Applications of Tree Automata Theory
Lecture II: Parsing — Basics and Evaluation

Andreas Maletti

Institute of Computer Science
Universität Leipzig, Germany

on leave from: Institute for Natural Language Processing
Universität Stuttgart, Germany

maletti@ims.uni-stuttgart.de

Yekaterinburg — August 23, 2014

Roadmap

1. Theory of Tree Automata
2. Parsing — Basics and Evaluation
3. Parsing — Advanced Topics
4. Machine Translation — Basics and Evaluation
5. Theory of Tree Transducers
6. Machine Translation — Advanced Topics

Always ask questions right away!

Parsing

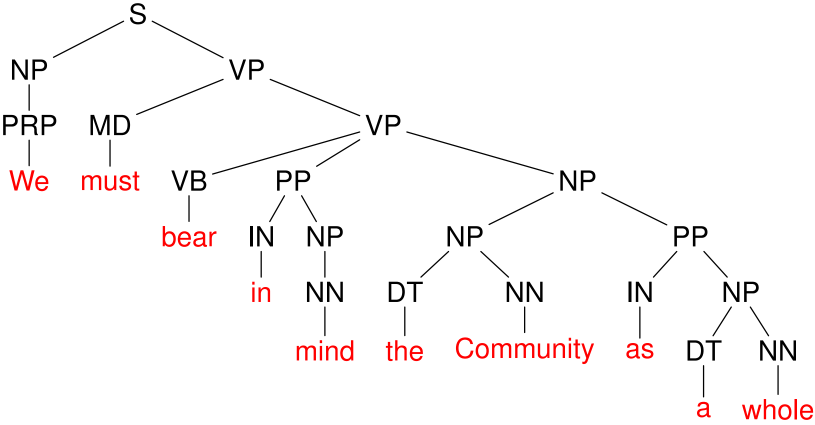
Problem [intuitive]

Parsing is the process of analyzing the syntactic structure of a sentence yielding the parse tree

- syntactic structure of a language linguistically motivated
- ideally expressed by a grammar

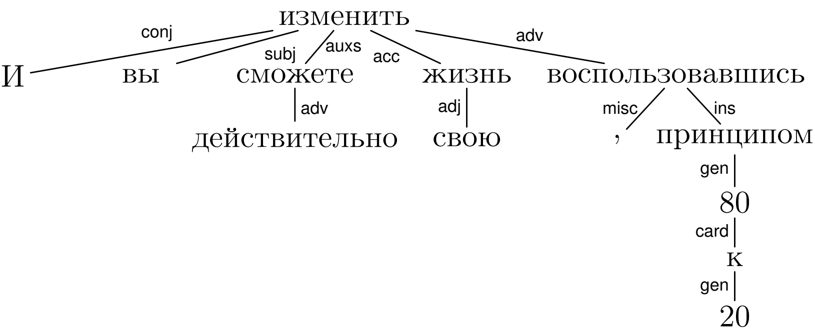
Parsing

We must bear in mind the Community as a whole



Parsing

И вы действительно сможете изменить свою жизнь, воспользовавшись принципом 80 к 20



Linguistic Background

Constituency Parsing

- each word is a lexical item
- each lexical item has a grammatical function
  $\rightarrow$ part-of-speech (POS) (noun, verb, etc.)
- they combine to phrases that form syntactic categories
  $\rightarrow$ constituents (noun phrase, verb phrase, etc.)

Linguistic Background

Constituency Parsing

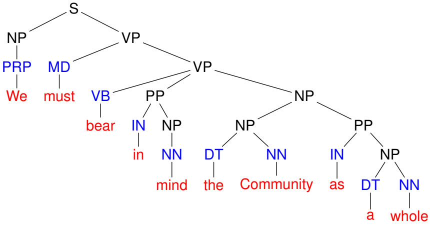
- each word is a lexical item
- each lexical item has a grammatical function
  $\rightarrow$ part-of-speech (POS) (noun, verb, etc.)
- they combine to phrases that form syntactic categories
  $\rightarrow$ constituents (noun phrase, verb phrase, etc.)

Other linguistic theories

- dependency grammars
- combinatory categorial grammars
- ...

Linguistic Background

red = lexical items   blue = POS   black = constituents



POS

- PRP = personal pronoun
- MD = modal (verb)
- VB = verb (base form)

Constituents

- S = sentence
- NP = noun phrase
- VP = verb phrase

Parsing

Problem [realistic]

- assume a hidden $g\colon Q^* \to T_\Sigma(Q)$ — reference parser
- given a finite set $T \subseteq T_\Sigma(Q)$ generated by g — example parse trees
- develop a system representing $f\colon Q^* \to T_\Sigma(Q)$ approximating g — parser

Parsing

Problem [realistic]

- assume a hidden $g\colon Q^* \to T_\Sigma(Q)$ reference parser
- given a finite set $T \subseteq T_\Sigma(Q)$ example parse trees
  generated by g
- develop a system representing $f\colon Q^* \to T_\Sigma(Q)$ parser
  approximating g

