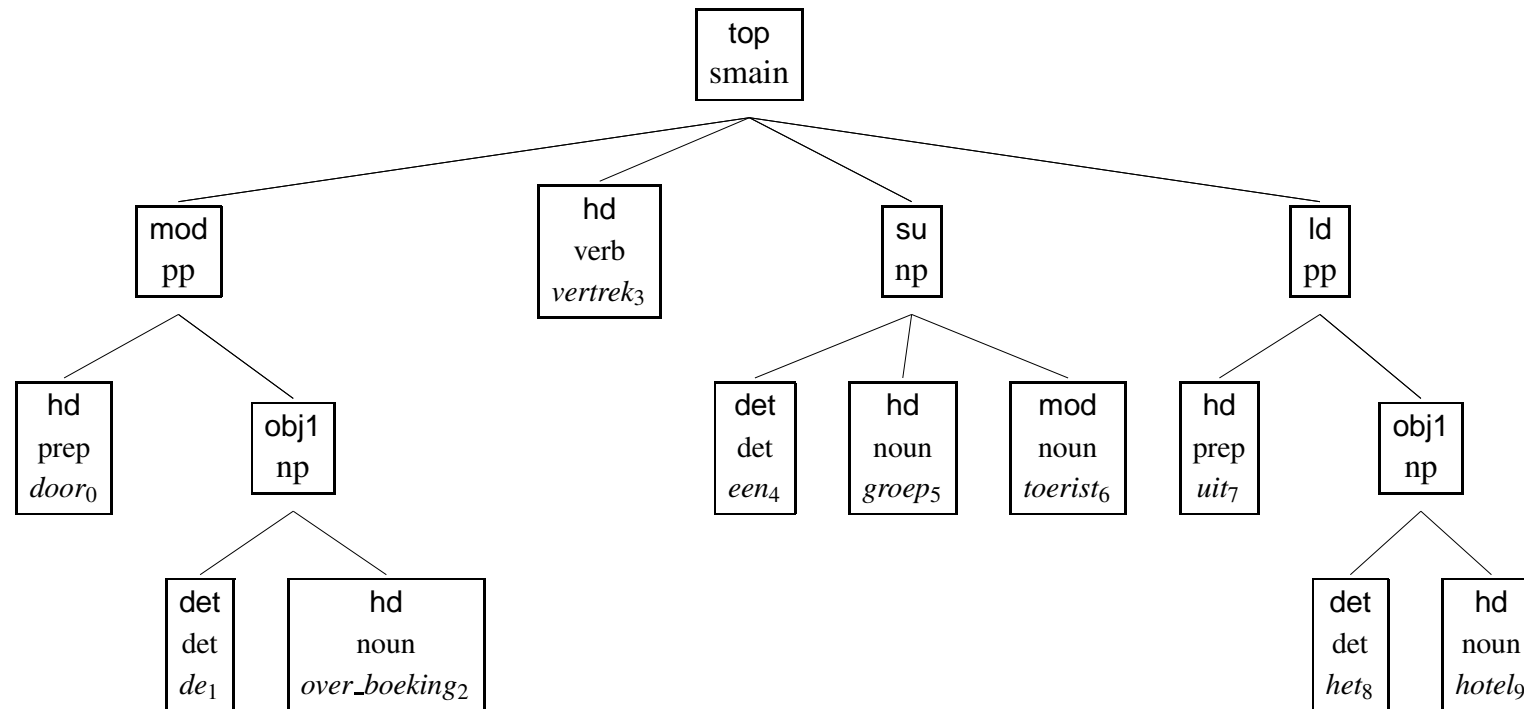
Vier jonge Rotterdammers willen deze zomer per auto naar Japan



Vier jonge Rotterdammers willen deze zomer per auto naar Japan

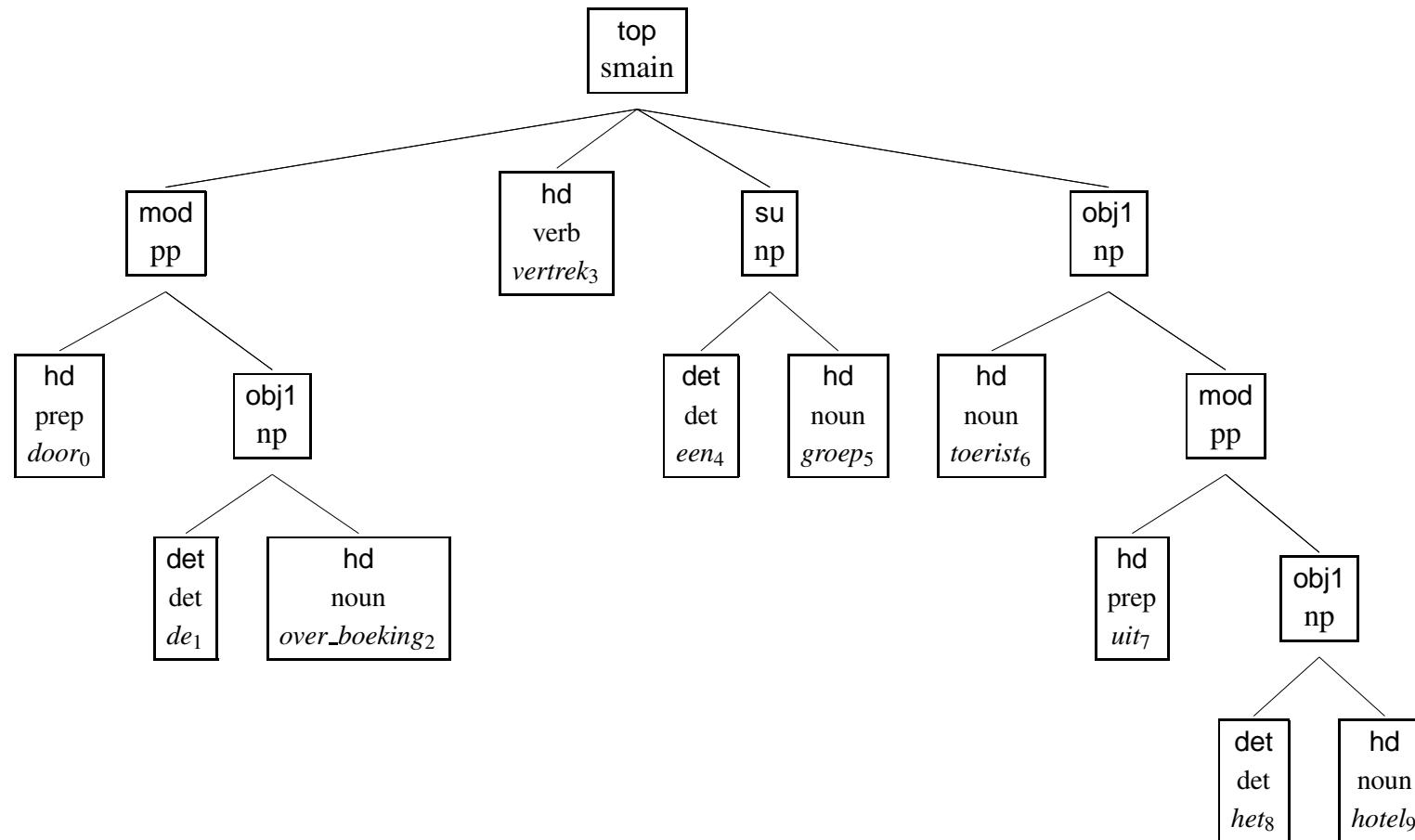


Door de overboeking vertrok een groep toeristen uit het hotel



- Zempléni: unambiguously literal sentence
- Alpino: 13 parses

Door de overboeking vertrok een groep toeristen uit het hotel



Disambiguation in Alpino

- Syntactic analysis
 - ★ Use POS-tagger to remove unlikely lexical categories
- select intended parse from parse forest
 - ★ Maxent disambiguation model
 - ★ best-first beam-search algorithm

Maxent Disambiguation Model

- Identify *features* for disambiguation: arbitrary characteristics of parses
- Training the model: assign a *weight* to each feature, by
 - ★ increase weights of features in the correct parse
 - ★ decrease weights of features in incorrect parses
- Applying the model:
 - ★ For each parse, sum weights of features occurring in it
 - ★ Select parse with highest sum

