

Parsning

Språkteknologi DH2418
Ola Knutsson

Varför parsning?

Grammatikkontroll
Fråge-svarssystem
Maskinöversättning
Semantisk analys (vad menas?)
Testa grammatikformaliser och
“grammatiker” (undvika länsstolslingvistik)

Parsningens olika nivåer

- Deep vs. shallow (djup vs. ytlig)
- Full vs. partial (fullständig vs. partiell)

Grundförutsättningar för fullständig parsning

- Vänster till höger?
- Är orden ordklasstaggade? Entydiggjorda?
- Från första ordet i satsen till sista ordet i satsen.
- Trädet måste ha en rot.
- I vilken ordning tolkas grammatikreglerna?
- Hitta alla träd (alla delträd). Parsning som sökning.
- Backtracking.

Parserns huvuduppgifter

- Känna igen orden som bildar grammatiska satser i indata
- Tolka grammatiken
- Tilldela struktur
- Styra hur sökningen sker

Problem:

Språkets flertydighet (och grammatikens!)

Toppen-ner parsning

- Mål-driven (goal-directed) sökning
- Utgår från toppnoden (S)
 - Toppen-ner, djupet-först
 - Toppen-ner, bredden-först
 - I vilken ordning tillämpas reglerna?

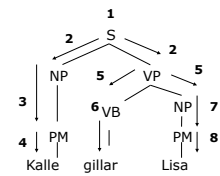
Botten-upp parsning

Data-driven sökning

Utgår från strängen (data/orden)

Botten-upp, bredden-först

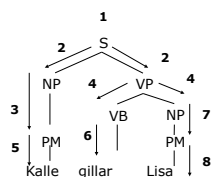
Toppen-ner, djupet-först



S → NP VP
VP → VB NP
VP → VB
NP → PM
VB → gillar
PM → Lisa
PM → Kalle

- 1) S,
- 2) NP VP
- 3) PM VP
- 4) Kalle VP
- 5) Kalle VB NP
- 6) Kalle gillar NP
- 7) Kalle gillar PM
- 8) Kalle gillar Lisa
- 9) \$

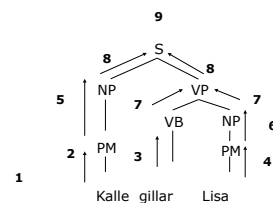
Toppen-ner, bredden-först



S → NP VP
VP → VB NP
VP → VB
NP → PM
VB → gillar
PM → Lisa
PM → Kalle

- 1) S,
- 2) NP VP
- 3) PM VP
- 4) PM VB NP
- 5) Kalle VB NP
- 6) Kalle gillar NP
- 7) Kalle gillar PM
- 8) Kalle gillar Lisa
- 9) \$

Botten-upp, bredden-först



S → NP VP
VP → VB NP
VP → VB
NP → PM
VB → gillar
PM → Lisa
PM → Kalle

- 1) Kalle gillar Lisa
- 2) PM gillar Lisa
- 3) PM VB Lisa
- 4) PM VB PM
- 5) NP VB PM
- 6) NP VB NP
- 7) NP VP
- 8) S
- 9) \$

Flertydig lexikal kategori: Ordklasstagning med disambiguering
Strukturell flertydighet: vi får flera parseträd till en sats.

Hängningsproblem: PP-attachment problem

Jag ser mannen med kikaren.

Han äter pizza med gaffel. Jämför:

Han äter pizza med ansjovis.

Coordination: Jämför:

Hunden eller katten äter fisken och köttet.

Hunden äter köttet och katten äter fisken.

NP problem: *Igår sjöng jag och Jonas Mozart.*

[jag och Jonas Mozart] eller [jag och Jonas] [Mozart]:

Vilka är problemen med dessa två angreppssätt?