Parsing

Problem [realistic]

- assume a hidden $g\colon Q^* \to T_\Sigma(Q)$ — reference parser
- given a finite set $T \subseteq T_\Sigma(Q)$ generated by g — example parse trees
- develop a system representing $f\colon Q^* \to T_\Sigma(Q)$ approximating g — parser

Clarification

- T generated by $g \iff T = g(L)$ for some finite $L \subseteq Q^*$
- for approximation we could use $|\{w \in Q^* \mid f(w) = g(w)\}|$

Statistical Parser Training

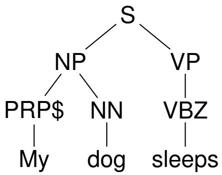
Training

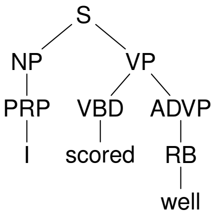
Statistical Parser Training

Observations

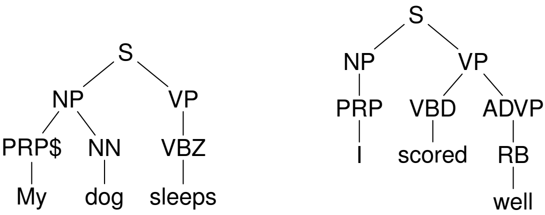
- linguistic knowledge encoded in training set $T \subseteq T_\Sigma(Q)$
- we want to build grammars that can generate $T' \supseteq T$
  we follow the historical development
  1. local tree grammars — LTG
  2. tree substitution grammars — TSG
  3. regular tree grammars — RTG

Statistical Parser Training





Statistical Parser Training



LTG Production Extraction

simply read of CFG productions:

S $\longrightarrow$ NP VP	NP $\longrightarrow$ PRP$ NN
PRP$ $\longrightarrow$ My	NN $\longrightarrow$ dog
VP $\longrightarrow$ VBZ	VBZ $\longrightarrow$ sleeps
NP $\longrightarrow$ PRP	PRP $\longrightarrow$ I
VP $\longrightarrow$ VBD ADVP	VBD $\longrightarrow$ scored
ADVP $\longrightarrow$ RB	RB $\longrightarrow$ well

Statistical Parser Training

Observations

- LTG offer unique explanation on tree level
- but ambiguity on the string level

$\rightarrow$ weighted productions

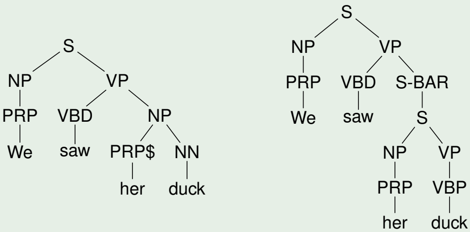
Statistical Parser Training

Observations

- LTG offer unique explanation on tree level
- but ambiguity on the string level

→ weighted productions

Illustration



Statistical Parser Training

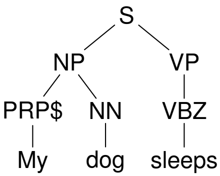
Definition

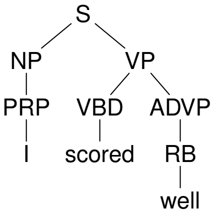
A weighted local tree grammar (wLTG) is a weighted CFG $G = (N, Q, S, P, \mathrm{wt})$

- finite set N — *nonterminals*
- finite set Q — *terminals*
- $S \subseteq N$ — *start nonterminals*
- finite set $P \subseteq N \times T_N(Q)$ — *productions*
- mapping $\mathrm{wt} \colon P \to [0, 1]$ — weight assignment

It will compute the weighted derivation trees of the wCFG

Statistical Parser Training





LTG Production Extraction

simply read of CFG productions and keep counts:

S ⟶ NP VP (2)	NP ⟶ PRP$ NN (1)
PRP$ ⟶ My (1)	NN ⟶ dog (1)
VP ⟶ VBZ (1)	VBZ ⟶ sleeps (1)
NP ⟶ PRP (1)	PRP ⟶ I (1)
VP ⟶ VBD ADVP (1)	VBD ⟶ scored (1)
ADVP ⟶ RB (1)	RB ⟶ well (1)

