Features of Statistical Parsers

Preliminary results

Mark Johnson

Brown University

TTI, October 2003

Joint work with Michael Collins (MIT)

Supported by NSF grants LIS 9720368 and IIS0095940

Talk outline

- Statistical parsing from PCFGs to discriminative models
- Linear discriminative models
  - conditional estimation and log loss
  - over-learning and regularization
- Feature design
  - Local and non-local features
  - Feature design
- Conclusions and future work

Why adopt a statistical approach?

- The interpretation of a sentence is:
  - *hidden*, i.e., not straight-forwardly determined by its words or sounds
  - *dependent on many interacting factors*, including grammar, structural preferences, pragmatics, context and general world knowledge.
  - *pervasively ambiguous* even when all known linguistic and cognitive constraints are applied
- *Statistics is the study of inference under uncertainty*
  - Statistical methods provide a systematic way of integrating weak or uncertain information

The dilemma of non-statistical CL

1. *Ambiguity explodes combinatorially*

   (162) *Even though it's possible to scan using the Auto Image Enhance mode, it's best to use the normal scan mode to scan your documents.*

   - Refining the grammar is usually self-defeating
     $\Rightarrow$ splits states $\Rightarrow$ makes ambiguity worse!
   - Preference information guides parser to correct analysis

2. *Linguistic well-formedness leads to non-robustness*

   - Perfectly comprehensible sentences receive no parses ...

Conventional approaches to robustness

- Some ungrammatical sentences are perfectly comprehensible e.g., *He walk*
  - Ignoring agreement ⇒ spurious ambiguity
    I saw the father of the children that speak(s) French
- Extra-grammatical rules, repair mechanisms, . . .
  - How can semantic interpretation take place without a well-formed syntactic analysis?
- A preference-based approach can provide a systematic treatment of robustness too!

Linguistics and statistical parsing

- *Statistical parsers are not "linguistics-free"*
  - The corpus contains linguistic information (e.g., the treebank is based on a specific linguistic theory)
  - Linguistic and psycholinguistic insights guide feature design
- *What is the most effective way to import linguistic knowledge into a machine?*
  - *manually specify* possible linguistic structures
    - ∗ by explicit specification (a grammar)
    - ∗ by example (an annotated corpus)
  - *manually specify* the model's features
  - *learn* feature weights from training data

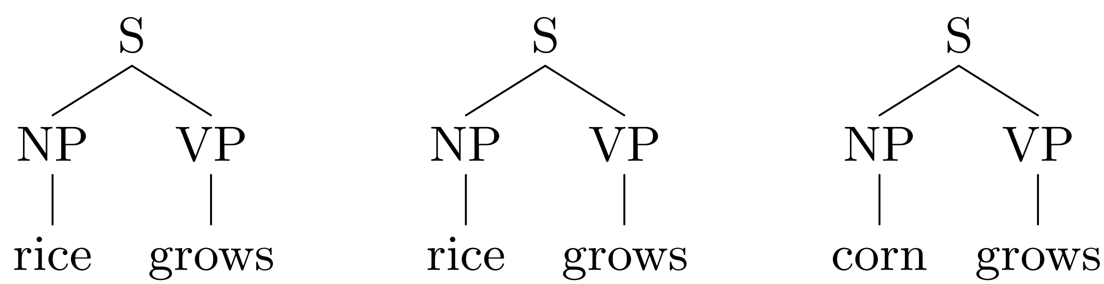
Framework of statistical parsing

- $\mathcal{X}$ is the set of sentences
- $\mathcal{Y}(x)$ is the set of possible linguistic analyses of $x \in \mathcal{X}$
- Preference or score $S_w(x, y)$ for each (x, y) parameterized by weights w
- Parsing a string x involves finding the highest scoring analysis

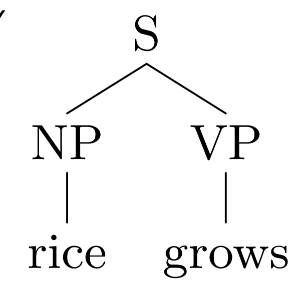
$$\hat{y}(x) = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}} S_w(x, y)$$

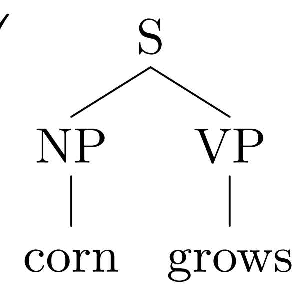
- Learning or training involves identifying w from data

PCFGs and the MLE

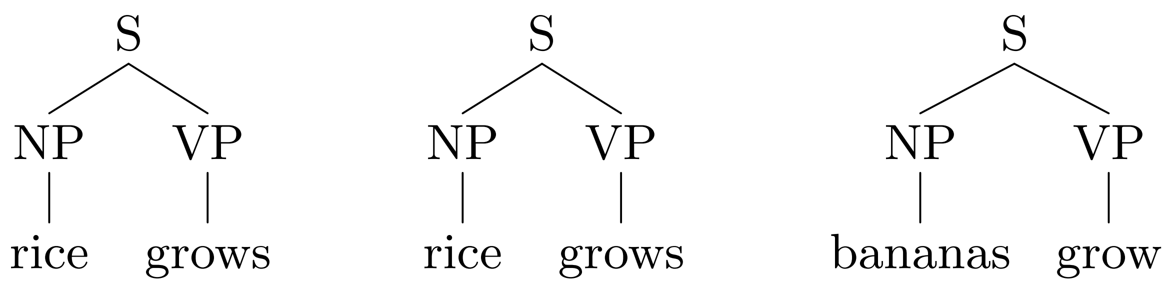


rule	count	rel freq
S $\rightarrow$ NP VP	3	1
NP $\rightarrow$ rice	2	2/3
NP $\rightarrow$ corn	1	1/3
VP $\rightarrow$ grows	3	1

