Collins Parser

(Collins 1997, 2005)

What is a supervised parser? When is it lexicalized? How are dependencies used for CFG parsing? What is a generative model? Why discriminative reranking? How is it evaluated? How good are the results?

Outline

- basics
  - (P)CFG
  - supervised learning
  - (lexicalized) PCFG
- Collins 1997: Probabilistic parser
  - model 1: generative version of (Collins 1996)
  - model 2: + complement/adjunct distinction
  - (model 3: + wh-movement model)
- Collins 2005: Reranking
  - reranker architecture
  - generative / discriminative, (log-)linear
- conclusion

Probabilistic CFG

- **CFG**

  $S \rightarrow NP \quad VP$

- **PCFG**

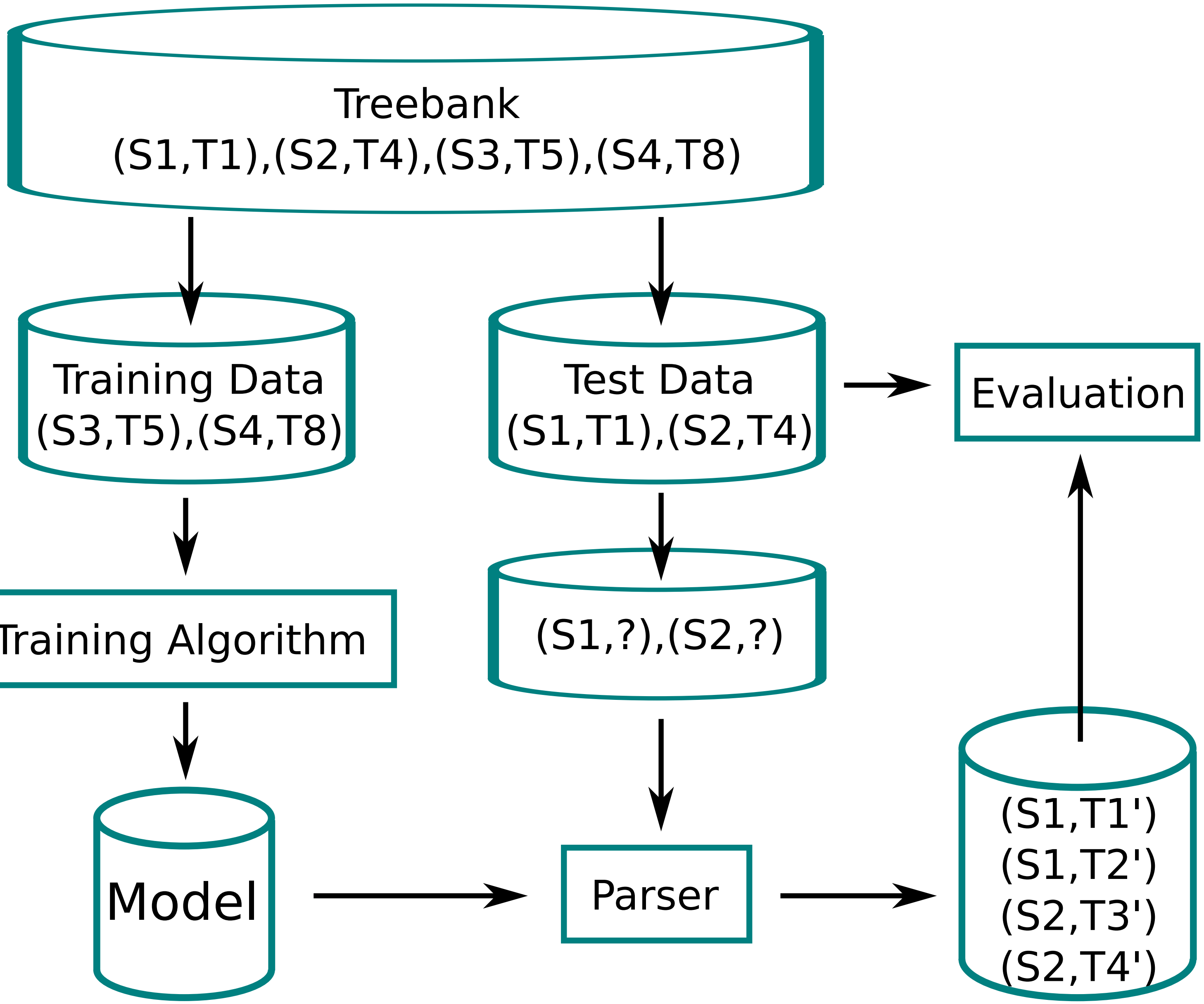
  $S \rightarrow NP \quad VP$ (90%)

  which means:

  - $\mathcal{P}(Rule_r = \langle NP, VP \rangle \mid Rule_l = S) = 0.9$
  - with normalization:

    $\sum_{rule_r} \mathcal{P}(rule_r | rule_l) = 1$

Supervised Parsing Architecture



Finding the Best Parse

$$T_{best} = \arg\max_T \mathcal{P}(T|S) = \arg\max_T \frac{\mathcal{P}(T,S)}{\mathcal{P}(S)} = \arg\max_T \mathcal{P}(T,S)$$

Two types of models

- **discriminative**:
  - $\mathcal{P}(T|S)$ estimated directly
  - $\mathcal{P}(T,S)$ distribution not available
  - no model parameters for generating S
- **generative**:
  - estimation of $\mathcal{P}(T,S)$
  - PCFG: $\mathcal{P}(T,S) = \prod_{rule \in S} \mathcal{P}(rule_r | rule_l)$

Lexicalization of Rules

add **head word** and its **PoS** tag to each nonterminal

$$S \rightarrow NP \quad VP$$

becomes

$$S(\mathbf{loves}, \mathbf{VB}) \rightarrow NP(John, NNP) \quad VP(\mathbf{loves}, \mathbf{VB})$$

let's write this as

$$P(\mathbf{h}) \rightarrow L_1(l_1) \quad H(\mathbf{h})$$

Collins 1997: Model 1

Tell a **head-driven** (lexicalized) generative story:

$$\mathcal{P}(rule_r|rule_l) = \mathcal{P}(L_n(l_n), \ldots, L_1(l_1), \mathbf{H}(\mathbf{h}), R_1(r_1), \ldots, R_m(r_m) \,|\, P(\mathbf{h}))$$

- generate heads first,

  then the left and right modifiers (independently)

$$= \mathcal{P}(\mathbf{H}(\mathbf{h}) \,|\, P(\mathbf{h})) \quad \cdot \quad \prod_{i=1}^{n+1} \mathcal{P}(L_i(l_i) \,|\, P(\mathbf{h}), \mathbf{H}(\mathbf{h}), \vec{\Delta}(i))$$
$$\cdot \quad \prod_{i=1}^{m+1} \mathcal{P}(R_i(r_i) \,|\, P(\mathbf{h}), \mathbf{H}(\mathbf{h}), \vec{\Delta}(i))$$

- stop generating modifiers when

  $L_{n+1}(l_{n+1}) = \mathsf{STOP}$ or

  $R_{m+1}(r_{m+1}) = \mathsf{STOP}$

- $\vec{\Delta} : \mathbb{N} \rightarrow \langle \text{neighbour?, verb in between?, } (0, 1, 2, > 2) \text{ commas in between?} \rangle$

Parameter Estimation

$$= \mathcal{P}(H(h) \mid P(h)) \quad \cdot \quad \prod_{i=1}^{n+1} \mathcal{P}(L_i(l_i) \mid P(h), H(h), \vec{\Delta}(i))$$

$$\cdot \quad \prod_{i=1}^{m+1} \mathcal{P}(R_i(r_i) \mid P(h), H(h), \vec{\Delta}(i))$$

parameters estimated by **relative frequency** in the training set (max. likelihood):

$$\mathcal{P}(H(h)|P(h)) = \frac{C(H(h), P(h))}{C(P(h))}$$

$$\mathcal{P}(L_i(l_i)|P(h), H(h), \vec{\Delta}(i)) = \frac{C(L_i(l_i), P(h), H(h), \vec{\Delta}(i))}{C(P(h), H(h), \vec{\Delta}(i))}$$

linearly smoothed with counts with less specific conditions (backoff)

Parsing

- Bottom-Up chart parsing
- PoS tag sentence
- each word is a potential head of a phrase
- calculate probabilities of modifiers
- go on

Dataset

Penn Treebank: Wall Street Journal portion

- sections 2-21 for training
  - 40k sentences
- section 23 for testing
  - 2,416 sentences

Evaluation

PARSEVAL evaluation measures:

$$\text{Labeled Precision (LP)} = \frac{\text{nr of } \textbf{correctly} \text{ predicted constituents}}{\text{nr of } \textbf{all predicted} \text{ constituents}}$$

$$\text{Labeled Recall (LR)} = \frac{\text{nr of } \textbf{correctly} \text{ predicted constituents}}{\text{nr of } \textbf{all correct} \text{ constituents in the gold parse}}$$