Statistical Parser Training

LTG Production Extraction

normalize counts: (here by left-hand side)

$$\mathrm{S} \longrightarrow \mathrm{NP\ VP} \quad (2)$$

$$\mathrm{NP} \longrightarrow \mathrm{PRP\$\ NN} \quad (1) \qquad \mathrm{NP} \longrightarrow \mathrm{PRP} \quad (1)$$

$$\mathrm{PRP\$} \longrightarrow \mathrm{My} \quad (1)$$

$$\mathrm{NN} \longrightarrow \mathrm{dog} \quad (1)$$

$$\mathrm{VP} \longrightarrow \mathrm{VBZ} \quad (1) \qquad \mathrm{VP} \longrightarrow \mathrm{VBD\ ADVP} \quad (1)$$

$$\mathrm{VBZ} \longrightarrow \mathrm{sleeps} \quad (1)$$

$$\mathrm{PRP} \longrightarrow \mathrm{I} \quad (1)$$

$$\mathrm{VBD} \longrightarrow \mathrm{scored} \quad (1)$$

$$\mathrm{ADVP} \longrightarrow \mathrm{RB} \quad (1)$$

$$\mathrm{RB} \longrightarrow \mathrm{well} \quad (1)$$

Statistical Parser Training

LTG Production Extraction

normalize counts: (here by left-hand side)

$$\mathrm{S} \xrightarrow{1} \mathrm{NP\ VP}$$

$$\mathrm{NP} \xrightarrow{0.5} \mathrm{PRP\$\ NN} \qquad \mathrm{NP} \xrightarrow{0.5} \mathrm{PRP}$$

$$\mathrm{PRP\$} \xrightarrow{1} \mathrm{My}$$

$$\mathrm{NN} \xrightarrow{1} \mathrm{dog}$$

$$\mathrm{VP} \xrightarrow{0.5} \mathrm{VBZ} \qquad \mathrm{VP} \xrightarrow{0.5} \mathrm{VBD\ ADVP}$$

$$\mathrm{VBZ} \xrightarrow{1} \mathrm{sleeps}$$

$$\mathrm{PRP} \xrightarrow{1} \mathrm{I}$$

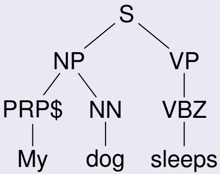
$$\mathrm{VBD} \xrightarrow{1} \mathrm{scored}$$

$$\mathrm{ADVP} \xrightarrow{1} \mathrm{RB}$$

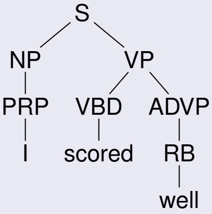
$$\mathrm{RB} \xrightarrow{1} \mathrm{well}$$

Statistical Parser Training

Weighted parses



weight: 0.25



weight: 0.25

Weighted LTG productions

(only productions with weight $\neq 1$)

$$\text{NP} \xrightarrow{0.5} \text{PRP\$ NN} \qquad \text{NP} \xrightarrow{0.5} \text{PRP}$$

$$\text{VP} \xrightarrow{0.5} \text{VBZ} \qquad \text{VP} \xrightarrow{0.5} \text{VBD ADVP}$$

Statistical Parser Training

Statistical Parsing Approach

given sentence *w*, return highest-scoring parse for *w*

Statistical Parser Training

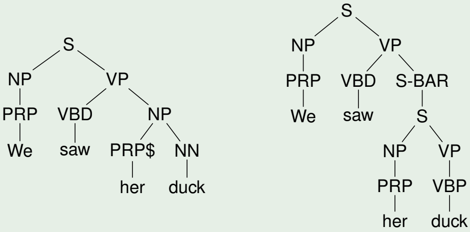
Statistical Parsing Approach

given sentence *w*, return highest-scoring parse for *w*

Consequence

The first parse should be prefered

("duck" more frequently a noun, etc.)



Statistical Parser Training

Observations

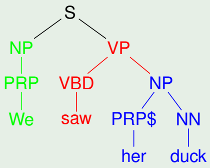
- works similarly for TSG
- needs additional cutting strategy to select fragments

Statistical Parser Training

Observations

- works similarly for TSG
- needs additional cutting strategy to select fragments

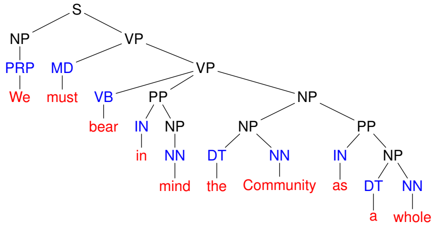
Illustration



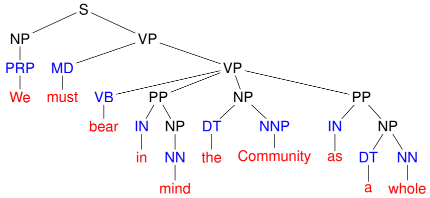
Evaluation

Parsing — Evaluation

BERKELEY parser [Reference]:



CHARNIAK-JOHNSON parser:



Parsing — Evaluation

Definition (ParseEval measures)

- precision = number of correct constituents (heading the same phrase as in reference) divided by number of all constituents in parse

Parsing — Evaluation

Definition (ParseEval measures)

- precision = number of correct constituents (heading the same phrase as in reference) divided by number of all constituents in parse
- recall = number of correct constituents divided by number of all constituents in reference

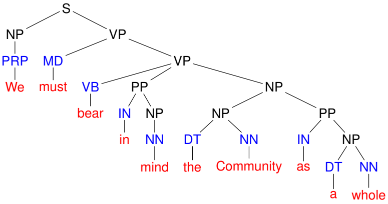
Parsing — Evaluation

Definition (ParseEval measures)