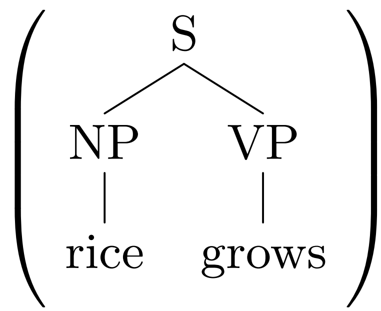
P () $= 2/3$

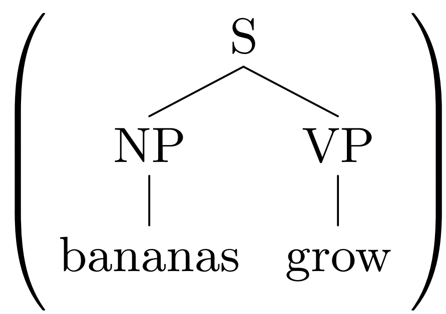
P () $= 1/3$

Non-local constraints



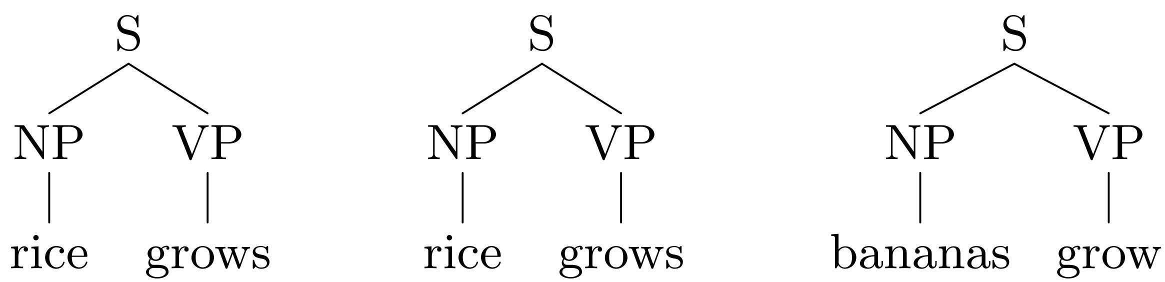
rule	**count**	**rel freq**
S $\to$ NP VP	3	1
NP $\to$ rice	2	2/3
NP $\to$ bananas	1	1/3
VP $\to$ grows	2	2/3
VP $\to$ grow	1	1/3

P  = 4/9

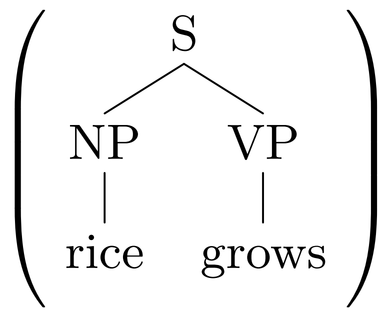
P  = 1/9

Z = 5/9

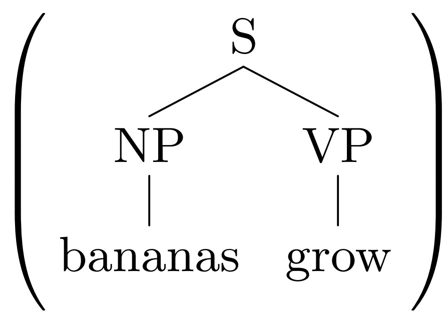
Renormalization



rule	**count**	**rel freq**
S → NP VP	3	1
NP → rice	2	2/3
NP → bananas	1	1/3
VP → grows	2	2/3
VP → grow	1	1/3

P 

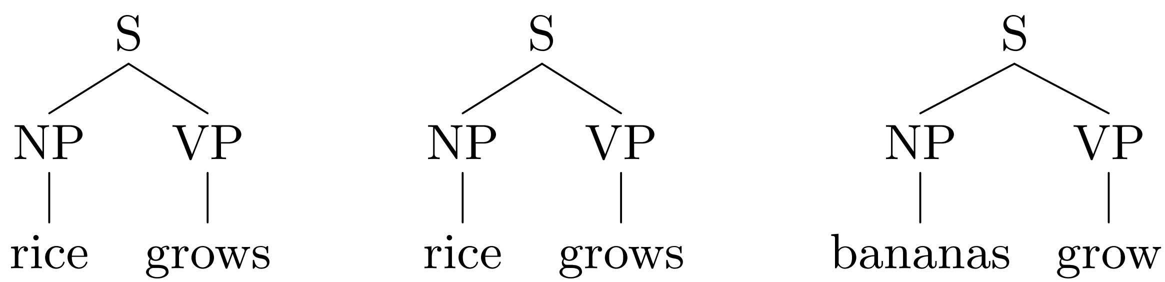
$= 4/9 \quad 4/5$

P 

$= 1/9 \quad 1/5$

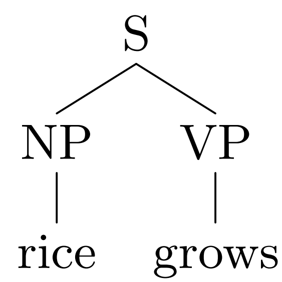
$Z = 5/9$

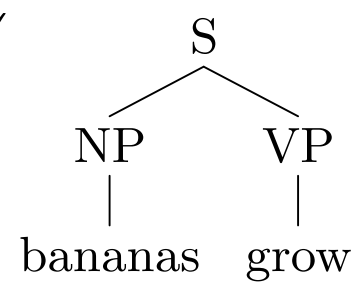
Other values do better!



rule	**count**	**rel freq**
S → NP VP	3	1
NP → rice	2	2/3
NP → bananas	1	1/3
VP → grows	2	1/2
VP → grow	1	1/2