where '*correct*' constituent $\leftrightarrow$ **same boundaries, same label**

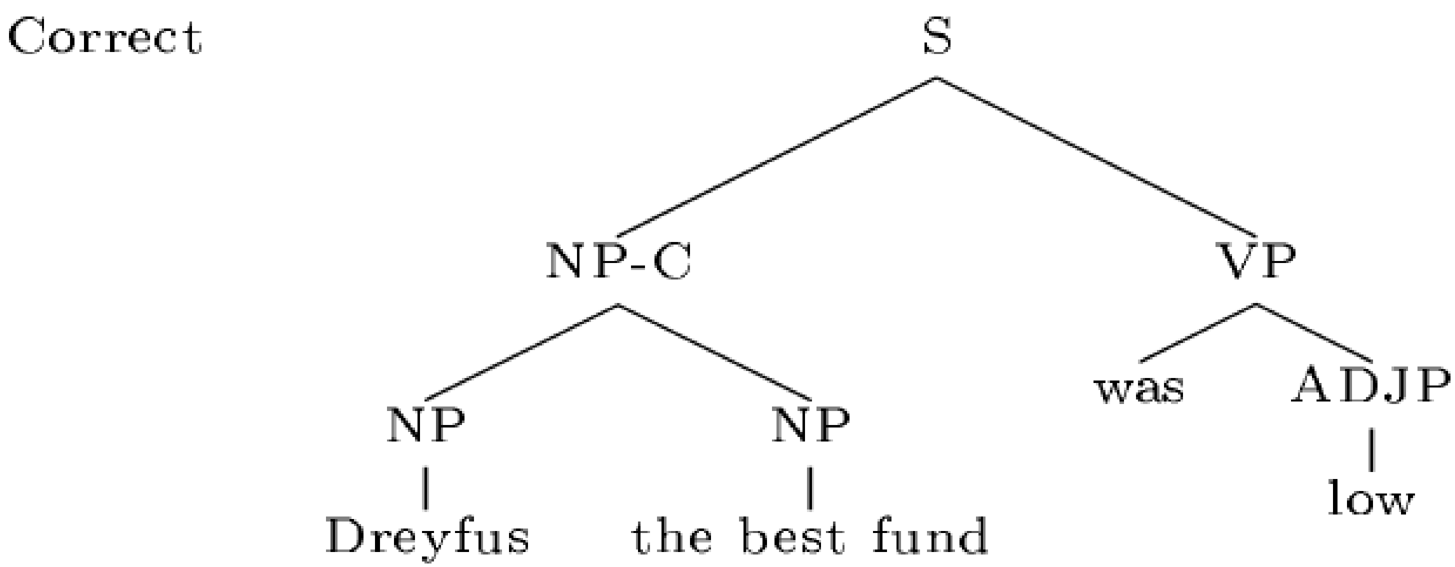
$\text{Crossing Brackets (CB)} =$ nr of constituents violating the boundaries in the gold parse

Results Model 1

MODEL	$\leq$ 40 Words (2245 sentences)					$\leq$ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	$\leq$ 2 CBs	LR	LP	CBs	0 CBs	$\leq$ 2 CBs
(Magerman 95)	84.6%	84.9%	1.26	56.6%	81.4%	84.0%	84.3%	1.46	54.0%	78.8%
(Collins 96)	85.8%	86.3%	1.14	59.9%	83.6%	85.3%	85.7%	1.32	57.2%	80.8%
Model 1	87.4%	88.1%	0.96	65.7%	86.3%	86.8%	87.6%	1.11	63.1%	84.1%