Training

- Requires a corpus of correct and incorrect parses
- Alpino Treebank:
 - ★ newspaper-part (cdbl) of Eindhoven corpus
 - ★ 145.000 words
 - ★ manually checked syntactic annotations
 - ★ annotations as proposed in CGN (Corpus of Spoken Dutch)

Problem: Efficiency

- Need access to *all* parses of a sentence
 - ★ training the model
 - ★ applying the model
- Number of parses can be exponential
- In practice, number of parses can be Really Big

Solution 1: Use Parse Forest

- Geman and Johnson (2002)
- Miyao and Tsujii (2002)
- Train model directly from forest
- Best parse can be computed efficiently from forest

Drawbacks

- Strong *Locality Requirement* on features
- Features are no longer arbitrary characteristics of parses
- Non-local features can be locally *encoded* in grammar, but
 - ★ Complicate grammar dramatically
 - ★ Reduce parser efficiency

Solution 2: Use Sample for training

- Osborne (2000): representative **small** sample of parses
- Take into account relative quality of parses during training
- Provides solution for cases where treebank structures are of different nature than parses
- Training material consists of parser output (annotated with quality score)

Construct Training Material

- Construct the first 1.000 parses of each sentence from the corpus
- For each parse, count the frequency of all features
- Compare each parse with the gold standard, and assign corresponding score
- Each parse is represented by a vector of feature frequencies and a quality score

Features

- Describe arbitrary properties of parses
- Need not be independent of each other
- Can encode a variety of linguistic (and other) preferences
- Linguistic Insights!

Features templates

r1(Rule)	Rule has been applied
r2(Rule,N,SubRule)	The N-th daughter of Rule is constructed by SubRule
r2_root(Rule,N,Word)	The N-th daughter of Rule is Word
r2_frame(Rule,N,Frame)	The N-th daughter of Rule is a word with subcat frame Frame
r3(Rule,N,Word)	The N-th daughter of Rule is headed by Word
mf(Cat1,Cat2)	Cat1 precedes Cat2 in the <i>mittelfeld</i>
f1(Pos)	POS-tag Pos occurs
f2(Word,Pos)	Word has POS-tag Pos
h(Heur)	unknown word heuristic Heur has been applied

Dependency feature templates

dep35(Sub,Role,Word)	Sub is the Role dependent of Word
dep34(Sub,Role,Pos)	Sub is the Role dependent of a word with POS-tag Pos
dep23(SubPos,Role,Pos)	a word with POS-tag SubPos is the Role dependent of a word with POS-tag Pos

Some non-local features

- In coordinated structure, the conjuncts are parallel or not
- In extraction structure, the extraction is local or not
- In extraction structure, the extracted element is a subject
- Constituent ordering in *mittelfeld*
 - ★ pronoun precedes full np
 - ★ accusative pronoun precedes dative pronoun
 - ★ dative full np precedes accusative full np

Features indicating bad parses

-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 indicating good parses

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)

Results Parse Selection

- cdb1-part of Alpino treebank (145,000 words annotated with dependency structures)
- ten-fold cross-validation
- Model should select best parse for each sentence out of maximally 1000 parses per sentence
- accuracy: proportion of correct named dependencies

Results Parse Selection

	accuracy %
baseline	59.9
oracle	88.3
model	83.3
rate	82.4
exact	56

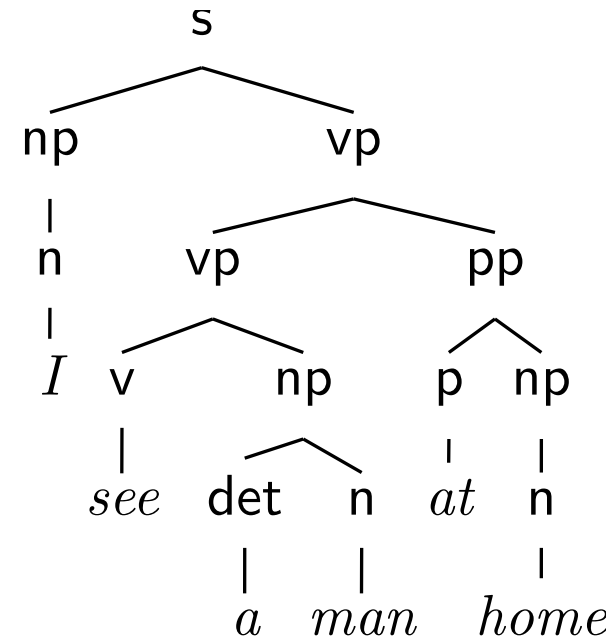
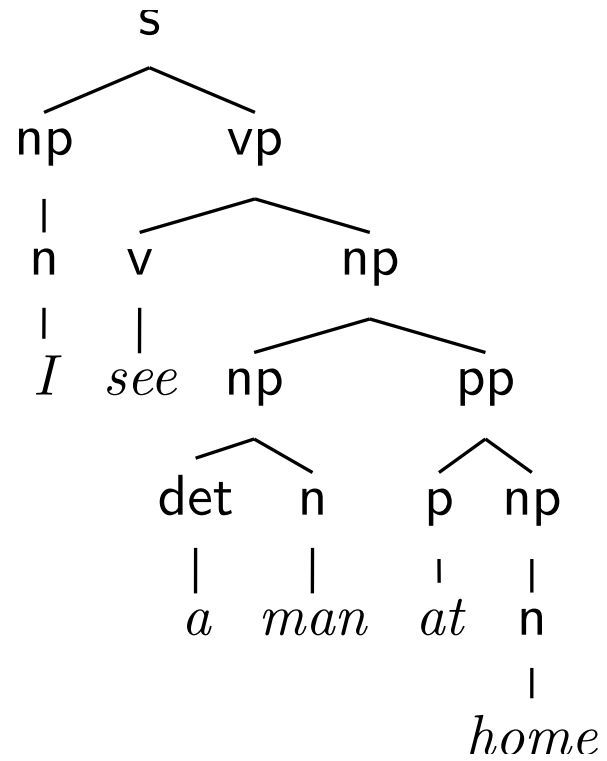
Remaining Problem

- How to find the best parse efficiently
- Dynamic programming algorithm not directly applicable
- Our contribution: *beam search* algorithm
 - ★ Parse Forest with larger domain of locality
 - ★ Beam Search Algorithm