(Abney 1997)

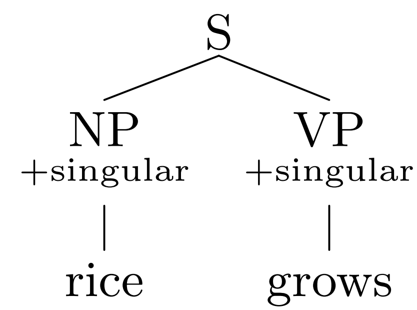
P () = 2/6 2/3

P () = 1/6 1/3

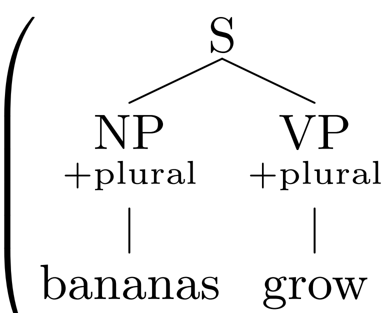
Z = 3/6

Make dependencies local – GPSG-style

rule	**count**	**rel freq**
$S \rightarrow \underset{+\text{singular}}{\text{NP}} \ \underset{+\text{singular}}{\text{VP}}$	2	2/3
$S \rightarrow \underset{+\text{plural}}{\text{NP}} \ \underset{+\text{plural}}{\text{VP}}$	1	1/3
$\underset{+\text{singular}}{\text{NP}} \rightarrow \text{rice}$	2	1
$\underset{+\text{plural}}{\text{NP}} \rightarrow \text{bananas}$	1	1
$\underset{+\text{singular}}{\text{VP}} \rightarrow \text{grows}$	2	1
$\underset{+\text{plural}}{\text{VP}} \rightarrow \text{grow}$	1	1

P (


) = 2/3

P (


) = 1/3

Generative vs. Discriminative models

Generative models: features are context-free

- rules (local trees) are "natural" features
- the MLE of w is easy to compute (in principle)

Discriminative models: features have unknown dependencies

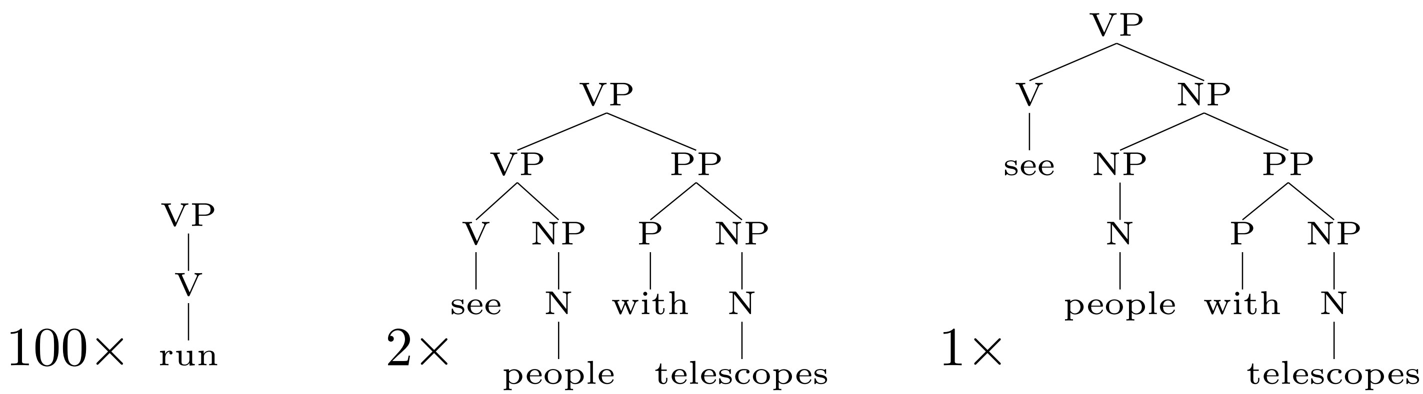
– no "natural" features

– estimating w is much more complicated

\+ features need not be local trees

\+ representations need not be trees

Generative vs Discriminative training



$\ldots \times 2/105 \times \ldots$ $\qquad$ $\ldots \times 1/7 \times \ldots$

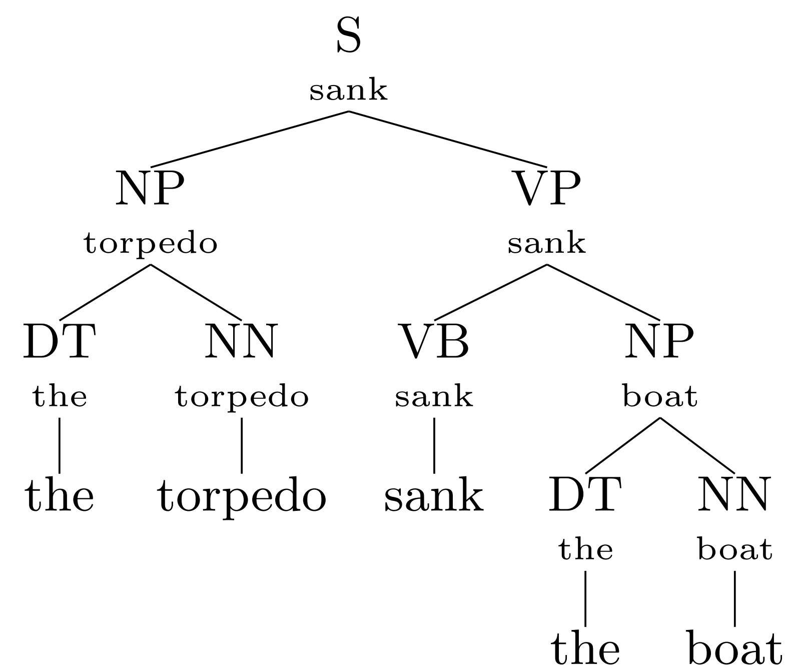
$\ldots \times 2/7 \times \ldots$ $\qquad$ $\ldots \times 1/7 \times \ldots$