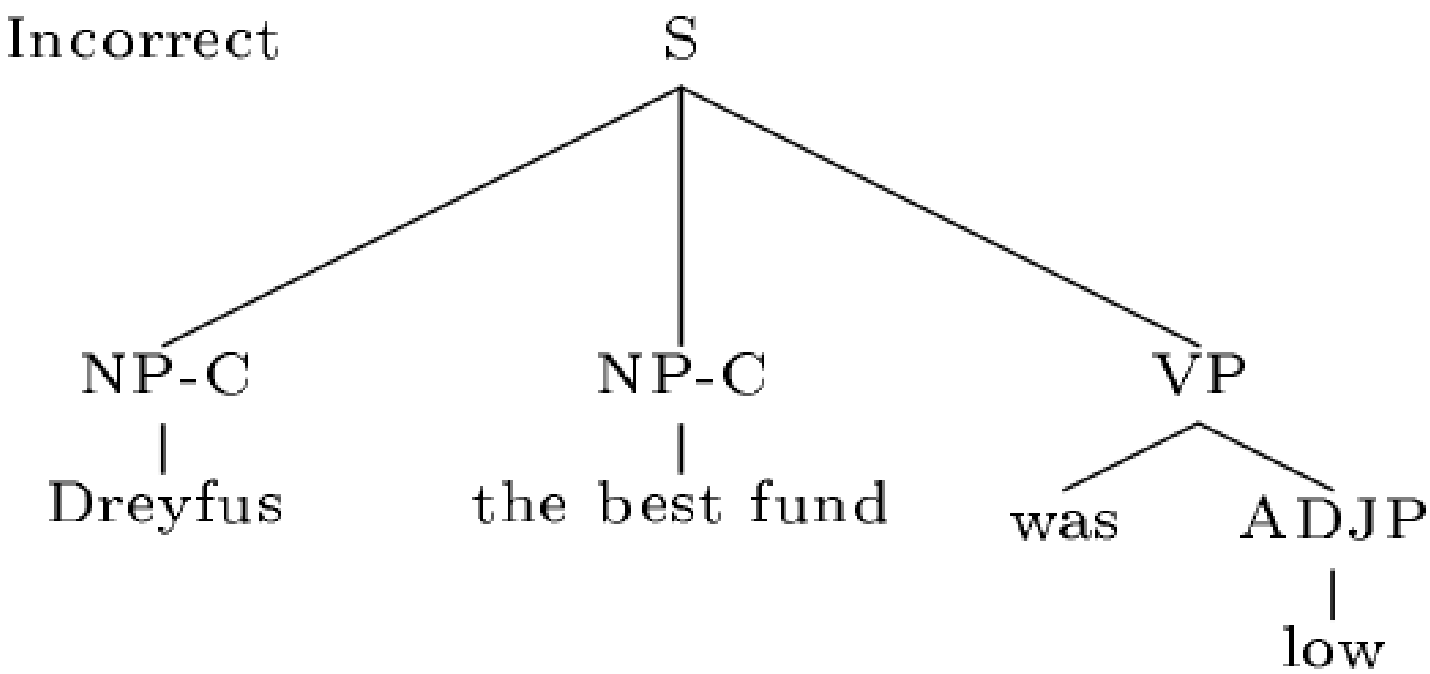
Subcategorization Problem

consider this parse:



Subcategorization Problem

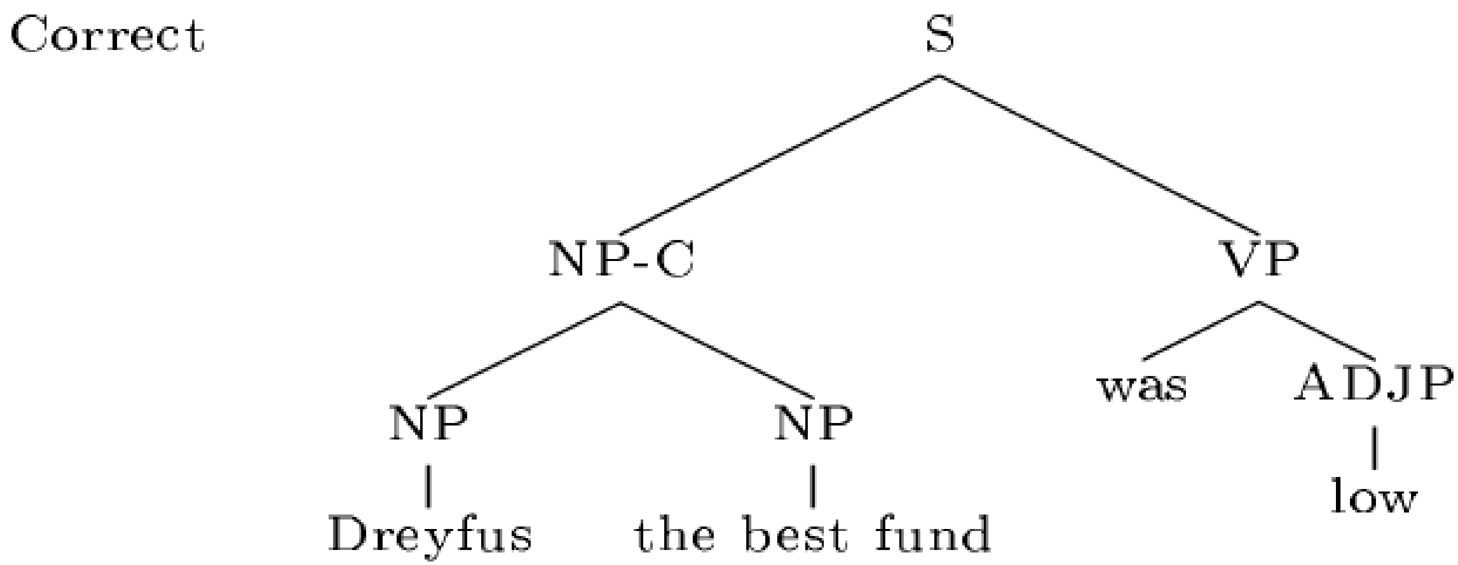
due to the independence of modifiers, Model 1 may parse:



Subcategorization Problem

Solution: distinguish modifiers into **complements** ('-C') and **adjuncts**

$\rightarrow$ estimate separate probabilities



$\rightarrow$ learn that $VP(\text{was})$ prefers only one complement.