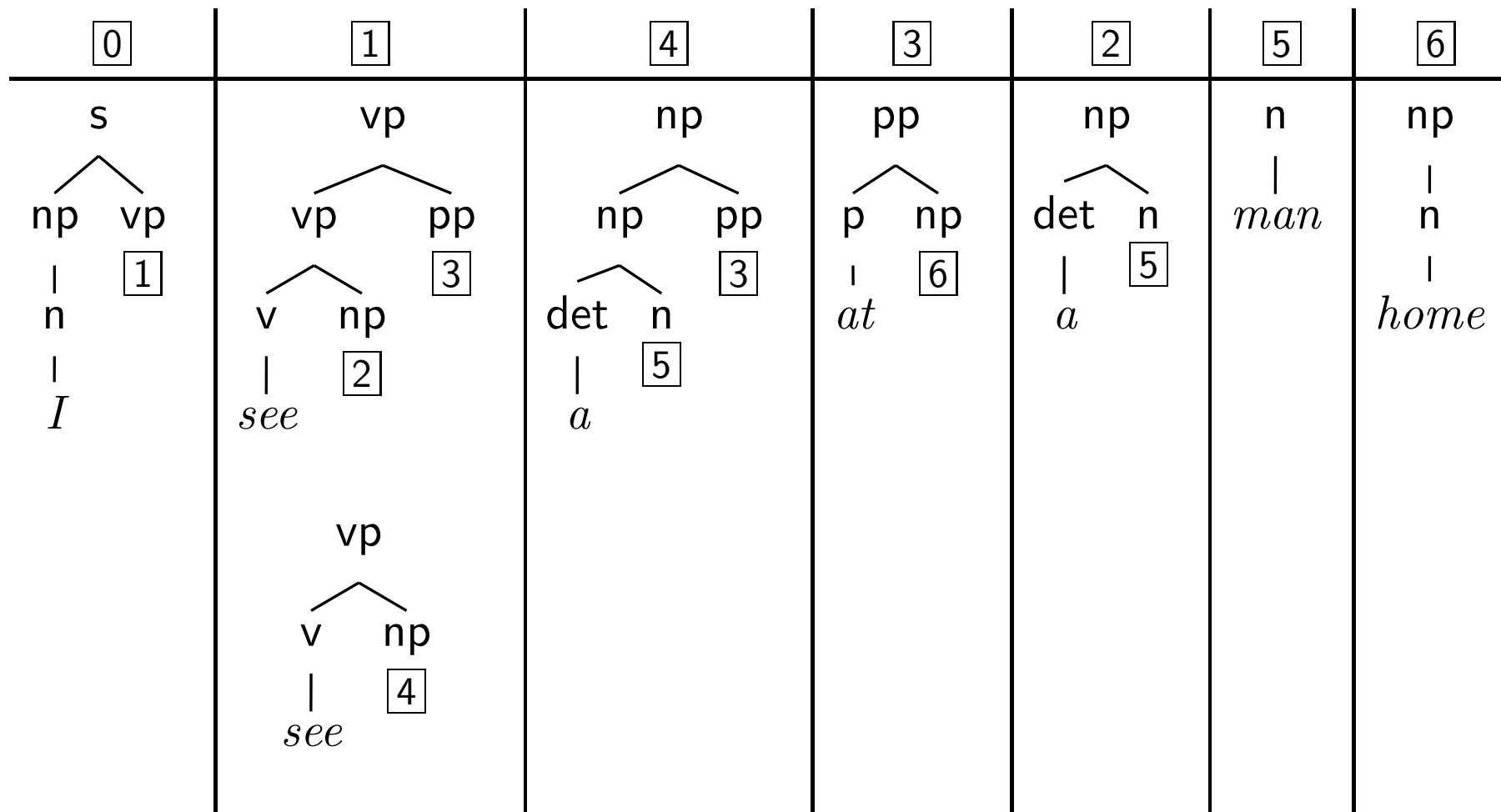
Parse Forest

- Left-corner parser
- Matsumoto et al. (1983); van Noord (1997)
- Chunks of parse forest are relatively large
 - ★ left-corner projections
- Explained by means of example

Example Parses



Example Parse Forest



Recover Best Parse from Parse Forest

- Order indexes
- For each index, construct *best* parse
 - ★ Using best parse of indexes constructed earlier

Properties

- Requires monotonicity
- *if sub-parse c_1 is better than c_2 , then it should be better in all contexts*
- Non-local features violate this restriction
- Solution: keep track of all b best parses per index

Beam search

- Order indexes
- For each index, construct *best* b parses
 - ★ Using all combinations of best b parses of indexes constructed earlier

Properties

- Larger beam \rightarrow better parse
- Smaller beam \rightarrow faster
- No guarantee that best parse is found
- But: in practice results are very good

Results beam search

beam	CA	CPU	out
1	84.82	0.14	0
2	85.18	0.18	0
4	85.36	0.28	0
8	85.49	0.39	0
16	85.60	0.56	0
32	84.87	0.90	4
∞	69.59	1	74



Overview

- Background and overview
- Error Mining for linguistic engineering
- Disambiguation 1: parse selection with log-linear model
- Disambiguation 2: incorporating selection restrictions
- Efficiency: Learning Efficient Parsing

Self-learned Selection Restrictions

- Reasonable Accuracy (about 90% accuracy named dependencies)
- Silly mistakes . . .

Self-learned Selection Restrictions

- Reasonable Accuracy (about 90% accuracy named dependencies)
- Silly mistakes
 - (2) Melk drinkt de baby niet
Milk drinks the baby not
intended: *The baby doesn't drink milk*
parser: *Milk doesn't drink the baby*

Self-learned Selection Restrictions

- Reasonable Accuracy (about 90% accuracy named dependencies)
- Silly mistakes . . .
 - (2) Melk drinkt de baby niet
Milk drinks the baby not
intended: *The baby doesn't drink milk*
parser: *Milk doesn't drink the baby*
- other things being equal, the parser prefers fronted subjects . . .

Silly mistakes: subject vs. object

- (3) Campari moet **u** gedronken hebben
intended: *You must have drunk Campari*
Alpino: *Campari must have drunk you*

Silly mistakes: subject vs. object

- (3) Campari moet **u** gedronken hebben
intended: *You must have drunk Campari*
Alpino: *Campari must have drunk you*
- (4) De wijn die **Elvis** zou hebben gedronken als hij wijn zou hebben gedronken
intended: *The wine Elvis would have drunk if he had drunk wine*
Alpino: *The wine that would have drunk Elvis if he had drunk wine*

Silly mistakes: subject vs. object

- (3) Campari moet **u** gedronken hebben
intended: *You must have drunk Campari*
Alpino: *Campari must have drunk you*
- (4) De wijn die **Elvis** zou hebben gedronken als hij wijn zou hebben gedronken
intended: *The wine Elvis would have drunk if he had drunk wine*
Alpino: *The wine that would have drunk Elvis if he had drunk wine*
- (5) De paus heeft **tweehonderd daklozen** te eten gehad
The pope had twohundred homeless people for dinner
Alpino: *The pope is a cannibal. . .*

Disambiguation Model . . . is insufficient

- Training: 7.150 sentences (cdbI-part of Eindhoven-corpus)
- Features: 22.500 features (after frequency cut-off)
- Features are capable, in principle, to represent bi-lexical preferences
- In training data: 3 occurrences of verb *to drink*
- *Not enough training data* to learn weights for bi-lexical features

The Plan

- Use parser to construct much more training data
- About 500 million words
- Estimate bi-lexical preferences with pointwise Mutual Information
- Integrate these in disambiguation model

How could this ever work?

How could this ever work?

- example: subject-object ambiguity
- most of the time: no ambiguity
- learn from the majority of non-ambiguous cases
- to select better parse in ambiguous cases

More Training Data

- TwNC, CLEF parsed with Alpino

words		500,000,000
sentences	100%	30,000,000
sentences without parse	0.2%	100,000
<hr/>		
sentences with fragments	8%	2,500,000
sentences with single full parse	92%	27,400,000

Extract Lexical Dependencies

- triples of Head, DependentHead, Relation
- `obj1(drink,milk)`
- use all types of dependencies (su, obj1, obj2, mod, det, app, ld, whd, rhd, cmp, . . .)
- Additional dependencies

Additional Lexical Dependencies

- Additional dependencies for coordination
 - ★ Bier_i of wijn_i dronk_i Elvis niet
- Additional dependencies for relative clauses
 - ★ De wijn_i die Elvis niet dronk_i

Frequency cut-off

- Frequency cut-off: at least 20 instances for each triple
- 2 million triple types are used
- Advantages:
 - ★ smaller model
 - ★ mutual information scores more reliable for higher frequencies

Bilexical preference

- Pointwise Mutual Information (Fano 1961, Church and Hanks 1990)

$$I(r(w_1, w_2)) = \log \frac{f(r(w_1, w_2))}{f(r(w_1, -))f(-(-, w_2))}$$

- compares actual frequency with expected frequency
- Example: $I(\text{obj1}(\text{drink}, \text{melk}))$
 - ★ $N=470,000,000$
 - ★ $C(\text{obj1}(\text{drink}, \text{melk}))$: 195
 - ★ $C(\text{obj1}(\text{drink}, -))$: 15713
 - ★ $C(-(-, \text{melk}))$: 10172
 - ★ expected count: 0.34
 - ★ actual count is about 560 times as big
 - ★ its log: 6.3

Highest scoring bilexical preferences between verbs and direct objects

bijltje	gooi_neer	throw the axe
duimschroef	draai_aan	turn thumb screws
'goes by'	time	
kostje	scharrel	earn a living
peentje	zweet	to sweat roots
traantje	pink_weg	
boontje	dop	
centje	verdien_bij	earn a penny
champagne_fles	ontkurk	uncork champagne bottle
dorst	les	satisfy thirst

