Automatische Extraktion von Prädikat-Argument-Strukturen

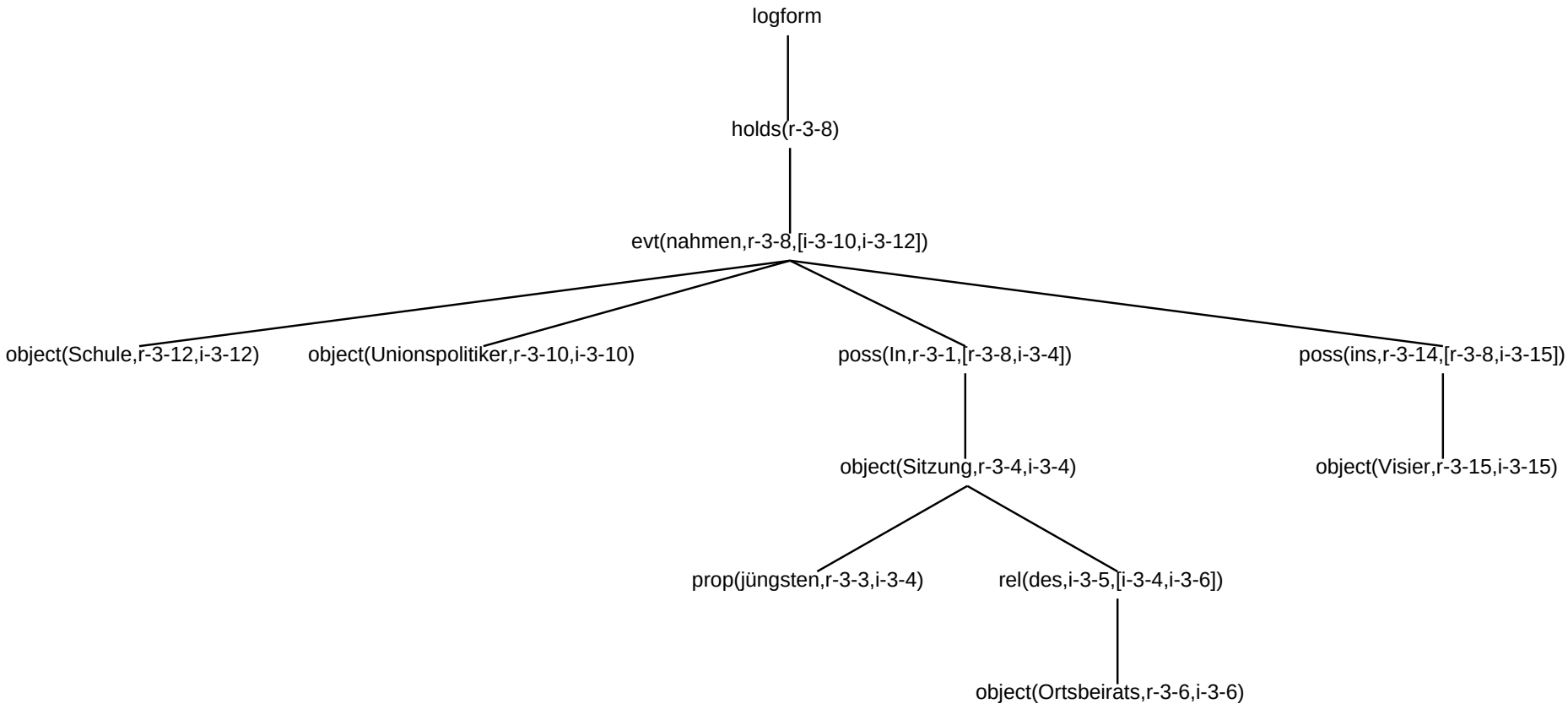
Manfred Klenner

Übersicht

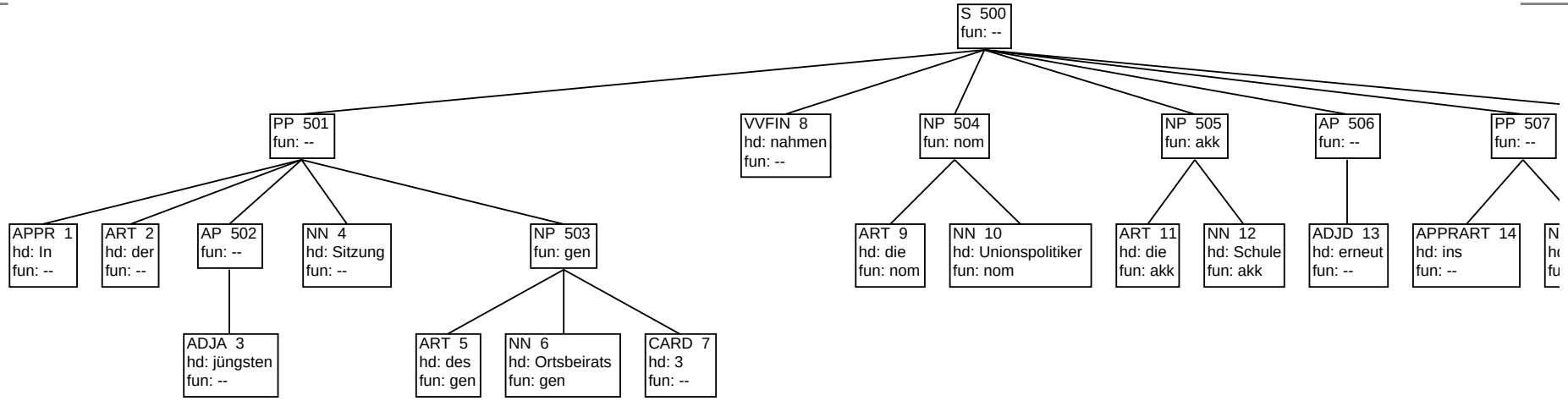
- Zielsetzung und Lernaufgabe
- Ressourcen und Tools
- Semantische Interpretation
- Extraktion von Prädikat-Argument-Strukturen
- Regellernen samt Empirie
- Fazit und Rückblick
- Appendix: Beschreibung der Parser und Grammatiken
- verschoben: Timbl und ILP