Toppen-ner:

- Vänsterrekursion
- Flertydighet från ovan
- Upprepad parsning av delträd (prövar onödiga hypoteser)

Botten-upp:

- Kan söka efter topp-noder och grannkonstituentier som inte finns.
- Försöker expandera flertydiga tolkningar av lexikala enheter.

Earleyparsning

- Dynamisk programmering, i det här fallet att spara undan delträd när de påträffas, för senare användning. Delträden läggs i en s.k. chart
- Använder look-up istället för att parsas igen
 - Look-up är mindre komplext än parsning
 - Earley-parseern använder en *Chart*, en lista med N+1 ingångar
 - Alla ord i meningen motsvarar en position i charten där de partiella träd som byggs upp hittills lagras
 - När strängen genomlöpts innehåller charten en kompakt representation av flera möjliga analyser av indata
 - Varje ingång i charten har tre typer av information:
 - delträd
 - hur långt man kommit i trädet
 - position i charten

Tre operatorer

Prediktor: att skapa nya tillstånd som representerar toppen-ner förväntningar som genereras under parsningsprocessen. Prediktorn tillämpas på varje tillstånd som har en icke-terminal till höger om ”punkten” som inte är en ordklasskategori. Läger upp tillstånd in den ”chart” som processas.

Skanner:

När ett tillstånd har en ordklasskategori till höger om ”punkten”, används skannern för att granska indata om det finns något ord med denna ”förutsagda” ordklasskategori. Här löses en hel del flertydighet upp. Läger upp tillstånd i en ”chart”.

Kompletterare: När ett tillstånd når slutet till höger i en regel backar komplementeraren tillbaka och fyller i kedjan av delanalyser. När t.ex. tillståndet $NP \rightarrow DT\ NN$ [1,3] letar komp. efter tillstånd som slutar på 1, t.ex. $VP \rightarrow VB\ NP$ [0,1]. Detta resulterar i sin tur i ett nytt komplett tillstånd $VP \rightarrow VB\ NP$ [0,3]. Läger upp tillstånd in den ”chart” som processas.

Tre exempeltillstånd

$S \rightarrow .\ VP$ [0,0]

En toppen-ner prediktion för ett speciellt S

Konstituenten skall börja i startposition av indata (första 0), andra 0 anger ”punktens” placering

$NP \rightarrow Dt\ .\ Nn$ [1,2]

En NP börjar i position 1, en Dt har parsats, och att en Nn står i tur.

$VP \rightarrow Vb\ NP$ [0,3]

Det finns ett träd VP som spänner över hela indata

Grammatik:

$S \rightarrow NP\ VP$

$S \rightarrow VP$

$NP \rightarrow Dt\ Nn$

$NP \rightarrow Pm$

$VP \rightarrow Vb$

$VP \rightarrow Vb\ NP$

$VP \rightarrow Vb\ NP\ PP$

$PP \rightarrow Prep\ NP$

Lexikon:

$Dt \rightarrow en\ | \ ett\ | \ de\ | \ den$

$Nn \rightarrow köp\ | \ flyg\ | \ måltid\ | \ biljett$

$Vb \rightarrow köp\ | \ flyga\ | \ köpa$

$Prep \rightarrow från\ | \ till$

$Pm \rightarrow Arlanda\ |$

$Landvetter$

Chart[0]

$\gamma \rightarrow .\ S$

[0, 0] Dummy start

$S \rightarrow .\ NP\ VP$

[0, 0] P

$NP \rightarrow .\ Dt\ Nn$

[0,0] P

$NP \rightarrow .\ Pm$

[0,0] P

$S \rightarrow .\ VP$

[0,0] P

$VP \rightarrow .\ Vb$

[0,0] P

$VP \rightarrow .\ Vb\ NP$

[0,0] P

Chart[1]

Vb → köp . [0,1] s
VP → Vb . [0,1] c
S → VP . [0,1] c
VP → Vb . NP [0,1] c
NP → . Dt Nn [1,1] p
NP → . Pm [1,1] p

Chart[2]