Highest scoring objects of *drink*

biertje, borreltje, glaasje, pilsje, pintje, pint, wijntje, alcohol, bier, borrel, cappuccino, champagne, chocolademelk, cola, espresso, koffie, kopje, limonade, liter, pils, slok, vruchtensap, whisky, wodka, cocktail, drankje, druppel, frisdrank, glas, jenever, liter, melk, sherry, slok, thee, wijn, blikje, **bloed**, drank, flesje, fles, kop, liter, **urine**, beker, **dag**, water, hoeveelheid, veel, wat

Highest scoring objects of eet, $I > 3$

boterhammetje, hapje, **Heart**, mens_vlees, patatje, **work**, biefstuk, boer_kool, boterham, broodje, couscous, drop, frietje, friet, fruit, gebakje, hamburger, haring, **home**, ijsje, insect, kaas, kaviaar, kers, koolhydraat, kroket, mossel, oester, oliebol, pannenkoek, patat, pizza, rundvlees, slak, soep, spaghetti, spruitje, stam_pot, sushi, taartje, varkensvlees, vlees, aardappel, aardbei, appel, asperge, banaan, boon, brood, chocolade, chocola, garnaal, gerecht, gras, groente, hap, kalkoen, kilo, kip, koekje, kreeft, maaltijd, paling, pasta, portie, rijst, salade, sla, taart, toetje, vet, visje, vis, voedsel, voer, worst, bordje, bord, chip, **dag**, ei, gram, ijs, kilo, knoflook, koek, konijn, paddestoel, plant, **service**, stukje, **thuis**, tomaat, vrucht, wat, wild, zalm. . .

Lexical preferences between verbs and MOD modifiers

overlangs	snijd_door	to cut in length
ten hele	dwaal	go astray fully
welig	tier	
achteruit	deins	move backward in fear
dunnetjes	doe_over	
ineen	schrompel	
omver	kegel	
on_zedelijk	betast	touch indecently
stief_moederlijk	bedeel	
stierlijk	verveel	
straal	loop_voorbij	
uiteen	rafel	

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblust	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke		

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblust	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke	nonsens	
gebalde		

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblost	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke	nonsens	
gebalde	vuist	
gefronste		

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblust	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke	nonsens	
gebalde	vuist	
gefronste	wenkbrauw	
bodemloze		

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblust	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke	nonsens	
gebalde	vuist	
gefronste	wenkbrauw	
bodemloze	put	bottomless pit
eenarmige		

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblost	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke	nonsens	
gebalde	vuist	
gefronste	wenkbrauw	
bodemloze	put	bottomless pit
eenarmige	bandiet	one-armed bandit
exhibitionistische		

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblust	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke	nonsens	
gebalde	vuist	
gefronste	wenkbrauw	
bodemloze	put	bottomless pit
eenarmige	bandiet	one-armed bandit
exhibitionistische	zelfverrijking	exhibitionistic self-enrichment
tiendaagse		

Lexical preferences between nouns and adjectives

endoplasmatisch	reticulum	
zelfrijzend	bakmeel	
waterbesparende	douchekop	
ongeblost	kalk	
onbevlekt	ontvangenis	immaculate conception
ingegroeid	teennagel	
knapperend	haardvuur	
geconsacreerde	hostie	
bezittelijk	voornaamwoord	possessive pronoun
pienterre	pookje	
afgescheurde	kruisband	
baarlijke	nonsens	
gebalde	vuist	
gefronste	wenkbrauw	
bodemloze	put	bottomless pit
eenarmige	bandiet	one-armed bandit
exhibitionistische	zelfverrijking	exhibitionistic self-enrichment
tiendaagse	veldtocht	ten-day campaign

Using association scores as disambiguation features

- new features $z(p, r)$ for each POS-tag p and dependency r

$z(\text{verb}, \text{su})$	$z(\text{noun}, \text{su})$	$z(\text{adj}, \text{su})$...
$z(\text{verb}, \text{obj1})$	$z(\text{noun}, \text{obj1})$...	
$z(\text{verb}, \text{mod})$	$z(\text{noun}, \text{mod})$		
$z(\text{verb}, \text{predm})$...		
...			

Using association scores as disambiguation features

- new features $z(p, r)$ for each POS-tag p and dependency r

$z(\text{verb}, \text{su})$	$z(\text{noun}, \text{su})$	$z(\text{adj}, \text{su})$...
$z(\text{verb}, \text{obj1})$	$z(\text{noun}, \text{obj1})$...	
$z(\text{verb}, \text{mod})$	$z(\text{noun}, \text{mod})$		
$z(\text{verb}, \text{predm})$...		
...			

- feature $z(p, r)$ is present in a parse if there is an r -dependency between word w_1 (with Pos-tag p) and word w_2

Using association scores as disambiguation features

- new features $z(p, r)$ for each POS-tag p and dependency r

$z(\text{verb}, \text{su})$	$z(\text{noun}, \text{su})$	$z(\text{adj}, \text{su})$...
$z(\text{verb}, \text{obj1})$	$z(\text{noun}, \text{obj1})$...	
$z(\text{verb}, \text{mod})$	$z(\text{noun}, \text{mod})$		
$z(\text{verb}, \text{predm})$...		
...			

- feature $z(p, r)$ is present in a parse if there is an r -dependency between word w_1 (with Pos-tag p) and word w_2
 - ★ the count of $z(p, r)$ is given by $I(r(w_1, w_2))$

Using association scores as disambiguation features

- new features $z(p, r)$ for each POS-tag p and dependency r

$z(\text{verb}, \text{su})$	$z(\text{noun}, \text{su})$	$z(\text{adj}, \text{su})$...
$z(\text{verb}, \text{obj1})$	$z(\text{noun}, \text{obj1})$...	
$z(\text{verb}, \text{mod})$	$z(\text{noun}, \text{mod})$		
$z(\text{verb}, \text{predm})$...		
...			

- feature $z(p, r)$ is present in a parse if there is an r -dependency between word w_1 (with Pos-tag p) and word w_2
 - ★ the count of $z(p, r)$ is given by $I(r(w_1, w_2))$
 - ★ only for $I > 0$

Using association scores as disambiguation features

- new features $z(p, r)$ for each POS-tag p and dependency r

$z(\text{verb}, \text{su})$	$z(\text{noun}, \text{su})$	$z(\text{adj}, \text{su})$...
$z(\text{verb}, \text{obj1})$	$z(\text{noun}, \text{obj1})$...	
$z(\text{verb}, \text{mod})$	$z(\text{noun}, \text{mod})$		
$z(\text{verb}, \text{predm})$...		
...			

- feature $z(p, r)$ is present in a parse if there is an r -dependency between word w_1 (with Pos-tag p) and word w_2
 - ★ the count of $z(p, r)$ is given by $I(r(w_1, w_2))$
 - ★ only for $I > 0$
 - ★ sum counts if there are multiple pairs of words with same relation

Using association scores as disambiguation features

- new features $z(p, r)$ for each POS-tag p and dependency r

$z(\text{verb}, \text{su})$	$z(\text{noun}, \text{su})$	$z(\text{adj}, \text{su})$...
$z(\text{verb}, \text{obj1})$	$z(\text{noun}, \text{obj1})$...	
$z(\text{verb}, \text{mod})$	$z(\text{noun}, \text{mod})$		
$z(\text{verb}, \text{predm})$...		
...			

- feature $z(p, r)$ is present in a parse if there is an r -dependency between word w_1 (with Pos-tag p) and word w_2
 - ★ the count of $z(p, r)$ is given by $I(r(w_1, w_2))$
 - ★ only for $I > 0$
 - ★ sum counts if there are multiple pairs of words with same relation
- In total, < 150 new features; therefore, treebank large enough to estimate their weights
- background described in Johnson and Riezler (NAACL 2000 Seattle)

Example

- Melk drinkt de baby niet
Milk, the baby does not drink
- correct analysis:
 - ★ $z(\text{verb,obj1})=6$
 - ★ $z(\text{verb,su})=3$
- alternative analysis:
 - ★ $z(\text{verb,obj1})=0$
 - ★ $z(\text{verb,su})=0$
- ★ weight $z(\text{verb,obj1})$: 0.0101179
- ★ weight $z(\text{verb,su})$: 0.00877976

Evaluation: Experiment 1

- ten-fold cross validation Alpino Treebank

	fscore	err.red.	exact	CA
	%	%	%	%
standard	87.41	74.60	52.0	87.02
+self-training	87.91	77.38	54.8	87.51

Evaluation: Experiment 2

- Full system D-Coi Treebank (Trouw newspaper part)

	prec	rec	fscore	CA
	%	%	%	%
standard	90.77	90.49	90.63	90.32
+self-training	91.19	90.89	91.01	90.73

Overview

- Background and overview
- Error Mining for linguistic engineering
- Disambiguation 1: parse selection with log-linear model
- Disambiguation 2: incorporating selection restrictions
- Efficiency: Learning Efficient Parsing

My favorite application of parser

My favorite application of parser

- Parsing! . . .