Zielsetzung: MLF

[In,der,jüngsten,Sitzung,des,Ortsbeirats,3,nahmen,die,Unionspolitiker,die,Schule,erneut,ins,Visier]



Zielsetzung: Verbprädikat



In:1 der:2 jüngsten:3 Sitzung:4 des:5 Ortsbeirats:6 3:7 nahmen:8 die:9 Unionspolitiker:10 die:11 Schule:12 erneut:13 ins:14 Visier:15 .:16

$\Rightarrow$ sem(3,[8,10,12])

$\Rightarrow$ als Prädikat: [ins_visier_]nehmen(Unionspolitiker,Schule)

$\Rightarrow$ als MLF: evt(nehmen,e-3-8,[o-3-10,o-3-12])

Lernaufgabe

gegeben: Syntaxbäume samt zugeordneten Verbrahmen

lerne: Regeln, die Syntaxbäume auf Verbrahmen abbilden

statt: Regeln von Hand schreiben also Regeln lernen lassen

Das Generieren einer Prädikat-Argument-Struktur umfasst:

- das Identifizieren der Argumente (Köpfe)
- das Zuweisen der Köpfe zu Argumentpositionen

Vorteil eines Lernansatzes

- Annotierte Daten sind in jedem Fall erforderlich
  - beim Engineeringansatz zur Evaluation
  - beim Lernansatz zur Regelinduktion

- Änderungen des Eingabeformats (der Bäume, nicht der Prädikatstrukturen)
  - Regeln müssen spezifisch auf die Parserausgabe zugeschnitten sein
    - verbesserte Versionen eines statistischen Parsers können sich in anderen Baumstrukturen manifestieren
    - Wechsel des Parsers ist immer ein Wechsel des Baumformats
  - manuell: neues Regelengineering (Anpassen oder Neuschreiben)
  - Lernansatz: neu Trainieren mit neuen Bäumen

- Rauschen (fehlerhafte Analysen)
  - die Anpassung an verrauschte Daten fällt ML-Verfahren leicht

Ressourcen

- Ausgangspunkt: Negrakorpus (20000 Sätze)
- 1001 Verbinstanzen (etwa 800 Sätze) und ihre Prädikat-Argument-Form (manuell annotiert)
  - Ellipsen und Passiv zwar annotiert, aber nicht verwendet
  - Komplement-PPs zwar als solche annotiert, aber nicht verwendet
  - 16 3-wertige, 531 2-wertige, 454 1-wertige Verben
  - Annotationsbeispiel

```
3: In:1 der:2 jüngsten:3 Sitzung:4 des:5 Ortsbeirats:6 3:7 nahmen:8
die:9 Unionspolitiker:10 die:11 Schule:12 erneut:13 ins:14 Visier:15
****3=8:10-12-15!-4?
```

- 7000 Verbinstanzen und ihre Prädikat-Argument-Form (automatisch generiert)
  - einfache Sätze automatisch ermittelt: keine Koordination, keine S-Komplemente etc.
  - anhand einfacher Regeln (5 Stück) extrahiert
  - 187 3-wertige, 3166 2-wertige, 3698 1-wertige Verben

Tools

	Grammatik	Korpus	Baumtyp	Labeltypen	Köpfe	Performanz
Gojol	PCFG	Negra	PSG Dependenz	Negra, +	ja	schnell, robust
LoPar	(P)CFG	Zeitung (EM)	X-Bar	speziell	nein	ok, 70% von Negra
BitPar	PCFG	Negra	PSG	Negra, +	nein	ok, robust

	Gojol	BitPar	DecTree
TIMBL	+		
ILP		+	
Regellerner			
- einfach	+	+	+
- gewichtet		+	+
- generalisiert		+	+

Semantische Interpretation: eine triviale Aufgabe?