Dt → en . [1,2] s
NP → Dt .Nn [1,2] c

Chart[3]

Nn → biljett . [2,3] s
NP → Dt Nn . [1,3] c
VP → Vb NP. [0,3] c
S → VP . [0,3] c

Partial och shallow parsning (Partiell och Ytparsning)

- Ofta utan fullständiga träd
- Ger analyser av de allra flesta meningar, robust?
- Behöver man veta satsen fullständiga uppbyggnad? Är det t.ex. endast alla NP man är ute efter?
- Chunking
- Bracketing
- Ytorienterad dependensparsning – ligger mitt emellan, kan innehålla begränsad semantisk information.

Robust parsning

En robust parser skall ge en korrekt eller användbar analys av 90 % av meningar (Briscoe). Åtminstone följande tre problem måste lösas:

1. Tokenisering/Sentensering, vad är en mening?
2. Disambiguation, att välja rätt analys bland alla syntaktiskt möjliga
3. Undergeneration, eller att hantera fall som ligger utanför systemets lexikala och syntaktiska täckning.

Robust parsning II

Ett system måste vara robust mot följande (enligt Menzel 1997):

- Oförutsägbarheten hos autentiska data
- brusiga miljöer (om talaren befinner sig i en larmig miljö)
- variationer mellan talare; dialekt och sociolekt
- ”felaktig” indata jämfört mot språknormen
- ofullständig indata, fragmentarisk
- okända ord och termer
- andraspråkstalare

Jabberwocky (Källgren 1992)

De iggla skviggarna trassade vombigt i den harliga gopen.

- Hurk, najdade den ena skviggen, maffar pem en bunne?
- Snå, bebbade den umre skviggen aldrig. Snafs på nerfen.
- Lytrik snafs det?
- Pej gemmer det fraskar för kloxigt.
- Så vatrik jeggade skviggen snafen bunne.

Gunnel Källgrens Morp-parser

- Den ytliga informationen i en text är underskattad.
- Minimalt lexikon med funktionsord
- Mönstermatchning av morfologiska och ytsyntaktiska mönster.
- Märker orden med ordklass, känner igen enkla NP och PP. (extrauppgift?)

Parsning med Finite-state

Karttunen, Chanod, Grefenstette & Schiller (1996)

Regular Expressions for Natural Language Engineering

- Välkänt och beprövat för morfologisk analys och generering. Men fungerar också för partiell parsning. Indata=ordklasstaggad text
- Ganska enkla operatorer kan användas till sådant som skulle vara komplicerat att uttrycka i andra språk/formalismer:

A, B och C är reguljära uttryck.

A | B Union

A B konkaternering

(A) Optionellt A; union med tomma strängen

A+ Iteration; en eller flera konkaterneringar av A

A* Kleene star; (A+)

~A komplementet(negation)

A & B Snittet, skärning

A – B Subtraktion

restriction: A => B _ C

replacement: A -> B

longest match replacement: A @-> B ... C

Kaskader av regler (regex)

Indata: de+DT glada+JJ studenterna+NN

define AP [Ord JJ]

define NP = [Ord [PN] | [(DT) ([AP]+) NN] | [PS ([AP]+) NN]]

regex NP @-> "[NP " ... " NP]"

Utdata: [NP de+DT glada+JJ studenterna+NN NP]

Indata: [NP de+DT glada+JJ studenterna+NN NP]

define HuvudN [NN => _NP]

regex HuvudN @-> "[HuvudN" ... "HuvudN]"

Utdata: [NP de+DT glada+JJ [HuvudN[studenterna+NN]HuvudN] NP]

Constraint Grammar (CG)

Parsningsalgoritm:

Tre delar:

1. Morfologisk disambiguering
2. Bestäm satsgränsern
3. Ytsyntaktisk disambiguering

viktigaste JJ
redskapen NN
vid PP
ympning NN
är VB
annars AB
papper NN
och KN
penna NN
, MID CLB
menade VB
han PN
. MAD CLB