complement information might also help identifying functional information like *subject*

Model 2

Extend Model 1:

$$\mathcal{P}(H(h)|P(h)) \cdot \mathcal{P}(LC|P(h),H(h)) \cdot \mathcal{P}(RC|P(h),H(h))$$

$$\cdot \prod_{i=1}^{m+1} \mathcal{P}(L_i(l_i)|P(h),H(h),\vec{\Delta}(i),\widetilde{LC_i})$$

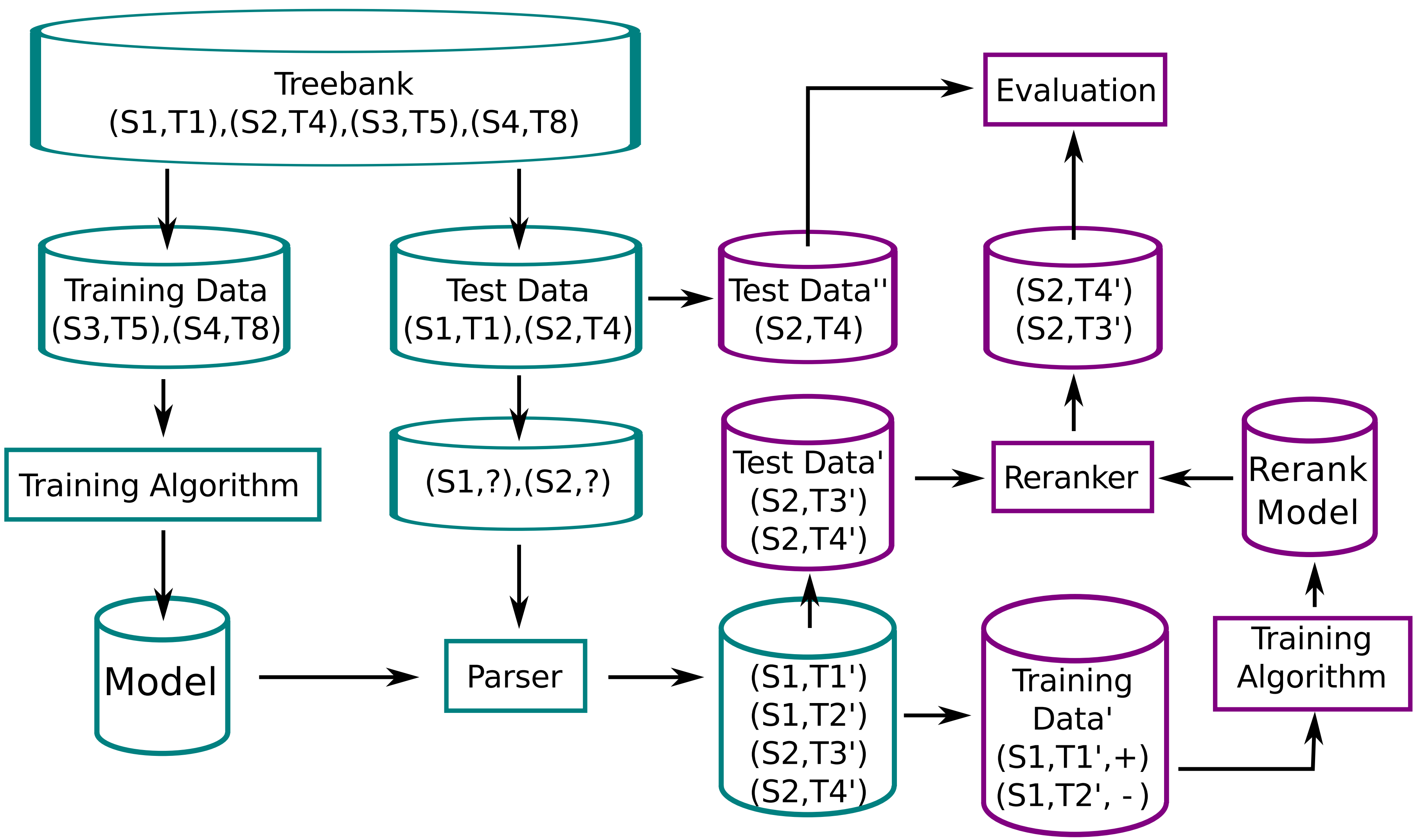
$$\cdot \prod_{i=1}^{n+1} \mathcal{P}(R_i(r_i)|P(h),H(h),\vec{\Delta}(i),\widetilde{RC_i})$$

- draw sets of allowed complements (subcat sets) for the left (LC) and right (RC) side
- generate each complement in LC/RC exactly once.
- no STOP before the subcat set is satisfied