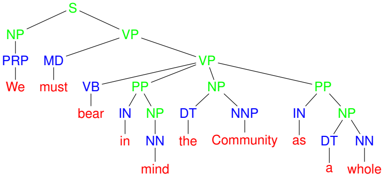
- precision = number of correct constituents
  (heading the same phrase as in reference)
  divided by number of all constituents in parse
- recall = number of correct constituents
  divided by number of all constituents in reference
- combined measure

$$F_\alpha = (1 + \alpha^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\alpha^2 \cdot \text{precision} + \text{recall}}$$

Reference



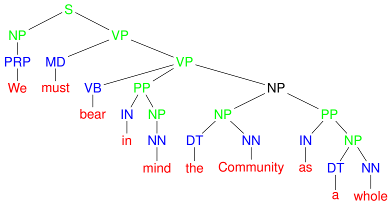
Parser output



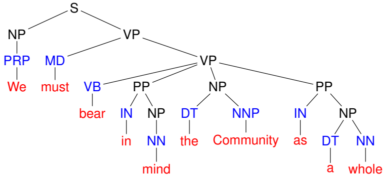
- precision $= \frac{9}{9} = 100\%$

Parsing — Evaluation

Reference



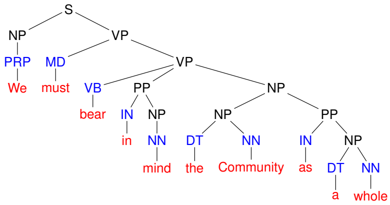
Parser output



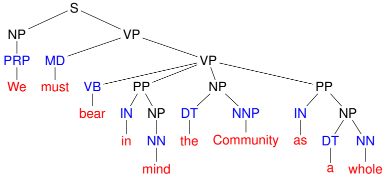
- precision $= \frac{9}{9} = 100\%$
- recall $= \frac{9}{10} = 90\%$

Parsing — Evaluation

Reference



Parser output



- precision $= \frac{9}{9} = 100\%$
- recall $= \frac{9}{10} = 90\%$
- $F_1 = 2 \cdot \frac{1 \cdot 0.9}{1 + 0.9} = 95\%$

Parser Experiments

Standardized Setup

- training data: PENN treebank Sections 2–21
  (articles from the WALL STREET JOURNAL)
- development test data: PENN treebank Section 22
- evaluation data: PENN treebank Section 23

Parser Evaluation

Experiment [Post, Gildea, 2009]

type	size	precision	recall	F_1
CFG	46k	75.37	70.05	72.61
“spinal” TSG	190k	80.30	78.10	79.18
“minimal subset” TSG	2,560k	76.40	78.29	77.33

Parser Evaluation

Experiment [Post, Gildea, 2009]

type	size	precision	recall	F_1
CFG	46k	75.37	70.05	72.61
“spinal” TSG	190k	80.30	78.10	79.18
“minimal subset” TSG	2,560k	76.40	78.29	77.33

These are bad compared to the state-of-the-art!

State-of-the-Art Models

Parser Evaluation

State-of-the-art models

- context-free grammars with latent variables ($\mathrm{CFG}_{\mathrm{lv}}$)
  [COLLINS, 1999]
  [KLEIN, MANNING, 2003]
  [PETROV, KLEIN, 2007]
- tree substitution grammars with latent variables ($\mathrm{TSG}_{\mathrm{lv}}$)
  [SHINDO et al., 2012]
- other models

Grammars with Latent Variables

Definition

A grammar with latent variables is (grammar with relabeling)

- a grammar G generating $L(G) \subseteq T_\Sigma(Q)$
- a (total) mapping $\rho\colon \Sigma \to \Delta$ functional relabeling

Remark

We use X-n for symbols that are relabeled to X $\rho(\text{X-}n) = \text{X}$

Grammars with Latent Variables

Definition (Semantics)

$$L(G, \rho) = \rho(L(G)) = \{\rho(t) \mid t \in L(G)\}$$

Language class: REL($\mathcal{L}$) for language class $\mathcal{L}$

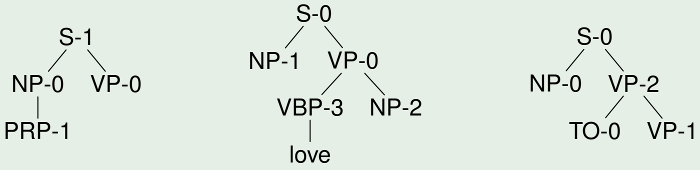
Grammars with Latent Variables

Definition (Semantics)

$$L(G, \rho) = \rho(L(G)) = \{\rho(t) \mid t \in L(G)\}$$

Language class: REL($\mathcal{L}$) for language class $\mathcal{L}$

Example (Typical fragments)