32

Disambiguering skall ske på all tre delproblemen

- Utgå ifrån värsta fallet: tilldelning av alla möjliga tolkningar (enligt lexikon)
- Varje regel skall förkasta så många tolkningar som möjligt.
- Regel efter regel tillämpas så länge det går – så länge det finns flertydighet kvar som kan tas bort med reglerna i grammatiken

Regelsyntax

Efter ordklassdisambiguering genomförs en morfosyntaktisk mappning:

<matchningsankare kontextvillkor
syntaktiska_taggar>

Ex: ((N) ((-IC PREP) (1 <<<)) (@<P))

N tilldelas den syntaktiska funktionen prepositionskomplement om N föregås av en preposition och följs av meningsslut.

Syntaktiska regler

=s! väljer en tolkning och förkastar övriga

=s0 förkastar en tolkning sparar övriga

Ej matchning av ordformer

@w är standard – vilket ord som helst

Mannen NN UTR SIN DEF @SUBJ @OBJ
@IOBJ
spelar VB PRS ACTIVE @+FMV
fotboll NN UTR SIN IND @SUBJ @OBJ
@IOBJ @<P
. CLB <PUNCT> @

36

SWECG-regler

`((@w =s! (@SUBJ) (1 @+FMV) (NOT *-1 @SUBJ) (NOT *1 @SUBJ))`

`((@w =s0 (@<P) (NOT *-1 PREP) (NOT *1 PREP)`

`((@w =s! (@+OBJ) (-1 @+FMV) (NOT *-1 @OBJ) (NOT *1 @OBJ))`

GTA – Granska Text Analyzer

Grunddesign:

- Arbetar på ordklasstaggad indata (entydiggjord)
- Tar ibland inte hänsyn till grammatisk/ogrammatisk
- Meningar delas upp i satser, därefter frasanalys. Inga fullständiga träd, varför?

Fraskategorier

NP: Han såg **den lilla mannen** på bänken

VC: Han **har spelat** kort hela natten

PP: Han såg spår **i sanden**

AP: Han ogillade **små vita** lögnar

ADVP: Han vill **inte** gå på bio.

INFP: Han tycker om **att spela**.

Regelmatchning/parsning

Varje regel kan appliceras var som helst i texten. Toppen-ner, djupet-först, vänster-höger
Stannar så fort något regelement ej matchar
Optimerad matchning med statistiska medel.
Reglerna förhandsgranskas och ett ”regelankare” väljs ut.

Delregler matchas endast när det behövs.
Alla delträd som matchas sparas undan.

En minimal NP

`Npmin@`

`X(wordcl=dt | wordcl=ps)?,`

`Y(wordcl=jj)*,`

`Z(wordcl=nn | wordcl=pm)`

`-->`

`action(help, gender:=Z.gender, num:=Z.num, if
X.wordcl=ps then spec:=def else spec:=Z.spec
end, case:=Z.case)`

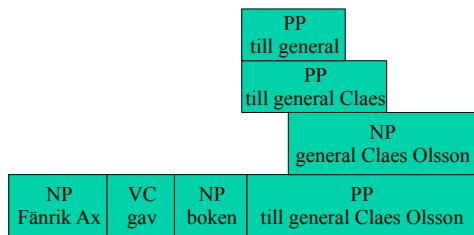
`Npkonj@`

```
{  
  (Npmin/X) (),  
  Y(wordcl=kn),  
  (Npmin/Z) ()  
-->  
action(help, num:=plu)  
}
```

`NP@`

```
{  
  Npkonj --> action(help);  
  ...;  
  Npmin --> action(help) }
```

The Tetris Algorithm



Webbtips och extrauppgifter

Functional Dependency Parser:

- www.connexor.com

GTA: <http://skrutton.nada.kth.se/grim/form.html>

Bygg en enkel Morp-parser

Bygg en enkel dependensparser