Rule	**count**	**rel freq**	**rel freq**
VP → V	100	100/105	4/7
VP → V NP	3	3/105	1/7
VP → VP PP	2	2/105	**2/7**
NP → N	6	6/7	6/7
NP → NP PP	1	1/7	**1/7**

Features in standard generative models

- *Lexicalization* or *head annotation* captures *subcategorization* of lexical items and primitive *world knowledge*
- Trained from Penn treebank corpus ($\approx$ 40,000 trees, 1M words)
- *Sparse data* is the big problem, so *smoothing* or *generalization* is most important!

$$\underset{\text{sank}}{\text{VP}} \rightarrow \underset{\text{sank}}{\text{VB}}\ \underset{\text{boat}}{\text{NP}}$$



Many useful features are non-local

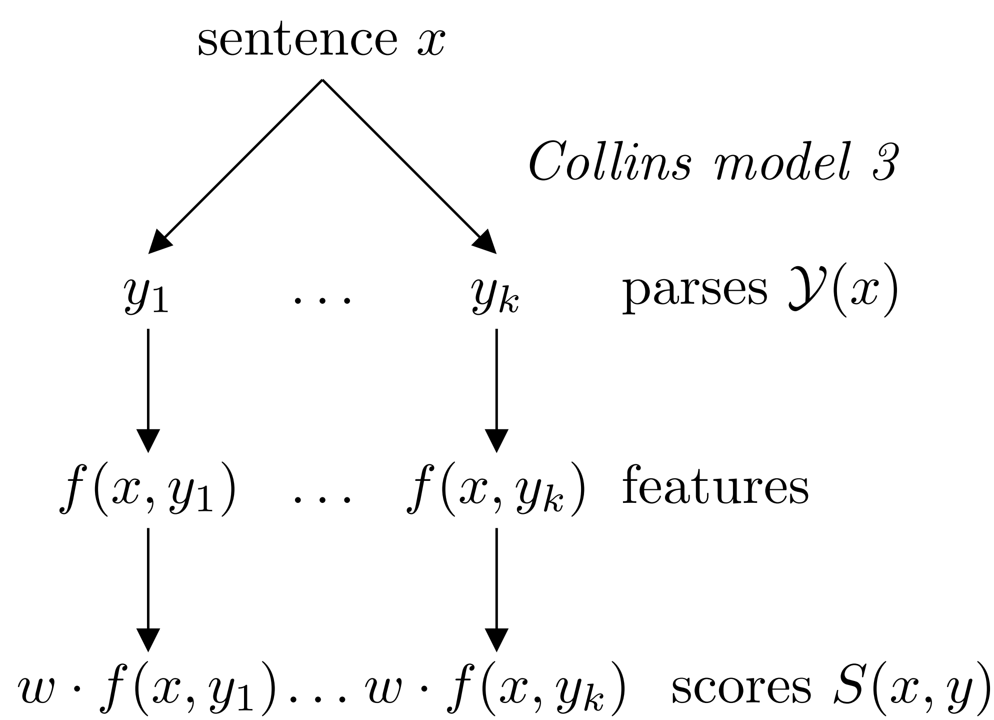
- Many desirable features are difficult to localize (i.e., express in terms of annotation on labels)
  - Verb-particle constructions
    Sam gave chocolates out/up/to Sandy
  - Head-to-head dependencies in coordinate structures
    [[*the students*] *and* [*the professor*]] *ate a pizza*
- Some features seem inherently non-local
  - Heavy constituents prefer to be at the end
    Sam donated to the library a collection ?*(that it took her years to assemble)*
  - Parallelism in coordination
    Sam saw a man with a telescope and a woman with binoculars
    ?*Sam [saw a man with a telescope and a woman] with binoculars*

Framework for discriminative parsing

- Generate *candidate parses* $\mathcal{Y}(x)$ for each sentence x
- Each parse $y \in \mathcal{Y}(x)$ is mapped to a feature vector $f(x, y)$
- Each feature f_j is associated with a weight w_j
- Define $S(x, y) = w \cdot f(x, y)$
- The highest scoring parse

$$\hat{y} = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}} S(x, y)$$

is predicted correct



Log linear models

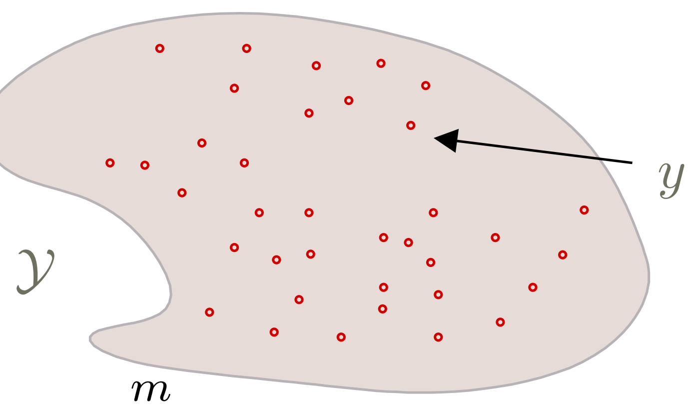
- The *log likelihood* is a *linear* function of feature values
- $\mathcal{Y}$ = set of syntactic structures (not necessarily trees)
- $f_j(y)$ = number of occurences of jth feature in $y \in \mathcal{Y}$