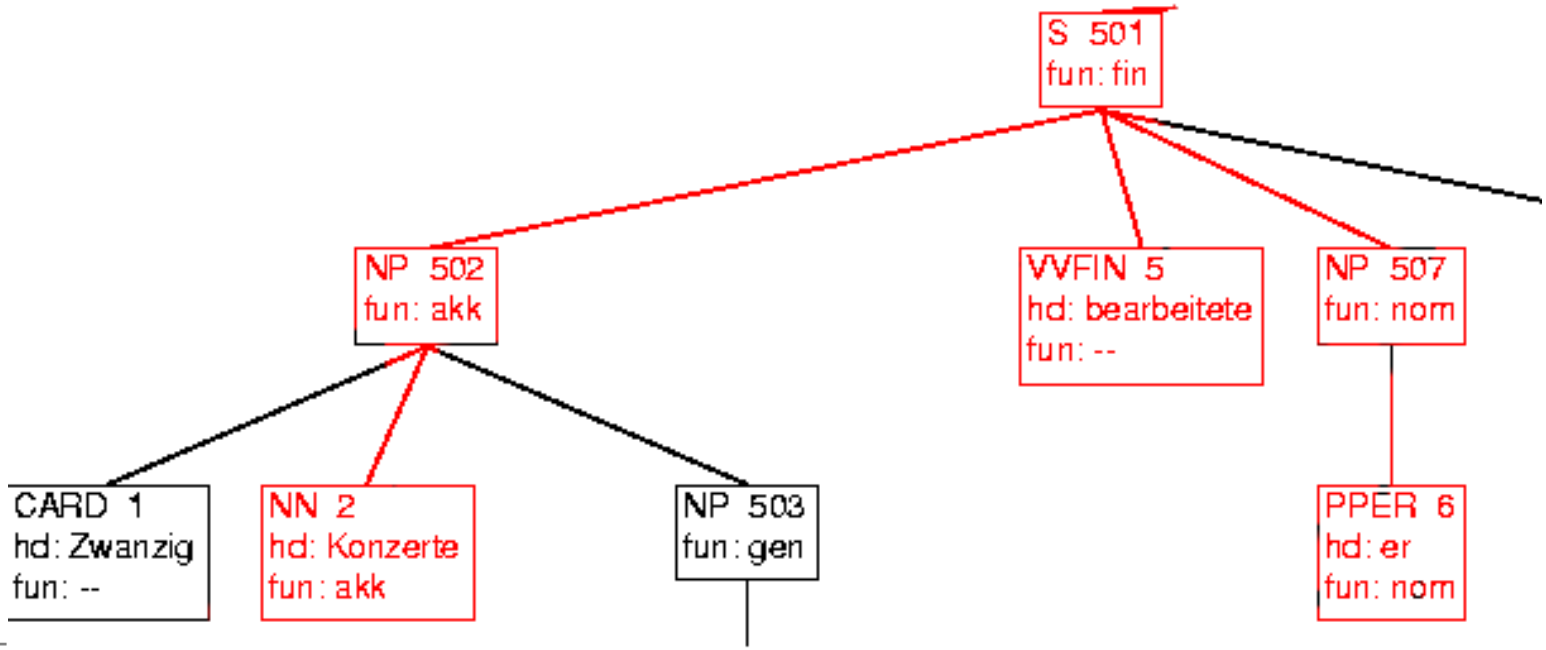
- Gegeben ein vernünftiger Parser (z.B. BitPar, der auch Kasus zuweist)
- Ist dann die Extraktion von Prädikat-Argument-Strukturen nicht trivial?
- Heuristik:
  - Suche VVFIN und alle Nomen im Nominativ, Akkussativ und Dativ, die unter dem selben Satzknoten hängen wie das Verb.
  - Bringe sie in eine kanonische Ordnung.
  - Gib aus: sem(Satz,[Verb,Nominativ,Akkussativ,Dativ]).
- Das Resultat ist enttäuschend: Präzision und Ausbeute bleiben unter 50 %

Semantische Interpretation: Hürden

- Kopfbestimmung: mehrere Nomen hintereinander
- koordinierte Strukturen: nicht die einzelnen NP-Köpfe sind Argumente, sondern ein Mengenobjekt
- viele Verben: wer mit wem? (welche Köpfe zu welchen Verben)
- Strukturkarussell: Konstituenten sind sonstwo (freie Wortstellung, Topikalisierung, Ellipsen)
  - wie weit (auch strukturell gesehen) darf ein Nomen im Akkusativ von seinem Verb entfernt sein?
- Komplementtyp: nicht nur NP-Komplemente, auch PP- und S-Komplemente (keine Kasusmarkierung)
- das grösste Problem: “das Probabilistische Parsing”
  - wie repräsentativ ist die Baumbankgrammatik?
  - wie verlässlich sind die erzeugten Phrasenstrukturen?
  - Fehler sind erwartbar, die Frage ist, wie systematisch sie sind

Regeln lernen: sem(22,[5,6,2])

- Basisalgorithmus:
  - suche einen Ankerknoten, d.i. ein Vorgänger-Knoten des Verbs (oft die Mutter), von dem ein Pfad zu den Komplementen geht
  - speichere die Pfade gemäss der linearen Abfolge
  - speichere ebenfalls bestimmte Merkmale wie Kategorie und Funktion der Kategorie
  - konserviere die Kopf-Argument-Abbildung (welche Köpfe realisieren welche Argumentstelle)



Regeln lernen: kanonische Form

- die kanonische Form der semantischen Repräsentation:

Subjekt	1. Argument
DirObjekt	2. Argument
IndirObjekt	3. Argument (oder 2. falls kein DirObjekt)
Satzkomplement	je nachdem 2. oder 3. Argument