My favorite application of parser

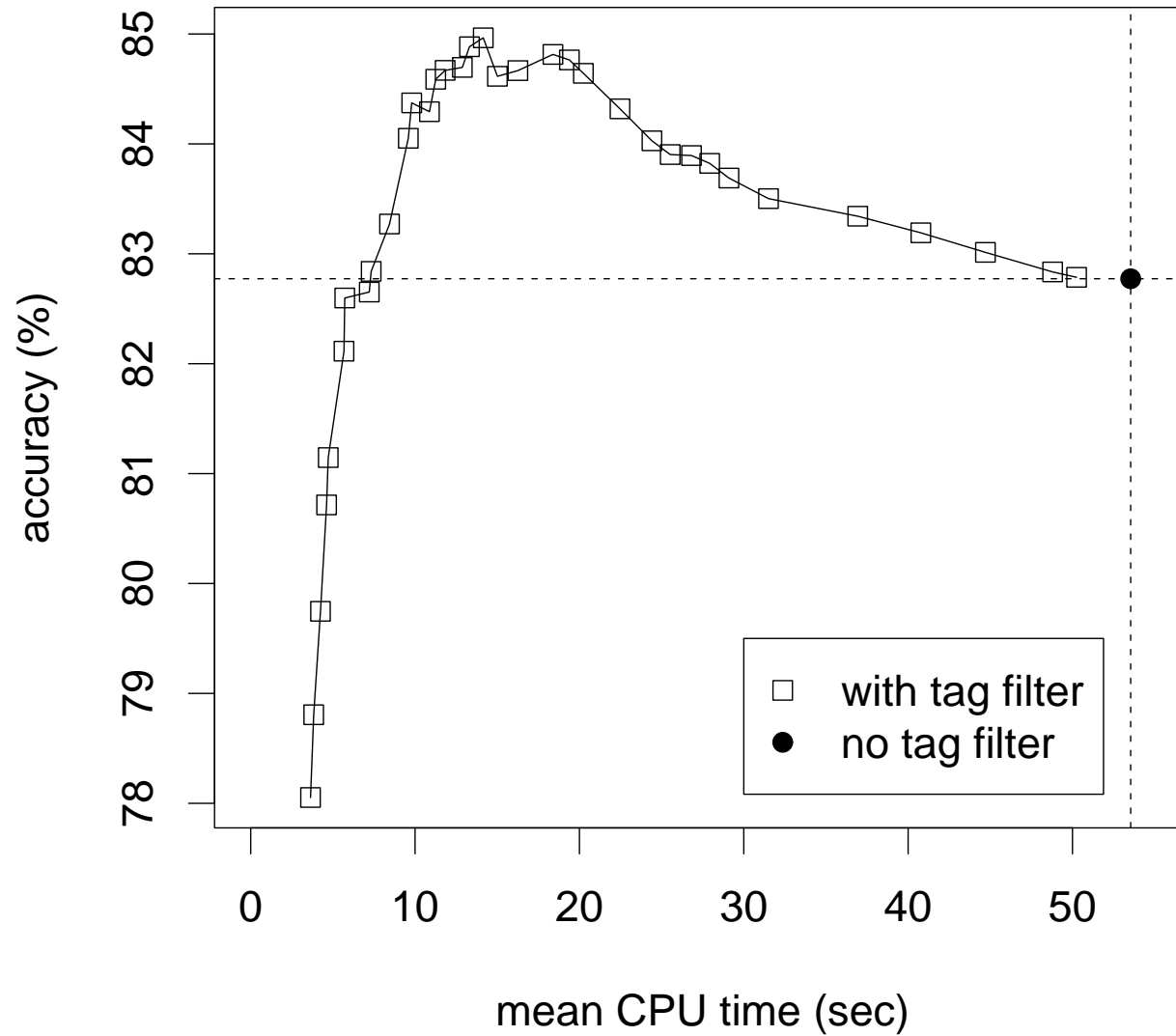
- Parsing! . . .
- Annotate data automatically
- Extract information
- Parser uses that information



Example: POS-tag filter

- Large corpus parsed by Alpino
- Keep track of lexical categories used in best parse
- Train tagger
- Tagger removes unlikely lexical categories
- Parser is faster and more accurate
 - ★ results confirmed now in OpenCCG

POS-tag filter: result

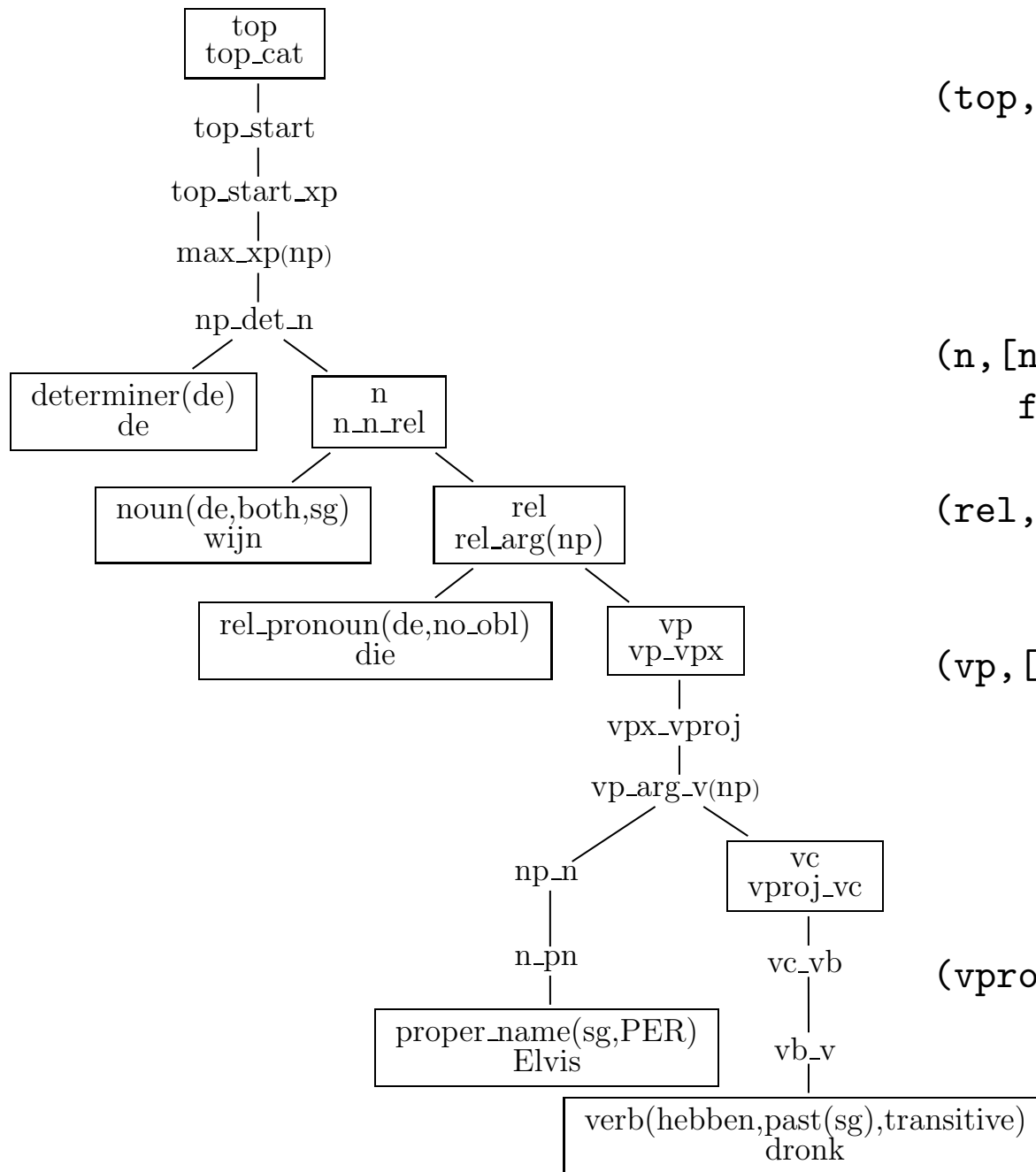


Example: Learning Efficient Parsing

- Large corpus parsed by Alpino
- Keep track of parse step sequences used for best parse
- During parsing: only allow parse step sequences observed earlier
- Parser is much faster, with almost equal accuracy

Learning Efficient Parsing: details

- left-corner parser (Matsumoto et al. 1983; Pereira & Shieber 1987; van Noord 1997)
- left-corner spline: sequences of rule applications in the context of a given goal
- example:
 - (6) De wijn die Elvis dronk
The wine which Elvis drank



(top, [determiner(de), np_det_n, max_xp(np), top_start_xp, top_start, top_cat, finish]).