Results Model 2

MODEL	≤ 40 Words (2245 sentences)					≤ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	≤ 2 CBs	LR	LP	CBs	0 CBs	≤ 2 CBs
(Magerman 95)	84.6%	84.9%	1.26	56.6%	81.4%	84.0%	84.3%	1.46	54.0%	78.8%
(Collins 96)	85.8%	86.3%	1.14	59.9%	83.6%	85.3%	85.7%	1.32	57.2%	80.8%
Model 1	87.4%	88.1%	0.96	65.7%	86.3%	86.8%	87.6%	1.11	63.1%	84.1%
Model 2	88.1%	88.6%	0.91	66.5%	86.9%	87.5%	88.1%	1.07	63.9%	84.6%

Reranking (Collins 2005) Architecture



UNIVERSITÄT DES SAARLANDES

Why rank again?

- consider **more features** of a parse tree
  - CFG rule occurrence (lexicalized / with grandparent node)
  - bigram (nonterminals only / lexicalized) occurrence
  - . . .
- **parser: generative model**
  - new random variables needed for every feature
    $\longrightarrow$ nr of joint-probability parameters grows exponentially with nr of features
    (must be avoided by a generative story introducing conditional independencies)
- **reranker: discriminative** (log-)linear classifier
  - treat every feature independently
  - simple to extend feature set

Log-Linear Models

for PCFG, one step is an application of a CFG-rule:

$$\begin{aligned}\mathcal{P}(T,S) &= \prod_{rule \in S} \mathcal{P}(rule_r | rule_l) \\ &= \prod_{rule \in \mathcal{G}} \mathcal{P}(rule_r | rule_l)^{C_S(rule)} \\ \Leftrightarrow log(\mathcal{P}(T,S)) &= \sum_{rule \in \mathcal{G}} log(\mathcal{P}(rule_r | rule_l)) \cdot C_S(rule)\end{aligned}$$

i.e. linear combination in log space

call $log(\mathcal{P}(rule_r|rule_l))$ '*feature weight*' and $C(rule)$ '*feature value*'

UNIVERSITÄT DES SAARLANDES

Results after Reranking

Model	$\leq$ 40 Words (2,245 sentences)				
	LR	LP	CBs	0 CBs	2 CBs
Charniak 1997	87.5%	87.4%	1.00	62.1%	86.1%
Collins 1999	88.5%	88.7%	0.92	66.7%	87.1%
Charniak 2000	90.1%	90.1%	0.74	70.1%	89.6%
ExpLoss	90.2%	90.4%	0.73	71.2%	90.2%

Model	$\leq$ 100 Words (2,416 sentences)				
	LR	LP	CBs	0 CBs	2 CBs
Charniak 1997	86.7%	86.6%	1.20	59.5%	83.2%
Ratnaparkhi 1998	86.3%	87.5%	1.21	60.2%	—
Collins 1999	88.1%	88.3%	1.06	64.0%	85.1%
Charniak 2000	89.6%	89.5%	0.88	67.6%	87.7%
ExpLoss	89.6%	89.9%	0.86	68.7%	88.3%

Conclusion

- Lexicalized parser
- ’Head-centric’ generative process
- Extensions for subcategorization (and wh-movement)
- Discriminative Reranking of results

Thanks for your attention!

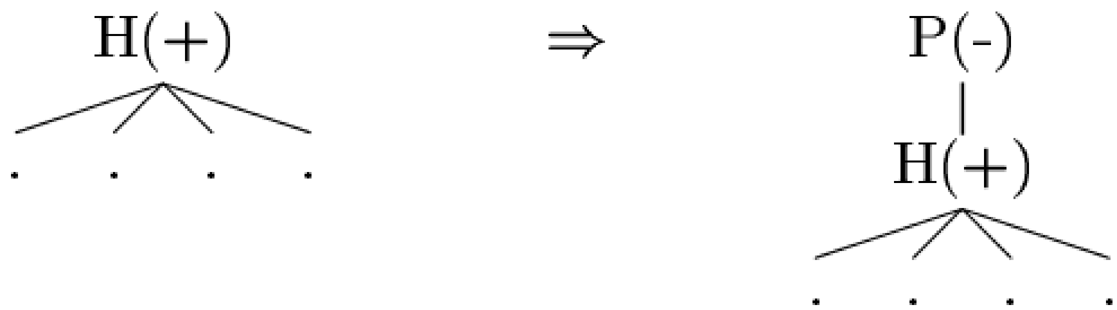
questions

discussion

Parsing 1/3

bottom up chart parsing:

choose a complete(+) phrase as head for a new phrase

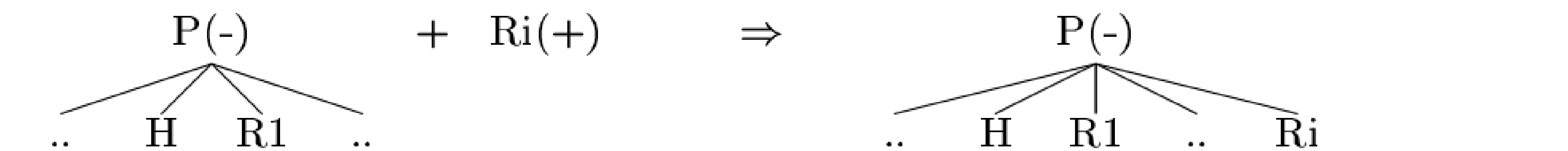


$$\text{Prob} = \text{X}$$

$$\text{Prob} = \text{X} \times \mathcal{P}_H(H \mid P, \ldots)$$

Parsing 2/3

add completed neighbouring phrases as modifiers



$\text{Prob} = \text{X}$ $\qquad$ $\text{Prob} = \text{Y}$ $\qquad$ $\text{Prob} = \text{X} \times \text{Y} \times \mathcal{P}_R(R_i(r_i) \mid P, H, \ldots)$

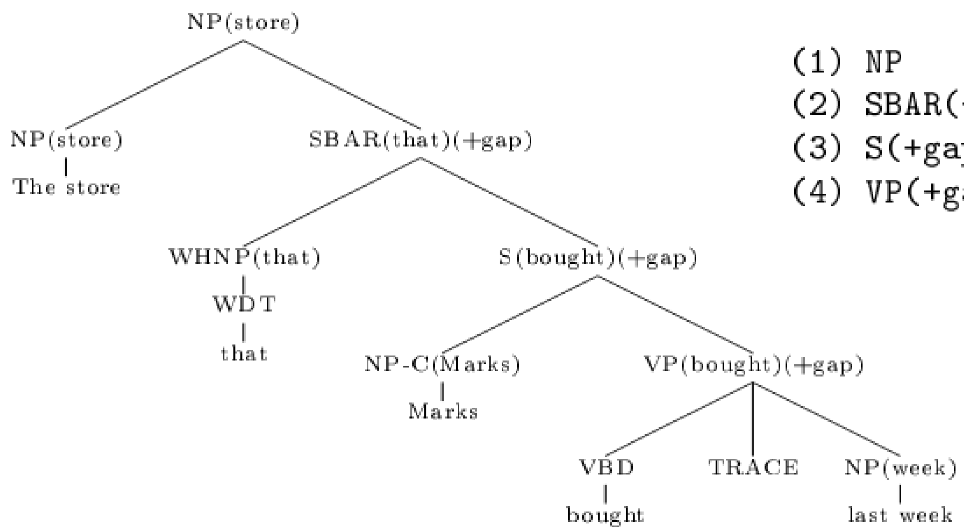
Parsing 3/3

complete by adding STOP modifiers

$$\text{Prob} = \text{X} \qquad\qquad \text{Prob} = \text{X} \times \mathcal{P}_L(STOP \mid ....) \times \mathcal{P}_R(STOP \mid ....)$$

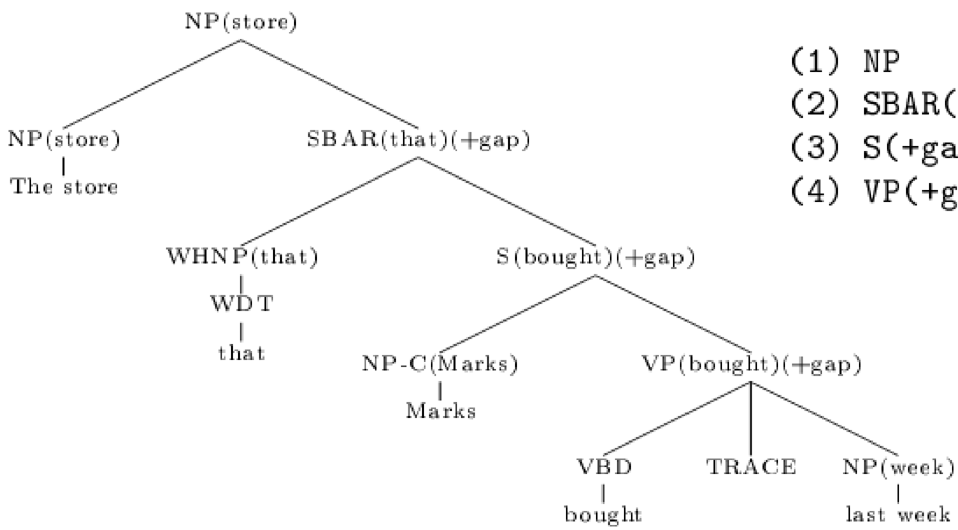
wh-Movement Rules

Solution: Account for (+gap) rules separately. $\longrightarrow$ Allow generation of a TRACE under a (+gap)-version of a nonterminal.




```
(1) NP          ->  NP     SBAR(+gap)
(2) SBAR(+gap)  ->  WHNP   S-C(+gap)
(3) S(+gap)     ->  NP-C   VP(+gap)
(4) VP(+gap)    ->  VB     TRACE NP
```