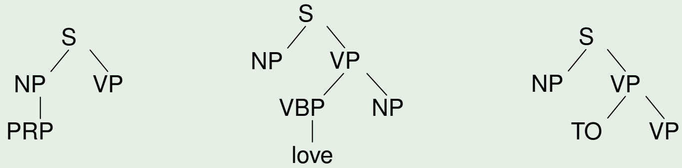
Grammars with Latent Variables

Definition (Semantics)

$$L(G, \rho) = \rho(L(G)) = \{\rho(t) \mid t \in L(G)\}$$

Language class: REL($\mathcal{L}$) for language class $\mathcal{L}$

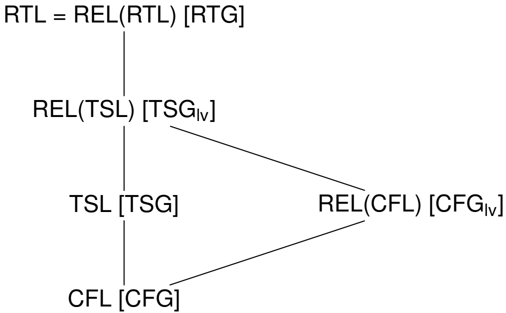
Example (Typical fragments)



Grammars with Latent Variables

Theorem

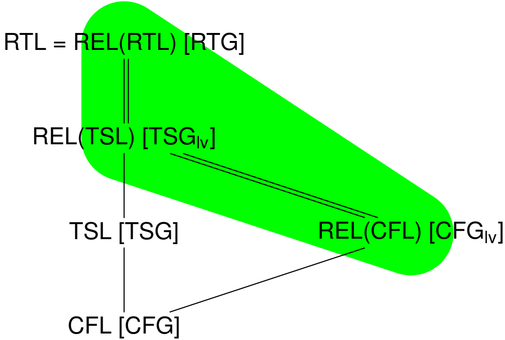
REL(LTL) $=$ REL(TSL) $=$ REL(RTL) $=$ RTL



Grammars with Latent Variables

Theorem

$\mathrm{REL}(\mathrm{LTL}) = \mathrm{REL}(\mathrm{TSL}) = \mathrm{REL}(\mathrm{RTL}) = \mathrm{RTL}$



Grammars with Latent Variables

Experiment [Shindo et al., 2012]

Grammar model	F_1 $\lvert w \rvert \leq 40$	F_1 full
TSG [Post, Gildea, 2009]	82.6	
TSG [Cohn et al., 2010]	85.4	84.7
CFG_{lv} [Collins, 1999]	88.6	88.2
CFG_{lv} [Petrov, Klein, 2007]	90.6	90.1
CFG_{lv} [Petrov, 2010]		91.8
TSG_{lv} (single) [Shindo et al., 2012]	91.6	91.1
TSG_{lv} (multiple) [Shindo et al., 2012]	92.9	92.4
Discriminative Parsers		
Carreras et al., 2008		91.1
Charniak, Johnson, 2005	92.0	91.4
Huang, 2008	92.3	91.7

Training of TA

Excursion: Berkeley Parser

Example (English grammar)

S-1 $\longrightarrow$ ADJP-2 S-1	$0.003545345598732 3125 \cdot 10^{0}$
S-1 $\longrightarrow$ ADJP-1 S-1	$2.108608433271444 \cdot 10^{-6}$
S-1 $\longrightarrow$ VP-5 VP-3	$1.6367163259885093 \cdot 10^{-4}$
S-2 $\longrightarrow$ VP-5 VP-3	$9.724998692152419 \cdot 10^{-8}$
S-1 $\longrightarrow$ PP-7 VP-0	$1.0686659961009547 \cdot 10^{-5}$
S-9 $\longrightarrow$ “ NP-3	$0.012551243773149695 \cdot 10^{0}$

Excursion: Berkeley Parser

Example (English grammar)

$$
\begin{array}{lr}
\text{S-1} \longrightarrow \text{ADJP-2 S-1} & 0.003545345598732125 \cdot 10^{0} \\
\text{S-1} \longrightarrow \text{ADJP-1 S-1} & 2.108608433271444 \cdot 10^{-6} \\
\text{S-1} \longrightarrow \text{VP-5 VP-3} & 1.6367163259885093 \cdot 10^{-4} \\
\text{S-2} \longrightarrow \text{VP-5 VP-3} & 9.724998692152419 \cdot 10^{-8} \\
\text{S-1} \longrightarrow \text{PP-7 VP-0} & 1.0686659961009547 \cdot 10^{-5} \\
\text{S-9} \longrightarrow \text{`` NP-3} & 0.012551243773149695 \cdot 10^{0}
\end{array}
$$

$\rightsquigarrow$ weighted tree automaton

Excursion: Berkeley Parser

Illustration

BERKELEY parser production:

$$\text{S-1} \longrightarrow \text{ADJP-2 S-1} \qquad 0.003545345598732312 5 \cdot 10^0$$

corresponds to tree automata production:

$$\text{S-1} \longrightarrow \text{S(ADJP-2, S-1)} \qquad 0.0035453455987323125 \cdot 10^0$$

Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions
   $\mathrm{S} \xrightarrow{c} \mathrm{NP}\ \mathrm{VP}$ corresponds to $\mathrm{S\text{-}1} \xrightarrow{c} \mathrm{S(NP\text{-}1, VP\text{-}1)}$
2. split all states and retrain

Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions

   $\mathrm{S} \xrightarrow{c} \mathrm{NP}\ \mathrm{VP}$ corresponds to $\mathrm{S\text{-}1} \xrightarrow{c} \mathrm{S(NP\text{-}1,\ VP\text{-}1)}$
2. split all states and retrain
3. check utility of splits

Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions
   $S \xrightarrow{c} NP\ VP$ corresponds to $S\text{-}1 \xrightarrow{c} S(NP\text{-}1, VP\text{-}1)$
2. split all states and retrain
3. check utility of splits
4. remerge if split not beneficial

Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions
   $\text{S} \xrightarrow{c} \text{NP VP}$ corresponds to $\text{S-1} \xrightarrow{c} \text{S(NP-1, VP-1)}$
2. split all states and retrain
3. check utility of splits
4. remerge if split not beneficial
5. back to 2 unless converged

Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions
   $\mathrm{S} \xrightarrow{c} \mathrm{NP}\ \mathrm{VP}$ corresponds to $\text{S-1} \xrightarrow{c} \mathrm{S}(\text{NP-1}, \text{VP-1})$
2. split all states and retrain
3. check utility of splits
4. remerge if split not beneficial
5. back to 2 — unless converged

Excursion: Berkeley Parser

State splitting

assume n states

- replace each production $\text{X-}i \xrightarrow{c} \sigma(\text{Y-}j, \text{Z-}\ell)$ by
  1. $\text{X-}i \xrightarrow{c_1} \sigma(\text{Y-}j, \text{Z-}\ell)$
  2. $\text{X-}i \xrightarrow{c_2} \sigma(\text{Y-}j, \text{Z-}(n+\ell))$
  3. $\text{X-}i \xrightarrow{c_3} \sigma(\text{Y-}(n+j), \text{Z-}\ell)$
  4. $\text{X-}i \xrightarrow{c_4} \sigma(\text{Y-}(n+j), \text{Z-}(n+\ell))$
  5. $\text{X-}(n+i) \xrightarrow{c_5} \sigma(\text{Y-}j, \text{Z-}\ell)$
  6. $\text{X-}(n+i) \xrightarrow{c_6} \sigma(\text{Y-}j, \text{Z-}(n+\ell))$
  7. $\text{X-}(n+i) \xrightarrow{c_7} \sigma(\text{Y-}(n+j), \text{Z-}\ell)$
  8. $\text{X-}(n+i) \xrightarrow{c_8} \sigma(\text{Y-}(n+j), \text{Z-}(n+\ell))$

Excursion: Berkeley Parser

Training the weights

- EM algorithm [Dempster, Laird, Rubin, 1977]
  (Expectation-Maximization)
- we present an inefficient version
  efficient version builds on inside & outside weights

Excursion: Berkeley Parser

Expectation Step

- derivation tree = element of $L(G)$ for the LTG G
  (trees before relabeling)
- for each tree $t \in T$ of the training set T
  build and score all derivation trees d such that $\rho(d) = t$

Given the current model, predict the latent variables

Excursion: Berkeley Parser

Expectation Step

$$S \xrightarrow{0.25} \text{NP-1 VP} \qquad S \xrightarrow{0.75} \text{NP-2 VP}$$

$$\text{VP} \xrightarrow{1} \text{sleeps} \qquad \text{NN} \xrightarrow{1} \text{dragon}$$

$$\text{NP-1} \xrightarrow{1} \text{DT-1 NN} \qquad \text{NP-2} \xrightarrow{1} \text{DT-2 NN}$$

$$\text{DT-1} \xrightarrow{0.9} \text{the} \qquad \text{DT-2} \xrightarrow{0.2} \text{the} \qquad \text{DT-2} \xrightarrow{0.8} \text{a}$$

Excursion: Berkeley Parser

Expectation Step

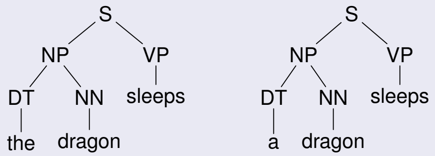
$$S \xrightarrow{0.25} \text{NP-1 VP} \qquad S \xrightarrow{0.75} \text{NP-2 VP}$$

$$\text{VP} \xrightarrow{1} \text{sleeps} \qquad \text{NN} \xrightarrow{1} \text{dragon}$$

$$\text{NP-1} \xrightarrow{1} \text{DT-1 NN} \qquad \text{NP-2} \xrightarrow{1} \text{DT-2 NN}$$

$$\text{DT-1} \xrightarrow{0.9} \text{the} \qquad \text{DT-2} \xrightarrow{0.2} \text{the} \qquad \text{DT-2} \xrightarrow{0.8} \text{a}$$