  (these features need not be conventional linguistic features)
- w_j are "feature weight" parameters

$$S_w(y) = \sum_{j=1}^{m} w_j f_j(y)$$

$$V_w(y) = \exp S_w(y)$$

$$Z_w = \sum_{y \in \mathcal{Y}} V_w(y)$$



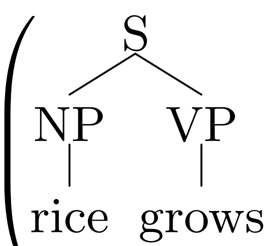
$$P_w(y) = \frac{V_w(y)}{Z_w}, \qquad \log P_\lambda(y) = \sum_{j=1}^{m} w_j f_j(y) - \log Z_w$$

PCFGs are log-linear models

$\mathcal{Y}$ = set of all trees generated by grammar G

$f_w(y)$ = number of times the jth rule is used in $y \in \mathcal{Y}$

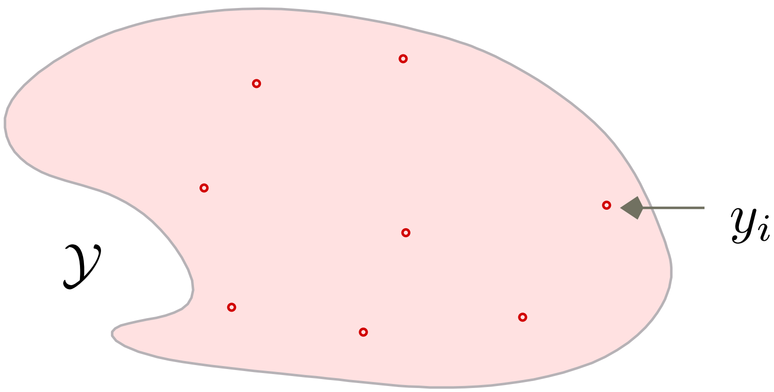
p_j = probability of jth rule in G $\qquad w_j = \log p_j$

$f\Bigg($ 

$\Bigg) = [\underbrace{1}_{\text{S}\rightarrow\text{NP VP}}, \underbrace{1}_{\text{NP}\rightarrow\text{rice}}, \underbrace{0}_{\text{NP}\rightarrow\text{bananas}}, \underbrace{1}_{\text{VP}\rightarrow\text{grows}}, \underbrace{0}_{\text{VP}\rightarrow\text{grow}}]$

$$\mathrm{P}_w(y) = \prod_{j=1}^{m} p_j^{f_j(y)} = \exp(\sum_{j=1}^{m} w_j f_j(\omega)) \quad \text{where } w_j = \log p_j$$

ML estimation for log linear models



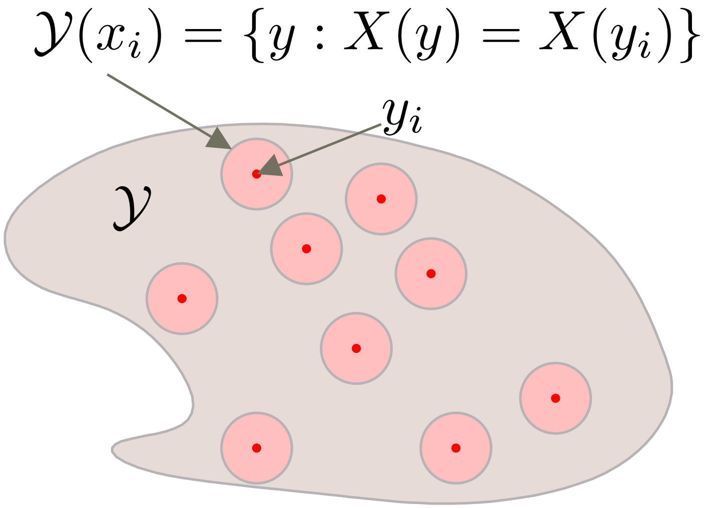
$$
\begin{aligned}
D &= y_1, \ldots, y_n \\
\widehat{w} &= \underset{w}{\operatorname{argmax}} L_D(w) \\
L_D(w) &= \prod_{i=1}^{n} \mathrm{P}_w(y_i)
\end{aligned}
$$

$$
\mathrm{P}_w(y) = \frac{V_w(y)}{Z_w} \quad V_w(y) = \exp \sum_j w_j f_j(y) \quad Z_w = \sum_{y' \in \mathcal{Y}} V_w(y')
$$

- For a PCFG, $\widehat{w}$ is easy to calculate, but …
- in general $\partial L_D / \partial w_j$ and Z_w are *intractable analytically and numerically*
- Abney (1997) suggests a Monte-Carlo calculation method

Conditional estimation and pseudo-likelihood

The *pseudo-likelihood* of w is the *conditional probability* of the *hidden part* (syntactic structure) w given its *visible part* (yield or terminal string) $x = X(y)$ (Besag 1974)



$$\begin{aligned} \widehat{w} &= \underset{\lambda}{\operatorname{argmax}} \operatorname{PL}_D(w) \\ \operatorname{PL}_D(w) &= \prod_{i=1}^{n} \operatorname{P}_\lambda(y_i|x_i) \\ \operatorname{P}_w(y|x) &= \frac{V_w(y)}{Z_w(x)} \end{aligned}$$

$$V_w(y) = \exp \sum_j w_j f_j(y) \qquad Z_w(x) = \sum_{y' \in \mathcal{Y}(x)} V_w(y')$$

Conditional estimation

- The pseudo-partition function $Z_w(x)$ is *much easier to compute* than the partition function Z_w
  - Z_w requires a sum over $\mathcal{Y}$
  - $Z_w(x)$ requires a sum over $\mathcal{Y}(x)$ (parses of x)
- *Maximum likelihood* estimates full joint distribution
  - learns $\mathrm{P}(x)$ and $\mathrm{P}(y|x)$
- *Conditional ML* estimates a conditional distribution
  - learns $\mathrm{P}(y|x)$ but not $\mathrm{P}(x)$
  - conditional distribution is what you need for parsing
  - cognitively more plausible?
- Conditional estimation requires labelled training data: no obvious EM extension

Conditional estimation

	Correct parse's features	**All other parses' features**
sentence 1	$[1, 3, 2]$	$[2, 2, 3]$ $[3, 1, 5]$ $[2, 6, 3]$
sentence 2	$[7, 2, 1]$	$[2, 5, 5]$
sentence 3	$[2, 4, 2]$	$[1, 1, 7]$ $[7, 2, 1]$
...	...	...