- die Wortformen sind an der Satzoberfläche linear geordnet, daher Mapping

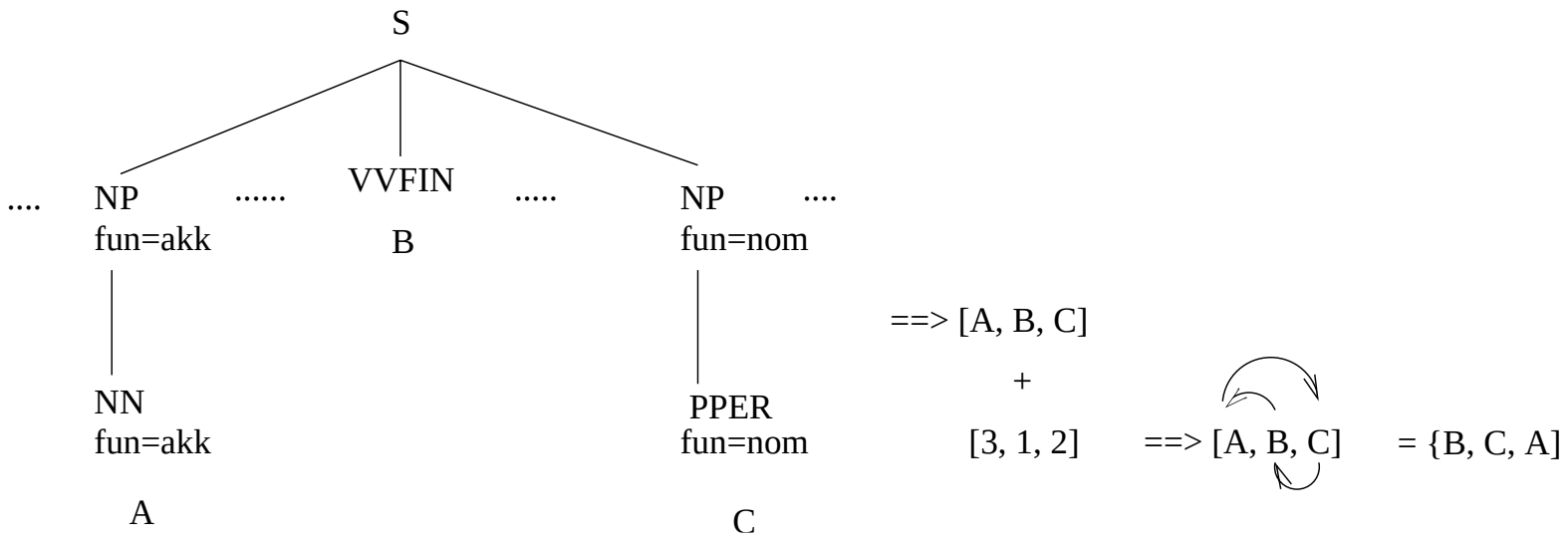
Zwanzig:1 Konzerte:2 Antonio:3 Vivaldis:4 bearbeitete:5 er:6 für:7 die:8 Orgel:9 .:10

- linear: [2,5,6] kanonisch: [5,6,2] Mapping: [3,1,2]
- Regel
  - Anker: 'S'
  - zu instantiierende Variablen: [A,B,C]
  - Mapping: [3,1,2]
  - Regelpfad (lineare Abfolge im Baum):

    [[_='NP'-akk,A='NN'-akk], [B='VVFIN'- –], [_='NP'-nom,C='PPER'-nom]]

Regeln als Baumgerippe

[[_=’NP’-akk,A=’NN’-akk], [B=’VVFIN’], [_=’NP’-nom,C=’PPER’-nom]]



Kopfheuristik

- Regeln sollen nur Köpfe abgreifen (z.B. bei mehreren Nomen unter NP)
- Kopfbestimmung nicht Teil der Regelinstantiierung
- Vorabbestimmung: welcher Knoten ist der Kopf der NP
- Häufigkeitsinformation als Kriterium:
  - gegeben n potentielle Köpfe (unter einer NP)
  - Sieger ist, wer am häufigsten in diesem Kontext Kopf war (im Trainingset).
- Trefferquote dieser Heuristik ist > 98%, z.B. 99.12 %
  - d.h. bei 1000 sind 8-9 Fehlentscheidungen
  - 1001-er Set hat 2546 Köpfe (inkl. Verben), d.h. etwa 20 Fehlentscheidungen
- Kopffehler verschlechtern die Präzision
- aufgrund von Kopffehlern rutscht die Ausbeute evtl. relativ zum Trainingset unter 100%
  - gelernt wird von den annotierten semantischen Formen ohne Rückgriff auf die Kopfheuristik
  - die so gelernten Regeln werden unter Anwendung der Kopfheuristik evaluiert
  - werden nicht alle Köpfe korrekt identifiziert, dann können auch nicht alle Trainingsbeispiele wiedergefunden werden (Recall < 100)

Etwas Empirie

- wie viele Regeln werden erzeugt?
  - 1001: 267 Regeln (223 im TrainSet (80%), 44 im TestSet (nachträglich errechnet))
  - 7000: 381 Regeln
- wie präzise und abdeckend sind sie? (1 = 1-wertiges Verb ..)

	P-Train	R	P-Test	R
2	84.2	98.4	79.1	70.9
1	99.8	99.7	99.3	86.6

1001

	prec	rec	f-meas
train	92.0	99.1	95.4
test	89.2	78.8	83.7

1001

	P-Train	R	P-Test	R
3	93.4	100	84.0	65.6
2	86.5	99.7	82.0	69.4
1	89.3	100	87.7	85.7

7000

	prec	rec	f-meas
train	87.7	99.9	94.5
test	84.9	77.6	81.1

7000

- Recall < 100 ist möglich, da heuristische Kopfregeln

Wieso überhaupt Fehler?

- zwei identische Regeln mit unterschiedlichem Mapping

```
?- rule(_,_,M1,R),rule(_,_,M2,R), M1\==M2.

M1 = [3,1,2],
M2 = [2,1,3],
R = [_A,_B,_C]-'S'-
    [[_D='NP'-akk,_A='NN'-akk],
     [_B='VVFIN'- --],
     [_E='NP'-nom,_C='NN'-nom]]
```

- Das:1 Volk:2 indessen:3 läßt:4 die:5 Einführung:6 der:7 Scharia:8 ziemlich:9 kalt:10

```
feature(178,2,fun=I).
I = nom ?
```