(n, [noun(de,both,sg), n_n_rel, finish]).

(rel, [rel_pronoun(de,no_obl), rel_arg(np), finish]).

(vp, [proper_name(sg,PER), n_pn, np_n, vp_arg_v(np), vpx_vproj, vp_vpx, finish]).

(vproj, [verb(hebben,past(sg),transitive), vb_v, vc_vb, vproj_vc, finish]).

Filtering left-corner splines

- Check if the step

$$(g, r_1 \dots r_{i-1}) \longrightarrow (g, r_1 \dots r_i)$$

is acceptable

- Context size

- ★ bigram: g, r_{i-1}, r_i

- ★ trigram: g, r_{i-2}, r_{i-1}, r_i

- ★ fourgram: $g, r_{i-3}, r_{i-2}, r_{i-1}, r_i$

- ★ prefix: $g, r_1 \dots r_i$

- Required evidence

- ★ relative frequency?

- ★ absolute frequency $> \tau$

- Best option: prefix filter with $\tau = 0$.

Example

current spline:

```
(vp, [proper_name(sg,PER),  
      n_pn,  
      np_n,  
      vp_arg_v(np)])
```

proposed rule: vpx_vproj

check training data for:

```
(vp, [proper_name(sg,PER),  
      n_pn,  
      np_n,  
      vp_arg_v(np),  
      vpx_vproj|_])
```

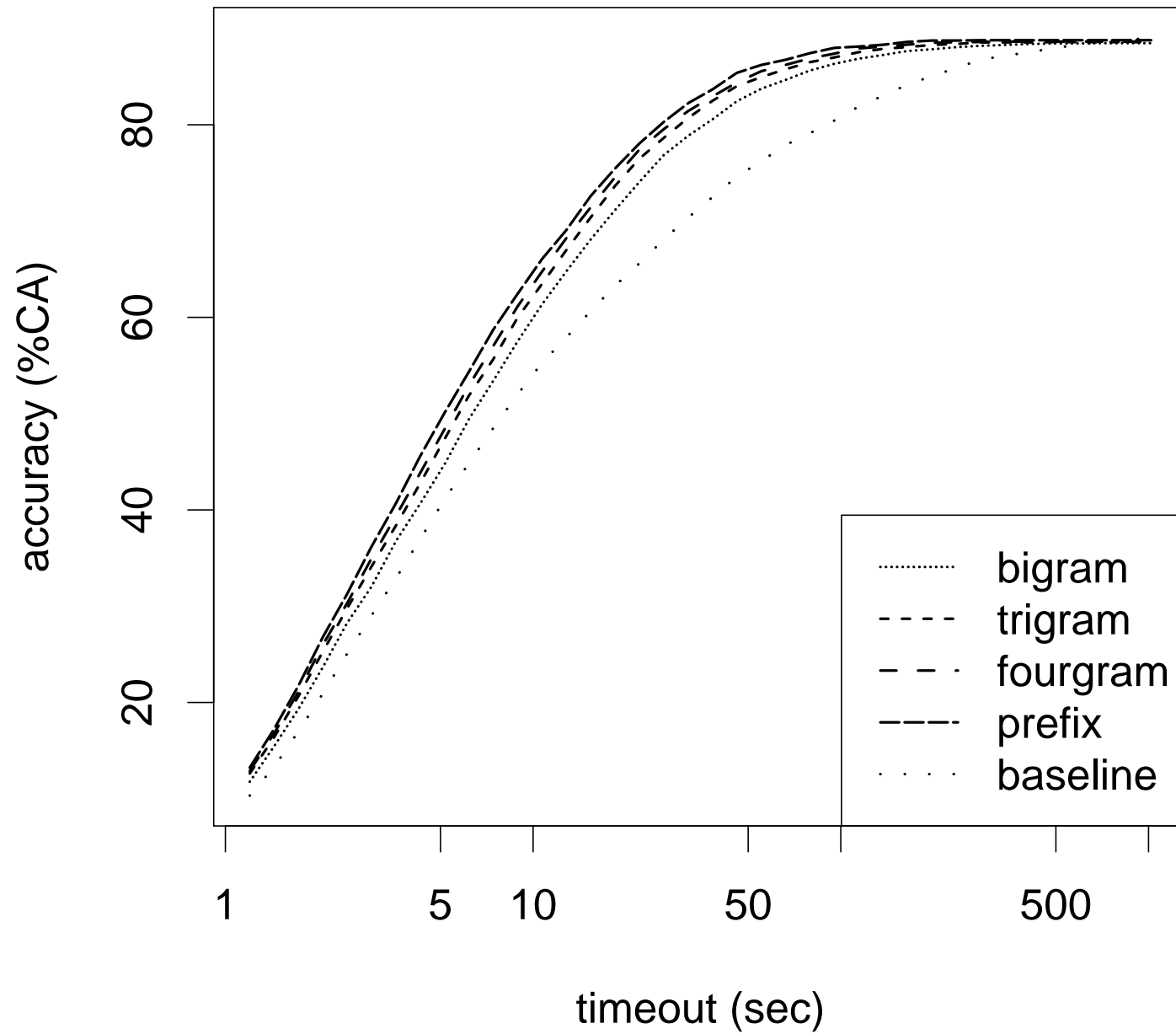
Implementation

- store table of observed partial splines
- hash-table (very large)
- only store hash-keys!