Training set



Excursion: Berkeley Parser

Expectation Step

$S \xrightarrow{0.25} \text{NP-1 VP}$ $\qquad$ $S \xrightarrow{0.75} \text{NP-2 VP}$

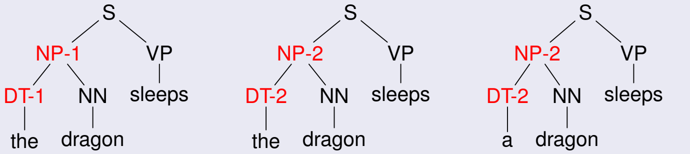
$\text{VP} \xrightarrow{1} \text{sleeps}$ $\qquad$ $\text{NN} \xrightarrow{1} \text{dragon}$

$\text{NP-1} \xrightarrow{1} \text{DT-1 NN}$ $\qquad$ $\text{NP-2} \xrightarrow{1} \text{DT-2 NN}$

$\text{DT-1} \xrightarrow{0.9} \text{the}$ $\qquad$ $\text{DT-2} \xrightarrow{0.2} \text{the}$ $\qquad$ $\text{DT-2} \xrightarrow{0.8} \text{a}$

Derivation trees

weight: $0.25 \cdot 0.9$ $\qquad$ weight: $0.75 \cdot 0.2$ $\qquad$ weight: $0.75 \cdot 0.8$



Excursion: Berkeley Parser

Maximization Step

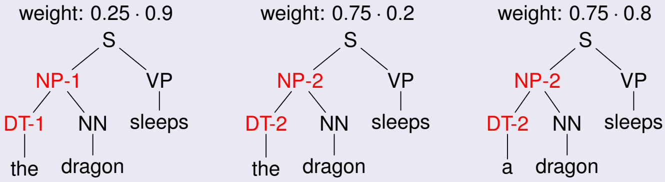
- weighted count $c(\rho)$ of occurrences of each production ρ (each occurrence weighted by derivation weight)
- reset production weights

$$\text{wt}'(\text{X-}i \longrightarrow \text{Y-}j\ \text{Z-}\ell) = \frac{c(\text{X-}i \longrightarrow \text{Y-}j\ \text{Z-}\ell)}{\sum_{j',\ell'} c(\text{X-}i \longrightarrow \text{Y-}j'\ \text{Z-}\ell')}$$

Re-estimate the model parameter given the predictions

Excursion: Berkeley Parser

Derivation trees

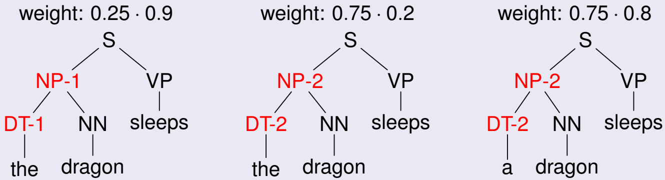


Maximization Step

- $c(\text{S} \longrightarrow \text{NP-1 VP}) = 0.25 \cdot 0.9 = 0.225$
- $c(\text{S} \longrightarrow \text{NP-2 VP}) = 0.75 \cdot 0.2 + 0.75 \cdot 0.8 = 0.75$

Excursion: Berkeley Parser

Derivation trees



Maximization Step

- $c(\mathrm{S} \longrightarrow \mathrm{NP\text{-}1\ VP}) = 0.25 \cdot 0.9 = 0.225$
- $c(\mathrm{S} \longrightarrow \mathrm{NP\text{-}2\ VP}) = 0.75 \cdot 0.2 + 0.75 \cdot 0.8 = 0.75$
- $\mathrm{wt}'(\mathrm{S} \longrightarrow \mathrm{NP\text{-}1\ VP}) = \frac{0.225}{0.225+0.75} = 0.23$
- $\mathrm{wt}'(\mathrm{S} \longrightarrow \mathrm{NP\text{-}2\ VP}) = \frac{0.75}{0.225+0.75} = 0.77$

Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions

   $\mathrm{S} \xrightarrow{c} \mathrm{NP\ VP}$ corresponds to $\mathrm{S\text{-}1} \xrightarrow{c} \mathrm{S(NP\text{-}1,\ VP\text{-}1)}$
2. split all states and retrain
3. check utility of splits

   (evaluate on development test)
4. remerge if split not beneficial
5. back to 2 (unless converged)

Excursion: Berkeley Parser

Training Approach

1. extract wCFG productions

   $\mathrm{S} \xrightarrow{c} \mathrm{NP\ VP}$ corresponds to $\mathrm{S\text{-}1} \xrightarrow{c} \mathrm{S(NP\text{-}1,\ VP\text{-}1)}$

2. split all states and retrain
3. check utility of splits

   (evaluate on development test)

4. remerge if split not beneficial
5. back to 2 (unless converged)

Excursion: Berkeley Parser

Determiner splits:

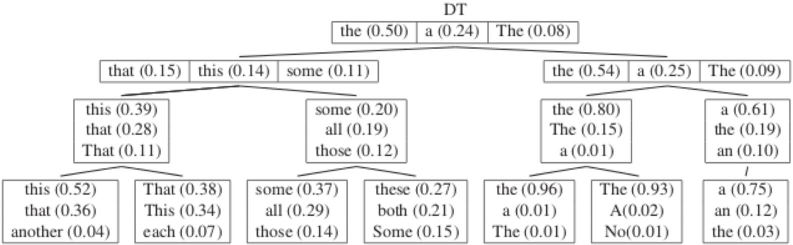


Figure taken from [PETROV et al, 2006]

Excursion: Berkeley Parser

VBZ

VBZ-0	gives	sells	takes
VBZ-1	comes	goes	works
VBZ-2	includes	owns	is
VBZ-3	puts	provides	takes
VBZ-4	says	adds	Says
VBZ-5	believes	means	thinks
VBZ-6	expects	makes	calls
VBZ-7	plans	expects	wants
VBZ-8	is	's	gets
VBZ-9	's	is	remains
VBZ-10	has	's	is
VBZ-11	does	Is	Does

NNP

NNP-0	Jr.	Goldman	INC.
NNP-1	Bush	Noriega	Peters
NNP-2	J.	E.	L.
NNP-3	York	Francisco	Street
NNP-4	Inc	Exchange	Co
NNP-5	Inc.	Corp.	Co.
NNP-6	Stock	Exchange	York
NNP-7	Corp.	Inc.	Group
NNP-8	Congress	Japan	IBM
NNP-9	Friday	September	August
NNP-10	Shearson	D.	Ford
NNP-11	U.S.	Treasury	Senate
NNP-12	John	Robert	James
NNP-13	Mr.	Ms.	President
NNP-14	Oct.	Nov.	Sept.
NNP-15	New	San	Wall

JJS

JJS-0	largest	latest	biggest
JJS-1	least	best	worst
JJS-2	most	Most	least

DT

DT-0	the	The	a
DT-1	A	An	Another
DT-2	The	No	This
DT-3	The	Some	These
DT-4	all	those	some
DT-5	some	these	both
DT-6	That	This	each
DT-7	this	that	each
DT-8	the	The	a
DT-9	no	any	some
DT-10	an	a	the
DT-11	a	this	the

CD

CD-0	1	50	100
CD-1	8.50	15	1.2
CD-2	8	10	20
CD-3	1	30	31
CD-4	1989	1990	1988
CD-5	1988	1987	1990
CD-6	two	three	five
CD-7	one	One	Three
CD-8	12	34	14
CD-9	78	58	34
CD-10	one	two	three
CD-11	million	billion	trillion

PRP

PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

RBR

RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

IN

IN-0	In	With	After
IN-1	In	For	At
IN-2	in	for	on
IN-3	of	for	on
IN-4	from	on	with
IN-5	at	for	by
IN-6	by	in	with
IN-7	for	with	on
IN-8	If	While	As
IN-9	because	if	while
IN-10	whether	if	That
IN-11	that	like	whether
IN-12	about	over	between
IN-13	as	de	Up
IN-14	than	ago	until
IN-15	out	up	down

RB

RB-0	recently	previously	still
RB-1	here	back	now
RB-2	very	highly	relatively
RB-3	so	too	as
RB-4	also	now	still
RB-5	however	Now	However
RB-6	much	far	enough
RB-7	even	well	then
RB-8	as	about	nearly
RB-9	only	just	almost
RB-10	ago	earlier	later
RB-11	rather	instead	because
RB-12	back	close	ahead
RB-13	up	down	off
RB-14	not	Not	maybe
RB-15	n't	not	also

Figure taken from [PETROV et al, 2006]

Parsing

Summary

- good source of (relevant) weighted tree automata
- large automata
  - English: 153 MB (1,133 states and 4,267,277 productions)
  - Chinese: 98 MB
- many operations still to be investigated
  (determinization, minimization, products, etc.)

Parsing

Summary

- good source of (relevant) weighted tree automata
- large automata
  - English: 153 MB (1,133 states and 4,267,277 productions)
  - Chinese: 98 MB
- many operations still to be investigated
  (determinization, minimization, products, etc.)

(Nondeterministic) Minimization

- unweighted: PSpace (NFA) and ExpTime (TA)
- weighted over field:
  PTime (wNFA) and randomized PTime (wTA)

Literature

Articles

- KLEIN, MANNING
  Accurate Unlexicalized Parsing Proc. 41st ACL, 2003
- PETROV, BARRETT, THIBAUX AND KLEIN
  Learning Accurate, Compact, and Interpretable Tree Annotation Proc. 44th ACL, 2006
- SHINDO, MIYAO, FUJINO AND NAGATA
  Bayesian Symbol-Refined Tree Substitution Grammars for Syntactic Parsing Proc. 50th ACL, 2012