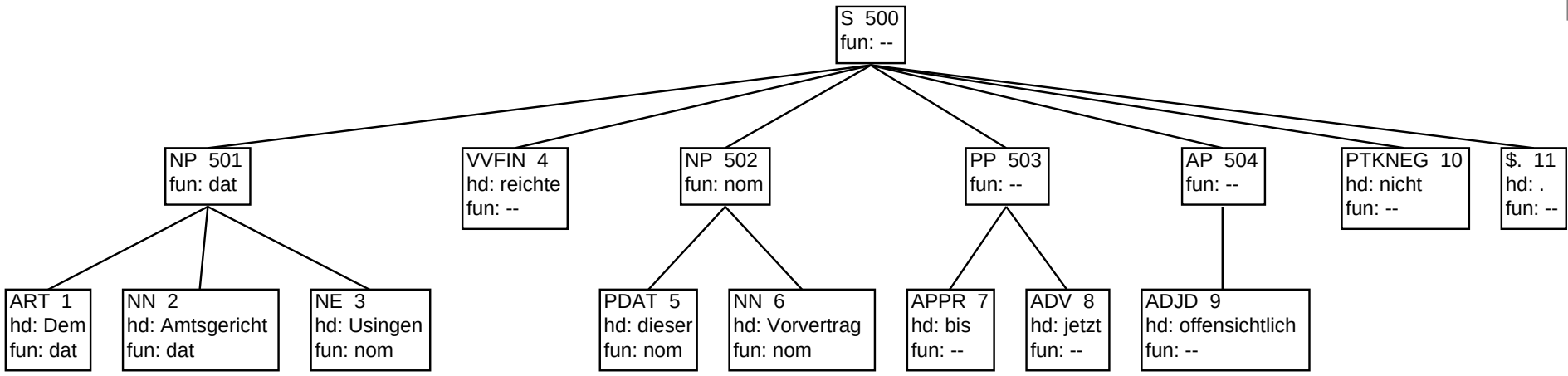
```
feature(178,6,fun=I).
I = akk ?
```

- Wer ist Schuld? – Der Autor, wer sonst!

- Er hat die seltenere (d.h. unwahrscheinlichere) Wortstellung bei morpho-syntaktisch ambigen Wortformen gewählt.

- Solche Regeln drücken die Präzision, da zu jeder korrekten eine falsche Analyse tritt.

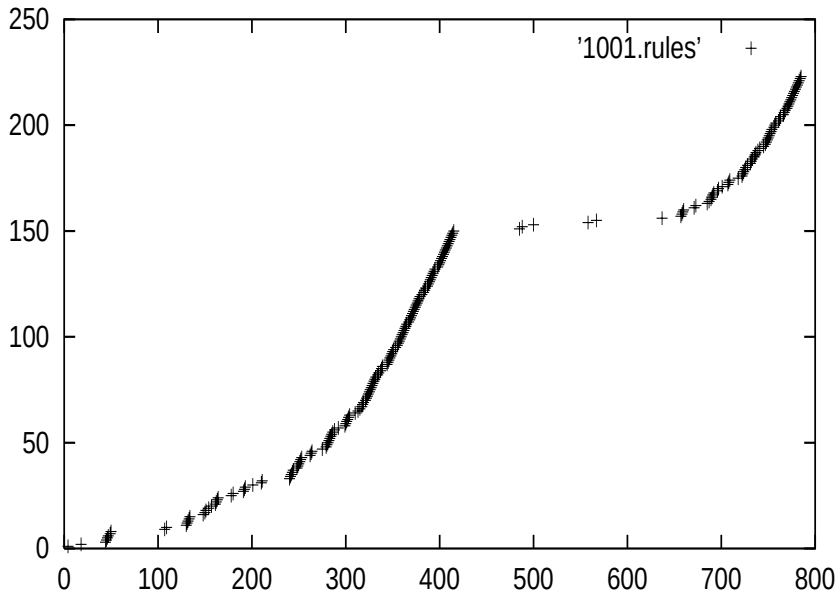
Zwei andere Fehler

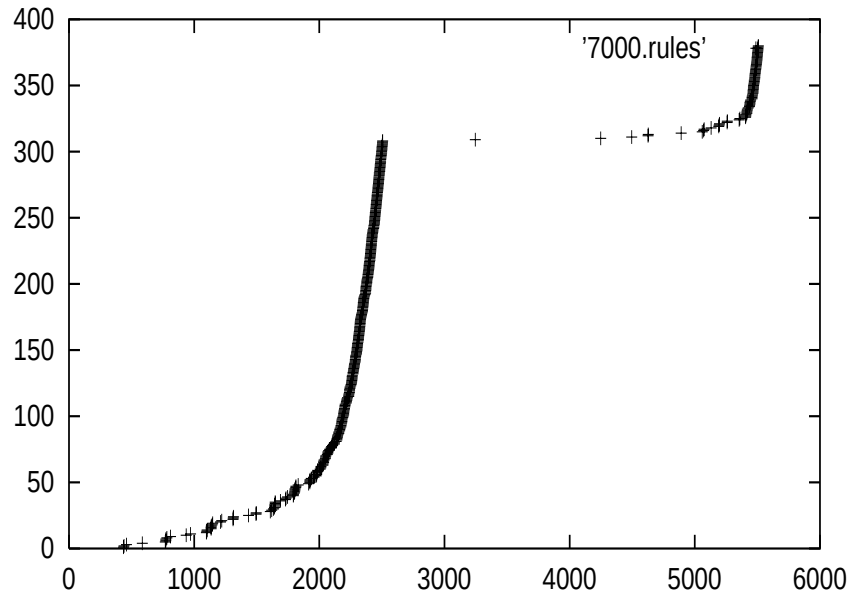


- “Dem Amtsgericht” wird als Dativ
- “Usingen” als Nominativ
- damit haben wir zwei Subjekte
- und ergo: zwei Regeln (intransive Verben) feuern

- Tagging-Fehler “.. dass das Englische (ADJA statt NN) Bankern missfällt”
- Regelpfad mit Terminal ADJA wird generiert, da im Trainingset “Englische” 1. Argument von missfällt ist (Gefahr vieler falschen Analysen)

Regeln: Konvergenz im Unendlichen?