- Training data is *fully observed* (i.e., parsed data)
- Choose w to maximize (log) likelihood of *correct* parses relative to other parses
- Distribution of *sentences* is ignored
- *Nothing is learnt from unambiguous examples*
- Other discriminative learners solve this problem in different ways

Pseudo-constant features are uninformative

	Correct parse's features	**All other parses' features**
sentence 1	$[1, 3, \mathit{2}]$	$[2, 2, \mathit{2}]$ $[3, 1, \mathit{2}]$ $[2, 6, \mathit{2}]$
sentence 2	$[7, 2, \mathit{5}]$	$[2, 5, \mathit{5}]$
sentence 3	$[2, 4, \mathit{4}]$	$[1, 1, \mathit{4}]$ $[7, 2, \mathit{4}]$
$\ldots$	$\ldots$	$\ldots$

- *Pseudo-constant features* are identical within every set of parses
- They contribute the same constant factor to each parses' likelihood
- They do not distinguish parses of any sentence $\Rightarrow$ irrelevant

Pseudo-maximal features ⇒ unbounded $\widehat{w_j}$

	Correct parse's features	**All other parses' features**
sentence 1	$[1, \mathit{3}, 2]$	$[2, \mathit{3}, 4]$ $[3, \mathit{1}, 1]$ $[2, \mathit{1}, 1]$
sentence 2	$[2, \mathit{7}, 4]$	$[3, \mathit{7}, 2]$
sentence 3	$[2, \mathit{4}, 4]$	$[1, \mathit{1}, 1]$ $[1, \mathit{2}, 4]$

- A *pseudo-maximal feature* always reaches its maximum value within a parse on the correct parse
- If f_j is pseudo-maximal, $\widehat{w_j} \to \infty$ (hard constraint)
- If f_j is pseudo-minimal, $\widehat{w_j} \to -\infty$ (hard constraint)

Regularization

- f_j is pseudo-maximal over training data $\not\Rightarrow$ f_j is pseudo-maximal over all $\mathcal{Y}$ (sparse data)

- With many more features than data, log-linear models can over-fit

- Regularization: add *bias* term to ensure $\widehat{w}$ is finite and small

- In these experiments, the regularizer is a polynomial penalty term

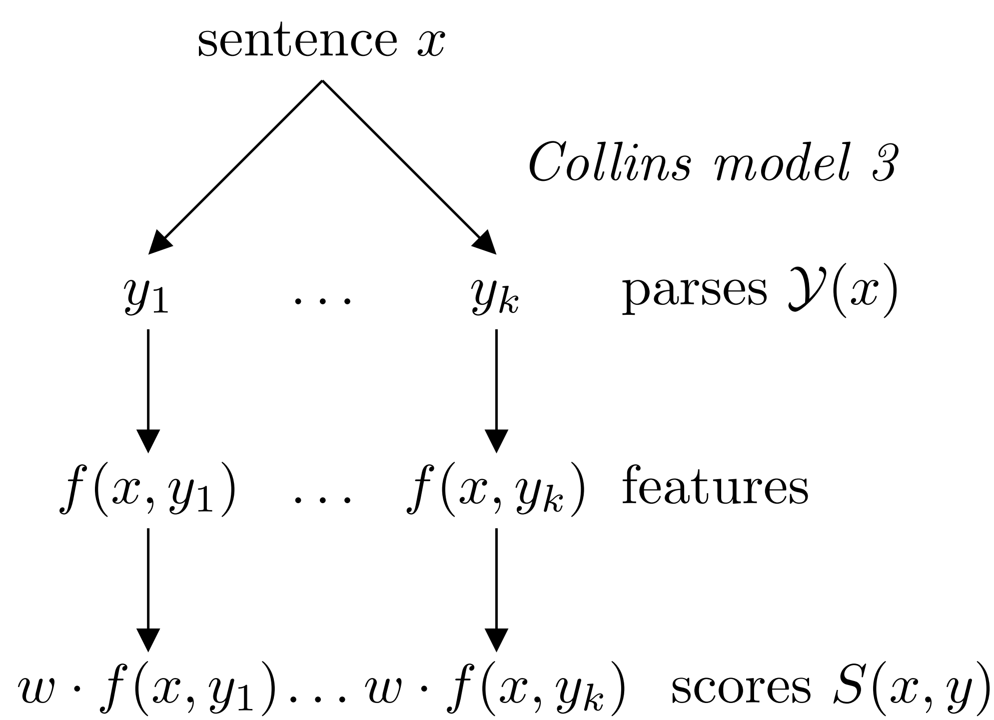
$$\widehat{w} \;=\; \underset{w}{\operatorname{argmax}} \log \mathrm{PL}_D(w) - c \sum_{j=1}^{m} |w_j|^p$$

Experiments in Discriminative Parsing

- Collins Model 3 parser produces a *set of candidate parses* $\mathcal{Y}(x)$ for each sentence x
- The discriminative parser has a weight w_j for each feature f_j
- The score for each parse is $S(x, y) = w \cdot f(x, y)$
- The highest scoring parse