wh-Movement Rule Analysis




```
(1) NP          ->  NP    SBAR(+gap)
(2) SBAR(+gap)  ->  WHNP  S-C(+gap)
(3) S(+gap)     ->  NP-C  VP(+gap)
(4) VP(+gap)    ->  VB    TRACE NP
```

we observe: A TRACE can be

- passed down the head (rule 3)
- passed down to one of the left / right modifiers
- discharged by a TRACE

Model 3

Extend Model 2: new random variable G with values:

- Head - passed down the head (3)
- Left/Right - passed down to one of the left / right modifiers (LC+=*gap* / RC+=*gap*)

the *gap* entry in LC/RC is discharged by a TRACE or a (gap)-phrase modifier phrase

$$\mathcal{P}(H(h)|P(h)) \cdot \mathcal{P}(LC|P(h), H(h)) \cdot \mathcal{P}(RC|P(h), H(h)) \cdot \mathcal{P}(G|P(h), H(h))$$

$$\cdot \prod_{i=1}^{m+1} \mathcal{P}(L_i(l_i)|P(h), H(h), \vec{\Delta}(i), \widetilde{LC_i})$$

$$\cdot \prod_{i=1}^{n+1} \mathcal{P}(R_i(r_i)|P(h), H(h), \vec{\Delta}(i), \widetilde{RC_i})$$

Results Model 3

MODEL	≤ 40 Words (2245 sentences)					≤ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	≤ 2 CBs	LR	LP	CBs	0 CBs	≤ 2 CBs
(Magerman 95)	84.6%	84.9%	1.26	56.6%	81.4%	84.0%	84.3%	1.46	54.0%	78.8%
(Collins 96)	85.8%	86.3%	1.14	59.9%	83.6%	85.3%	85.7%	1.32	57.2%	80.8%
Model 1	87.4%	88.1%	0.96	65.7%	86.3%	86.8%	87.6%	1.11	63.1%	84.1%
Model 2	88.1%	88.6%	0.91	66.5%	86.9%	87.5%	88.1%	1.07	63.9%	84.6%
Model 3	88.1%	88.6%	0.91	66.4%	86.9%	87.5%	88.1%	1.07	63.9%	84.6%

Practical Issues - Smoothing

sparse data for full conditioning set $\rightarrow$ needs backoff

Back-off Level	$\mathcal{P}_H(H \mid ...)$	$\mathcal{P}_G(G \mid ...)$ $\mathcal{P}_{LC}(LC \mid ...)$ $\mathcal{P}_{RC}(RC \mid ...)$	$\mathcal{P}_{L1}(L_i(lt_i) \mid ...)$ $\mathcal{P}_{R1}(R_i(rt_i) \mid ...)$	$\mathcal{P}_{L2}(lw_i \mid ...)$ $\mathcal{P}_{R2}(rw_i \mid ...)$
1	P, w, t	P, H, w, t	P, H, w, t, Δ, LC	L_i, lt_i, P, H, w, t, Δ, LC
2	P, t	P, H, t	P, H, t, Δ, LC	L_i, lt_i, P, H, t, Δ, LC
3	P	P, H	P, H, Δ, LC	L_i, lt_i
4	—	—	—	lt_i

linear combination: $\hat{p} = \lambda \cdot \hat{p_{mle}} + (1 - \lambda) \cdot \hat{p}_{backoff}$

recursively stacked: $\hat{p}_{backoff} = \lambda' \cdot \hat{p}_{mle'} + (1 - \lambda') \cdot \hat{p}_{backoff'}$

all words occurring less than 5 times are replaced by UNKNOWN

History-Based Models

history-based model (generative, structured):

$\mathcal{P}(T, S) = \prod_{i=1}^{n} \mathcal{P}(d_i | \Phi(d_1, \ldots, d_{i-1}))$

i.e. a pair (t, s) is generated by a sequence of steps $D = \langle d_1, \ldots, d_n \rangle$

Boosting

- machine-learning algorithm
- composition of (typically) simple classifiers
- repeatedly add a new classifier which is trained with particular focus on the samples that are incorrectly classified by the previous zoo of classifiers

Here:

- each simple classifier has exactly one binary feature
- learning finds the feature that helps the most to improve the results of the previous classifier zoo