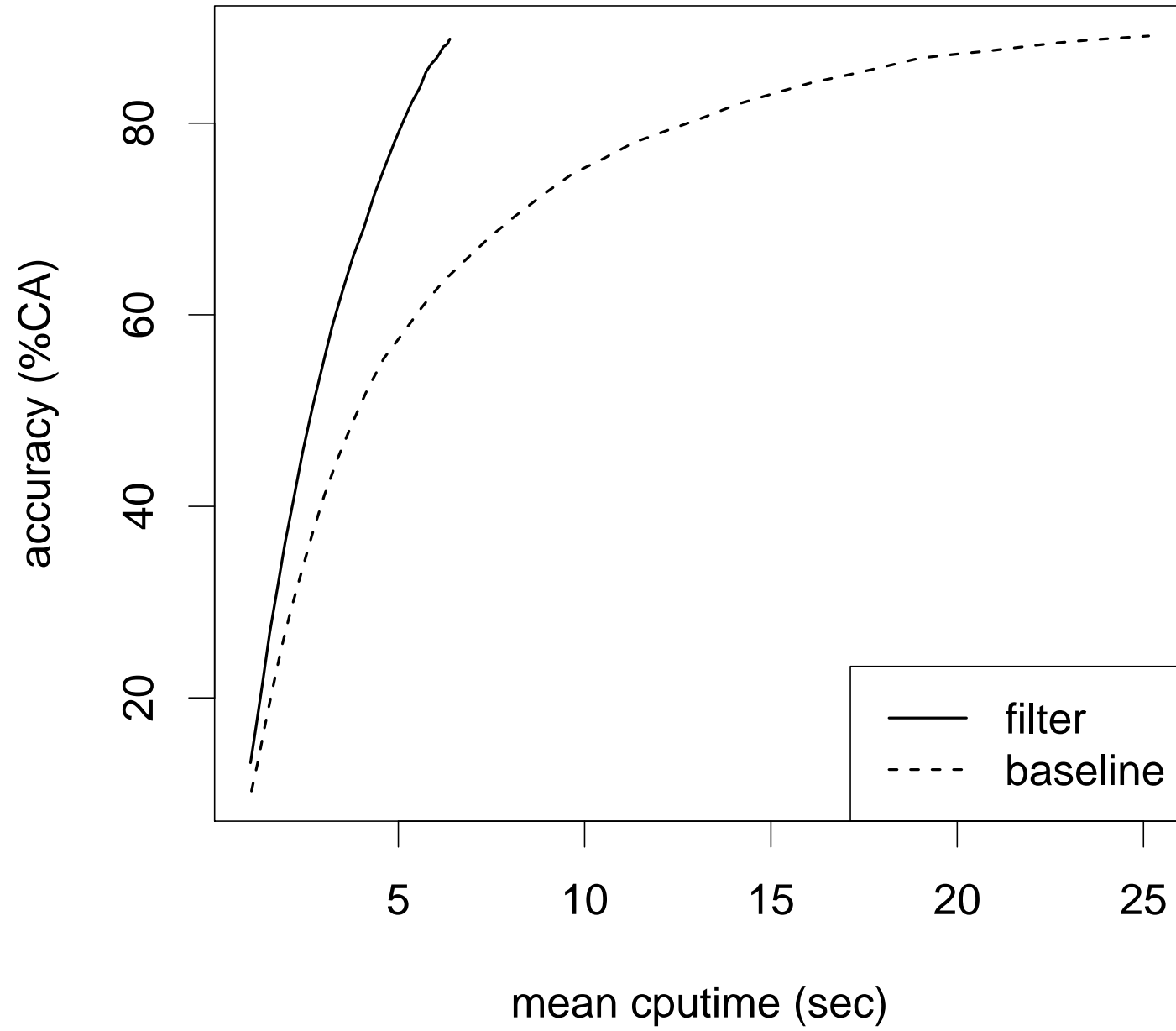
Experiments

- mean CPU-time?
 - ★ CPU-times vary wildly for different inputs
 - ★ CPU-time is not linear in sentence length
 - ★ Irrelevant for on-line application
- Alternative: assume time-out per sentence
 - ★ If a sentence times out, your accuracy is 0.00
 - ★ Compute accuracy for a given time-out
 - ★ On-line scenario: compare accuracies for various time-outs
 - ★ Off-line scenario: compare accuracies for mean CPU-time (with time-outs)