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}(x)} S(x, y)$$

is predicted correct

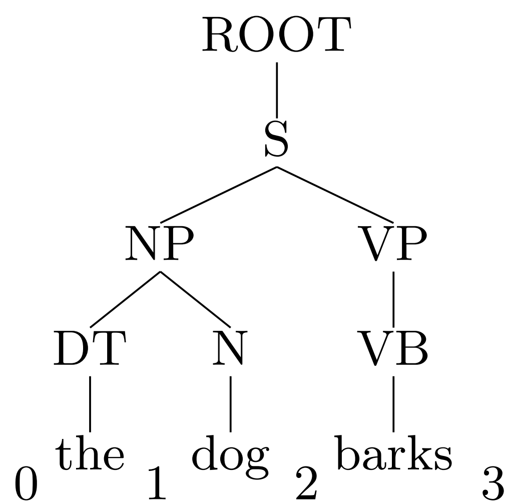


Evaluating a parser

- A node's *edge* is its label and beginning and ending *string positions*
- $E(y)$ is the set of edges associated with a tree y (same with forests)
- If y is a parse tree and $\bar{y}$ is the correct tree, then

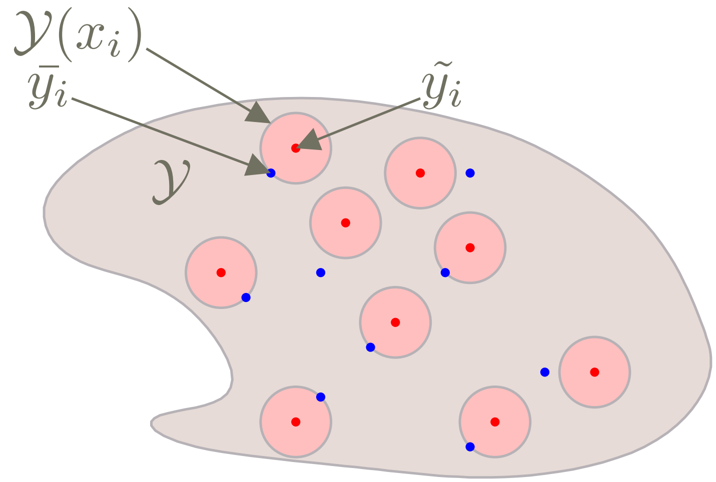
precision	$P_{\bar{y}}(y) = \lvert E(y) \rvert / \lvert E(y) \cap E(\bar{y}) \rvert$
recall	$R_{\bar{y}}(y) = \lvert E(\bar{y}) \rvert / \lvert E(y) \cap E(\bar{y}) \rvert$
f score	$F_{\bar{y}}(y) = 2/(\mathrm{P}_{\bar{y}}(y)^{-1} + R_{\bar{y}}(y)^{-1})$

	Edges	
(0	NP	2)
(2	VP	3)
(0	S	3)



Training the discriminative parser

- The sentence x_i associated with each tree $\bar{y}_i$ in the training corpus is parsed with the Collins parser, yielding $\mathcal{Y}(x_i)$
- Best parse $\tilde{y}_i = \operatorname{argmax}_{y \in \mathcal{Y}(x_i)} F_{\bar{y}_i}(y)$
- w is chosen to maximize the regularized log pseudo-likelihood $\sum_i \log \mathrm{P}_w(\tilde{y}_i | x_i) + R(w)$



Baseline and oracle results

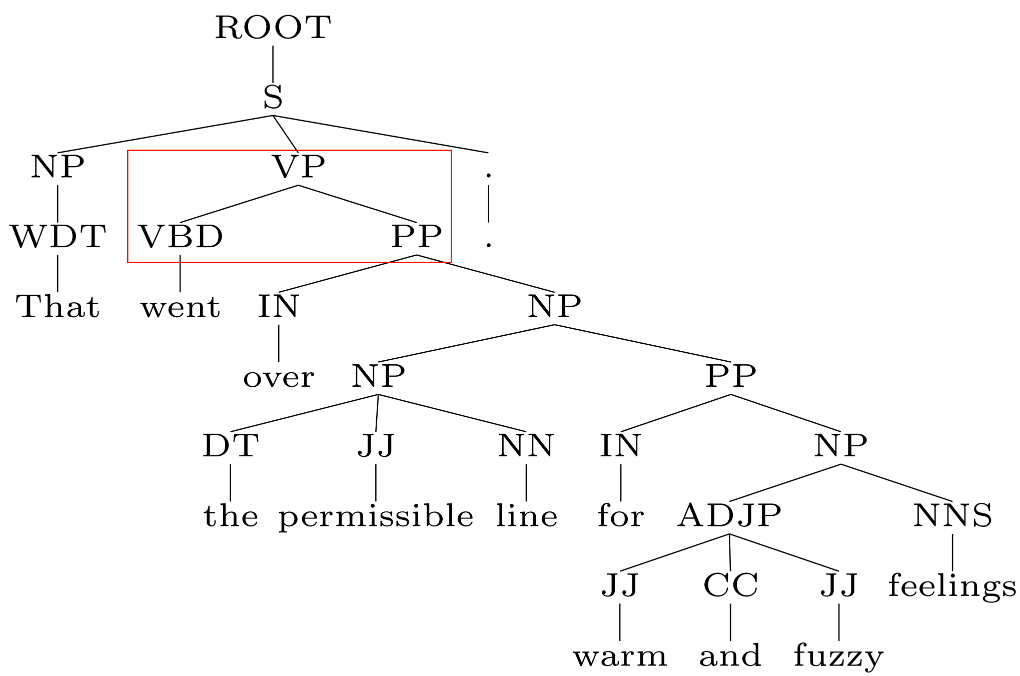
- Training corpus: 36,112 Penn treebank trees, development corpus 3,720 trees from sections 2-21
- Collins parser failed to produce a parse on 115 sentences
- Average $|\mathcal{Y}(x)| = 36.1$
- Collins parser f-score = 0.882 (picking parse with highest probability under Collins' generative model)
- Oracle (maximum possible) f-score = 0.953 (i.e., evaluate f-score of closest parses $\tilde{y}_i$)