223 Regeln im TrainSet (267)

- 147 Regeln mit 1 Verbinstanz
- 36 mit 2, ...
- 1 mit 61, ... 1 mit 70

381 Regeln im TrainSet

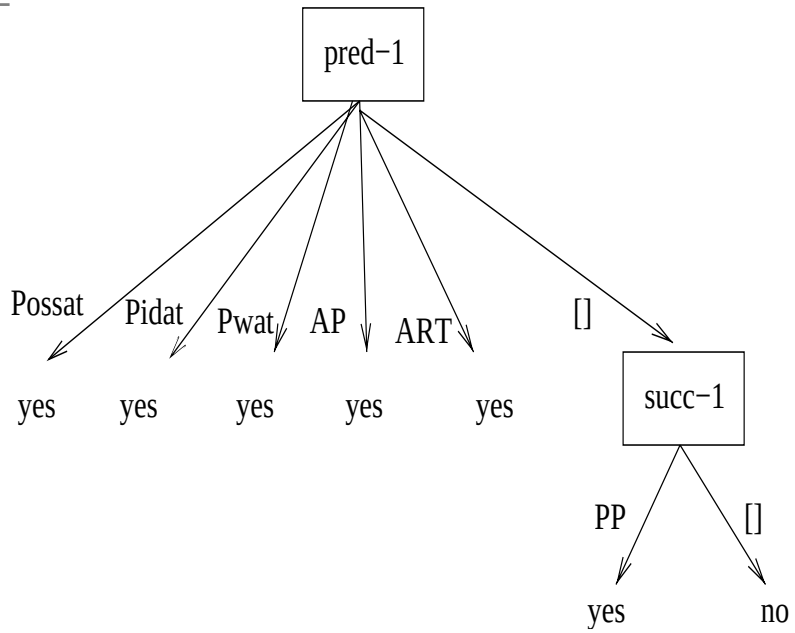
- 183 Regeln mit 1 Verbinstanz
- 40 mit 2, ...
- 1 mit 100, ..., 1 mit 220

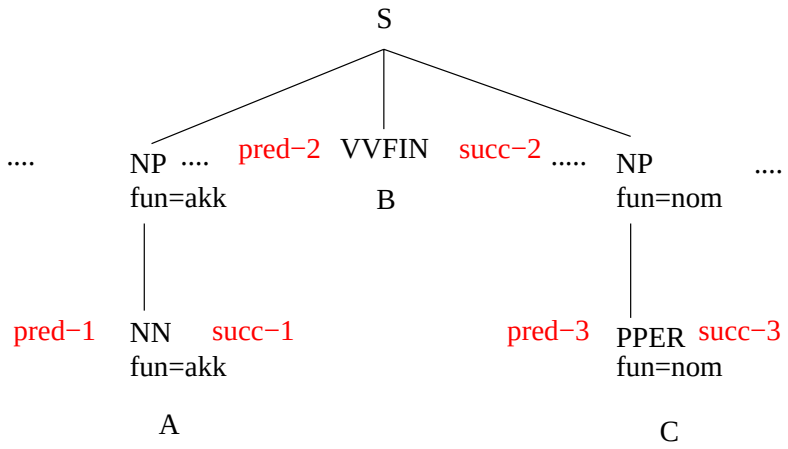
(X-Achse: Anzahl der bearbeiteten Verben, Y-Achse: Anzahl der generierten Regeln)

Probleme und Lösungsansätze

- Präzision schlecht:
  - Regeln widersprechen einander
  - mehrere Regeln (mit unterschiedl. Mappings) sind auf ein- und denselben Baum anwendbar
  - Regeln sind gänzlich falsch (Adjektiv-Fall)
  - Lösung: Diskriminationslernen bzw. Regelgewichtung, evtl. Regellöschung
- zu viele Regeln: Regelgeneralisierung
- Recall (für TestSet) schlecht: Regelgeneralisierung

Regeldiskrimination durch Entscheidungsbäume





Regelformat:

```
if pred-1='AP'-->yes
if pred-1='ART'-->yes
if pred-1='PIDAT'-->yes
if pred-1='PPOSAT'-->yes
if pred-1='PWAT'-->yes
if pred-1=[] & succ1='PP'-->yes
if pred-1=[] & succ1=[]-->no
```

Konzeption

- Ziel: Anwendbarkeit einer Regel restringieren
- Mittel: pro Regel einen Entscheidungsbaum lernen
- Bedeutung: gegeben eine Regel sei grundsätzlich anwendbar, der Entscheidungsbaum sagt dann “ja” oder “nein”
- Training: positive und negative Beispielen als Eingabe an den Lerner

- Beispiele werden als Mengen von Attribut-Wert-Paare beschrieben
- Attribute könnten sein: Geschwisterknoten auf terminaler Ebene (auch maximale Projektion), alle Labels der Knoten (i.e. fun, cat, hd), strukturelle Restriktionen, etc.
- Konzeption: Auswahl beliebig vieler Attribute, das Verfahren entscheidet, welche relevant sind
- Werte sind durch die Werte der Labels gegeben, auch binäre Werte denkbar
- positives Beispiel (fiktiv):
  yes-[prec-1=’ART’,fun_prec-1=’nom’,succ-1=[],fun_succ-1=[],...]
- Prologimplementierung verwendet, prominent C4.5