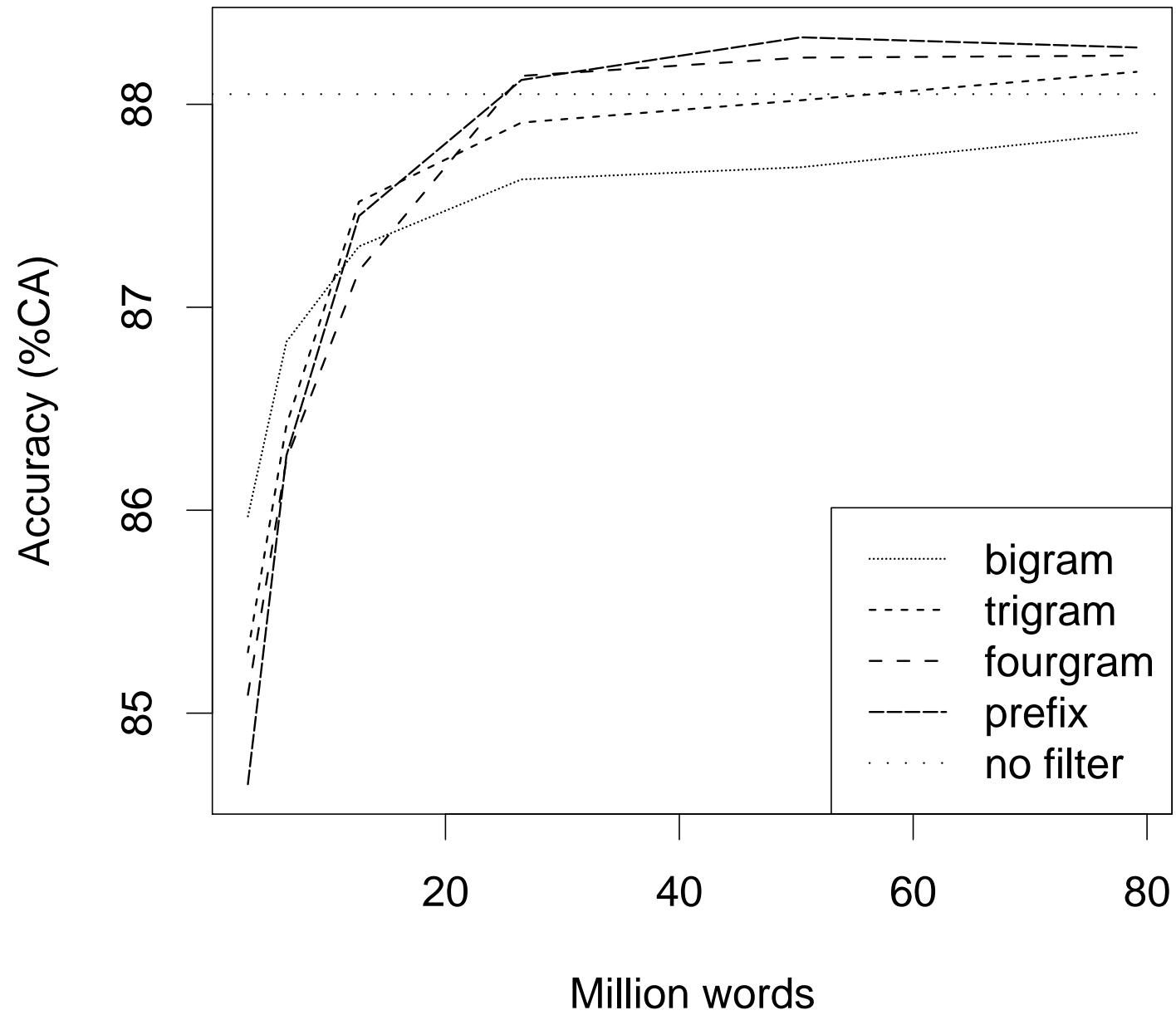
Results on-line scenario



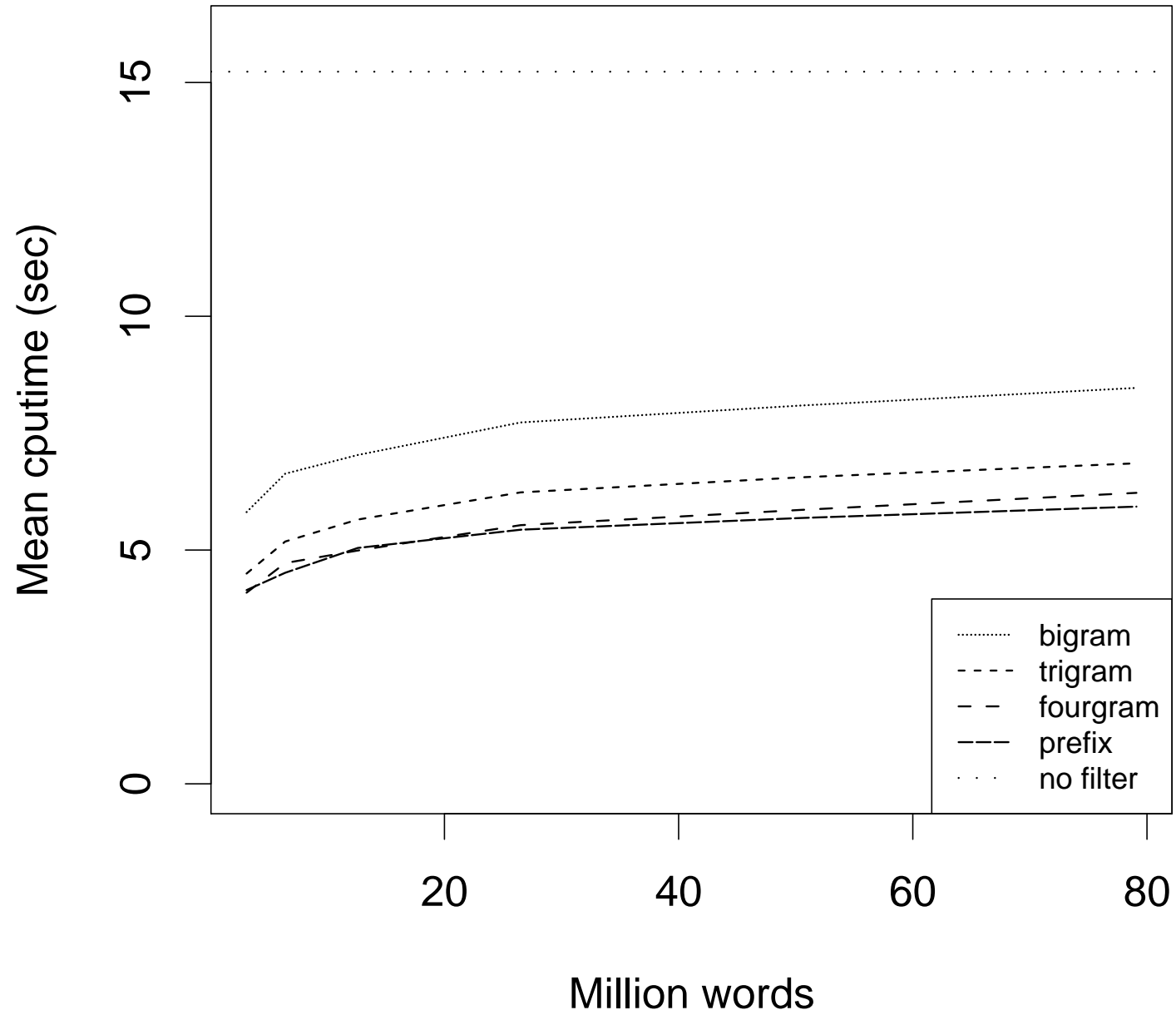
Results off-line scenario



Amount of data and accuracy



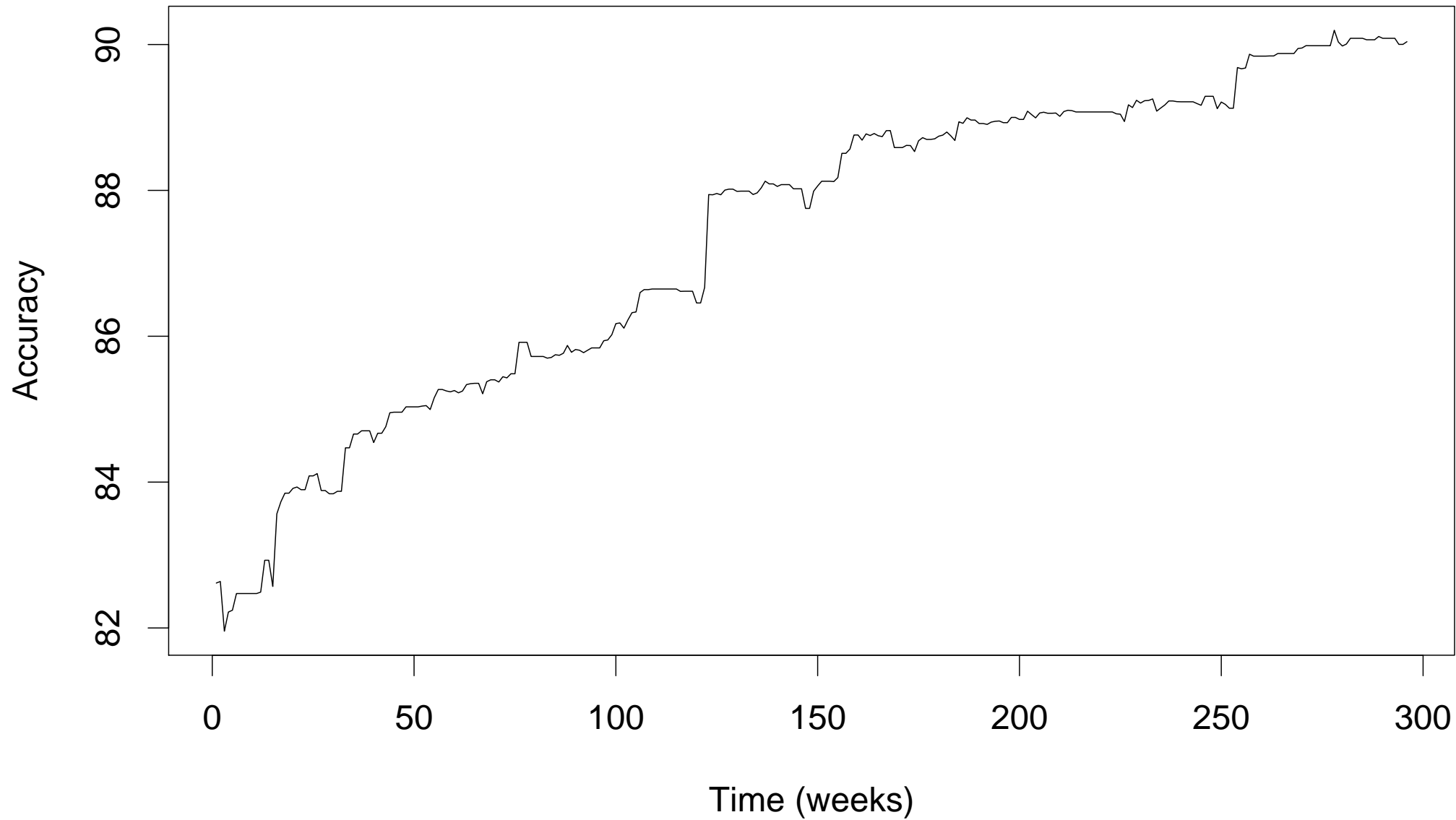
Amount of data and CPU-time



Conclusion

- Illustrated some aspects of one specific parser for one specific language
- General theme: treebanks and corpora are enormously important
 - ★ Treebanks for training disambiguation component
 - ★ Huge corpora for error mining
 - ★ Self-learning techniques on huge corpora improve:
 - * lexical analysis (tagger)
 - * disambiguation (selection restrictions)
 - * efficiency (restrict parser to focus on promising computations)

Development



It's free!

`http://www.let.rug.nl/~vannoord/alp/Alpino/`

`http://www.let.rug.nl/~vannoord/trees/`

`http://www.let.rug.nl/~dekok/`

Presentation based on following publications

- Error mining:
 - ★ Gertjan van Noord. Error Mining for Wide-Coverage Grammar Engineering. In: ACL 2004, Barcelona
 - ★ Benoît Sagot and Éric de la Clergerie. Error Mining in Parsing Results. In: ACL/COLING 2006, Sydney
 - ★ Daniel de Kok, Gertjan van Noord. A generalized method for iterative error mining in parsing results. Talk presented at CLIN 19, January 22 2009, Groningen

- Disambiguation 1
 - ★ Robert Malouf, Gertjan van Noord. Wide Coverage Parsing with Stochastic Attribute Value Grammars. In: IJCNLP-04 Workshop Beyond Shallow Analyses - Formalisms and statistical modeling for deep analyses.
 - ★ Gertjan van Noord, Robert Malouf. Wide Coverage Parsing with Stochastic Attribute Value Grammars. Unpublished manuscript.

Presentation based on following publications (2)

- Disambiguation 2
 - ★ Gertjan van Noord. Self-trained Bilexical Preferences to Improve Disambiguation Accuracy. To appear in a book on parsing technology, based on selected papers from the IWPT 2007, CONNL 2007, and IWPT 2005 workshops, edited by Harry Bunt, Paola Merlo and Jakim Nivre, published by Springer.
 - ★ Gertjan van Noord. Using Self-Trained Bilexical Preferences to Improve Disambiguation Accuracy. In: Proceedings of the Tenth International Conference on Parsing Technologies. IWPT 2007, Prague. Pages 1–10.

- Efficiency
 - ★ Gertjan van Noord, Learning Efficient Parsing. To appear in EACL 2009, Athens.