Komplexerer Entscheidungsbaum

```
Mapping: [2,3,1]
Regel:   [[_='NP'-nom,A='NN'-nom],
          [D='VP'-pp,_='NP'-akk,B='NN'-akk],
          [D='VP'-pp,C='VVPP'- --]]

if maxpro_pred1='KOUS'-yes
if maxpro_pred1='PP'-no
if maxpro_pred1=[] & maxpro_pred2='AVP'-yes
if maxpro_pred1=[] & maxpro_pred2='PP'-yes
if maxpro_pred1=[] & maxpro_pred2=[] & maxpro_pred3='AP'-yes
if maxpro_pred1=[] & maxpro_pred2=[] & maxpro_pred3='NP'-yes
if maxpro_pred1=[] & maxpro_pred2=[] & maxpro_pred3='PP' & pred1='AP'-yes
if maxpro_pred1=[] & maxpro_pred2=[] & maxpro_pred3='PP' & pred1='ART'-no
```

Etwas Empirie dazu

→ von den 267 Regeln erhalten nur 29 Entscheidungsbäume

→ die anderen sind unkritisch, d.h. sie akzeptieren keine negativen Beispiele

	P-Train	R	P-Test	R
2	99.2	98.1	97.3	65.2
1	1.0E+02	99.3	99.5	76.1

1001+dt

	pred	rec	f-meas
train	99.6	98.7	99.1
test	98.4	70.7	82.3

1001+dt

	P-Train	R	P-Test	R
2	84.2	98.4	79.1	70.9
1	99.8	99.7	99.3	86.6

1001n

	pred	rec	f-meas
train	92.0	99.1	95.4
test	89.2	78.8	83.7

1001n

Präzision sehr gut, aber Recall schlechter

Regelkondensierung durch Generalisierung

- die beiden Regeln unterscheiden sich in einem Label

```
[[_='NP'-nom,A='PPER'-nom],    [[_='NP'-nom,C='NN'-nom],
                 ^^^^^^^^^^                     ^^^^^^^^
 [B='VVFIN'- --],               [B='VVFIN'- --],
 [_='NP'-akk,C='NN'-akk]]       [_='NP'-akk,C='NN'-akk]]
```

- kompaktere Darstellung (Disjunktion von Labels)

```
[[_='NP'-nom,A=['PPER'-nom,'NN'-nom]],
                 ^^^^^^^^^^^^^^^^^^^
 [B='VVFIN'- --],
 [_='NP'-akk,C='NN'-akk]]
```

- iterative Generalisierung führt zu:

```
[[_='NP'-nom,A=['PDS'-nom,'PPER'-nom,'NN'-nom,'KON'-'CD','NE'-nom]],
 [B='VVFIN'- --],
 [_='NP'-akk,C=['KON'-'CD','NN'-akk,'PRF'-akk]]]).
```

Was sagt die Empirie dazu

- Reduktion der Regelmenge: 267 $\rightarrow$ 81

	P-Train	R	P-Test	R
2	83.0	98.4	79.2	74.2
1	99.8	99.7	99.3	86.9

1001+g

	pred	rec	f-meas
train	91.4	99.1	95.1
test	89.3	80.6	84.7

1001+g

	P-Train	R	P-Test	R
2	84.2	98.4	79.1	70.9
1	99.8	99.7	99.3	86.6

1001n

	pred	rec	f-meas
train	92.0	99.1	95.4
test	89.2	78.8	83.7

1001n

- Leichte Verbesserung (Recall besser, Präzision schlechter)

Regelgewichtung

- leider: widersprüchliche Regeln sind (wohl) unvermeidbar (statistisches Parsing)
- aber: es ist unsinnig alle Regeln einer solchen inkonsistenten Regelklasse (simultan) anzuwenden
- daher: am besten wenden wir nur die beste an
- und: die beste Regel ist die, welche die meisten positiven Beispiele hat
  - das ist eine einfache Häufigkeitszählung
  - die restlichen Regeln kann man löschen (sie kommen nie zur Anwendung)
- diese Strategie ergibt sehr gute Resultate
- evtl. Regel unter Konfidenzwert löschen (Erhöhung der Präzision, Absenken der Ausbeute)

Vergleich

	P-Train	R	P-Test	R
2	98.0	97.4	97.7	70.1
1	99.8	99.7	99.3	86.6

1001+w

	pred	rec	f-meas
train	98.9	98.6	98.7
test	98.5	78.4	87.3

1001+w

	P-Train	R	P-Test	R
2	84.2	98.4	79.1	70.9
1	99.8	99.7	99.3	86.6

1001n

	pred	rec	f-meas
train	92.0	99.1	95.4
test	89.2	78.8	83.7

1001n

	P-Train	R	P-Test	R
2	99.2	98.1	97.3	65.2
1	1.0E+02	99.3	99.5	76.1

1001+dt

	pred	rec	f-meas
train	99.6	98.7	99.1
test	98.4	70.7	82.3

1001+dt

Related Work

- Grammatical Relation (GR) Finding
  - Sabine Buchholz (2002): Memory-Based Grammatical Relation Finding. PhD thesis, Tilburg University
  - Buchholz-Szenario: TIMBL, Ziel ist die Identifizierung GR über einem *shallow* parse
  - BitPar-Szenario: GR über Kasus teilweise rekonstruierbar (evtl. aber fehlend, mehrfach oder falsch), GR manchmal nicht gegeben (bei Verb-Komplementen). GR-Label interessiert nicht, Hypostasierung einer kanonischen Argumentreihenfolge.
- Semantic Role Labeling (Zuweisung von semantischen Rollen zu Phrasen (!))
  - Gildea & Jurafsky (2002): statistisch, FrameNet (>40.000 Sätze), Collins Parser
  - Punyakanok et al. (2004): Chunking, Tagging und ILP
- Literatur:
  - Vasin Punyakanok, Dan Roth, Wen-tau Yih,& Dave Zimak (2004) (Coling ’04): Semantic Role Labeling via Integer Linear Programming Inference, Coling ’04
  - Daniel Gildea and Daniel Jurafsky (2002). Automatic Labeling of Semantic Roles. Computational Linguistics 28:3, 245-288.
  - Conference on Computational Natural Language Learning (CoNLL-2004, http://cnts.uia.ac.be/conll2004/, Shared Task: Semantic Role Labeling)

Fazit

- mit der erreichten Präzision kann man (gut) leben
- Verbesserung der Ausbeute durch z.B. mehr Trainingsbeispiele
  - Hoffnung: Regelmenge konvergiert, Varianz der Parsingstrukturen ist finit
- Erweiterung auf andere Extraktionsaufgabe (z.B. Passiv, Adjektivinterpretation) ist einfach, es braucht nur entsprechende Trainingsbeispiele
  - Übergenerelle Regeln können immer mit DecTree-Mitteln eingeschränkt werden, so dass man die Präzision kontrollieren kann
  - die Kontrollmöglichkeiten bei der Ausbeute sind nicht so klar definiert, insbesondere die Erhöhung bei nur wenig sinkender Präzision
- Einschätzung: der Einsatz von ML-Verfahren (im Sinne von Ersetzen des Regellerners) bringt keine Verbesserung
- Behauptung: Fortschritte auf computerlinguistischer Seite sind viel wichtiger und durchschlagender als die Anwendung allgemeiner ML-Verfahren über den Resultaten flacher CL-Verfahren (Chunking, Tagging)

Rückblick

- zuerst war Gojol (+ LoPar) da und die Ergebnisse mit der einfachen Version des Regellerners waren eher schlecht
- 1. Grund: Gojol erkennt nur Subjekt
  - dem Versuch, mit Gertwol andere grammatische Funktionen zu erkennen, war mässiger Erfolg beschieden
- 2. Grund: Gojols Phrasenstrukturen sind tendentiell chaotischer als die von BitPar (unbestätigte Annahme)
- Vermutung: Gojols Dependenzbäume sind besser
- Lopar (Schulte im Walde Grammatik): minimaler Recall, da die Bäume sehr tief, die Label sehr zahlreich, Regeln also fast immer idiosynkratisch
- daher Versuch mit *Machine Learning* Verfahren (und Gojol)
- zur Auswahl: C4.5 (*decision tree*) und Timbl (IBL=*instance-based learning*, bzw. *k nearest neighbour*, bzw. *memory-based learner*)
- Mängel dieser Verfahren: Einbau linguistischer Constraints
- dies ist mit ILP möglich, ... aber das ist eine andere Geschichte

Appendix: Gojols Parser

- Probabilistischer Parser
  - trainiert mit (Teilen der) NEGRA Treebank
  - liefert trotzdem keine Negrabäume
  - zusätzliche, teilweise andere syntaktische Labels
  - berechnet Dependenzen, keine Auszeichnung dir. und indir. Objekte
- Gojol ist leidlich schnell
- Gojol ist robust, aber eigenwillig bei Sonderzeichen
- Gojol gibt die n besten Analysen (im Prinzip)
- Syntaktische Evaluierung: Clematide (2002)

Appendix: BitPar

- Probabilistischer Parser
  - CYK-artiger Parser für PCFG (Schmid, 2004)
  - verwendet Bitvektoren zur Speicherung
  - besonders geeignet für grosse (stark ambige) Baumbankgrammatiken und lange Eingabesätze (Schmid)
  - Bitvektoroperationen zur Parallelisierung z.B. der Kombination zweier Konstituenten anhand einer binären Regel
- BitPar minimiert die Kosten der Kantenbildung (edges)
- BitPar gibt alle Analysen oder Viterbi Parse (bester Score)

Appendix: Michael Schiehlens CFG

- nicht lexikalisiertes probabilistisches Parsing (!)
- trainiert mit NEGRA Treebank, siehe Schliehen (2004)
- liefert keine Negrabäume (schade?)
  - zusätzliche (speziellere) POS Tags (VSFIN)
  - berechnet Kasus von NPen
- Grammatikabdeckung: 1124 (lange) Sätze aus “Tod in Venedig” - 50 haben “no parse”
- BitPar schafft mit der Schiehlen Negra-Treebank-PCFG 1000 Sätze in 1 Stunde

Appendix: LoPar Parser und Schulte im Walde CFG

LoPar:

- Parser für lexikalisierte PCFG und kopf-lexikalisierte PCFG
  - symbolisches oder statistisch Parsing, siehe Schmid (2000)
  - auch: Tagging, Chunking

Grammatik:

- Sabine Schulte im Walde (2003): verb-zentrierte CFG für das Deutsche
  - X-Bar-Schema (d.h. grässlich tiefe Bäume),
  - eigenwillige syntaktische Label (Verbrahmen und Kasus Teil der Labels)
  - Zusatzinformation kann aber nutzbar gemacht werden (z.B. Kasus)
- ME (Maximum Expectation) optimiert anhand eines 22 Millionen Worte Korpus
- Leider schafft die Grammatik nur 60-70% der NEGRA-Sätze - das ist zuwenig
- Evaluierung siehe: Clematide (2002)