$\Rightarrow$ Oracle (maximum possible) error reduction 0.601

Expt 1: Only "old" features

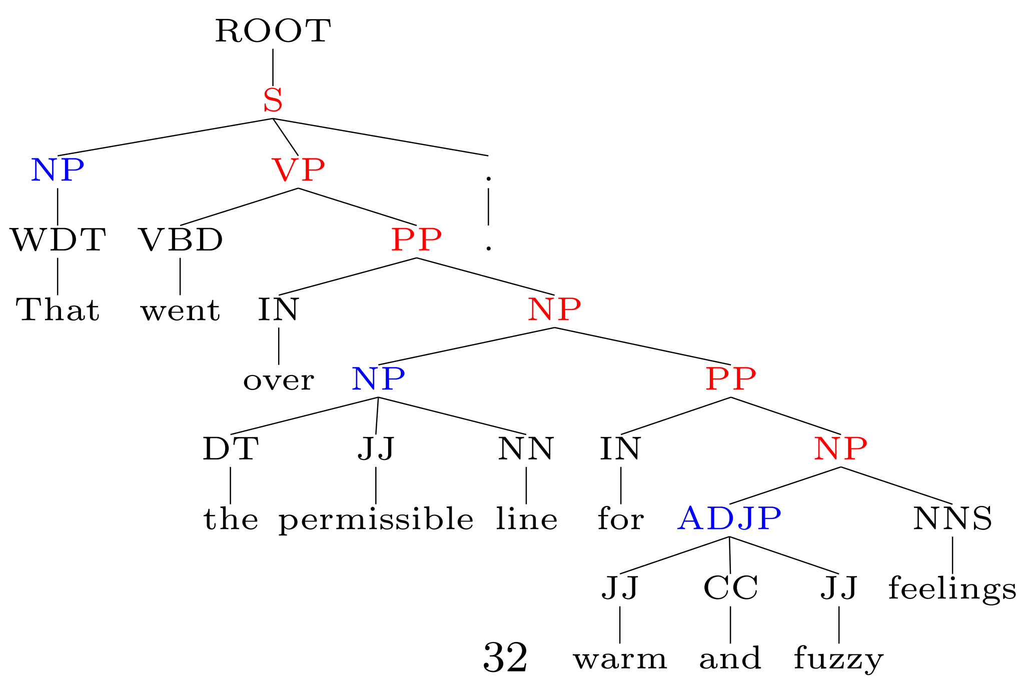
- Collins' parser already conditions on lexicalized rules
- Features: (1) log Collins probability, (9717) local tree features
- Feature selection: features must vary on 5 or more sentences
- Results: f-score $= 0.886$; $\approx 4\%$ error reduction

$\Rightarrow$ discriminative training may produce better parsers



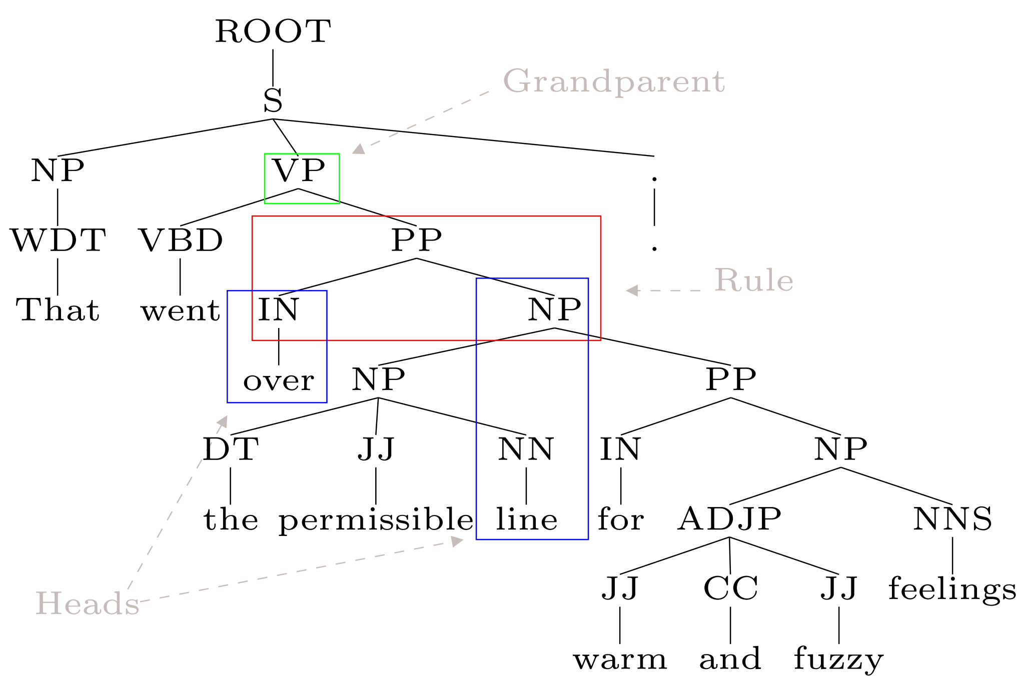
Expt 2: Rightmost branch bias

- The RightBranch feature's value is the number of nodes on the right-most branch (ignoring punctuation)
- Reflects the tendancy toward right branching
- LogProb + RightBranch: f-score = 0.884 (probably significant)
- LogProb + RightBranch + Rule: f-score = 0.889



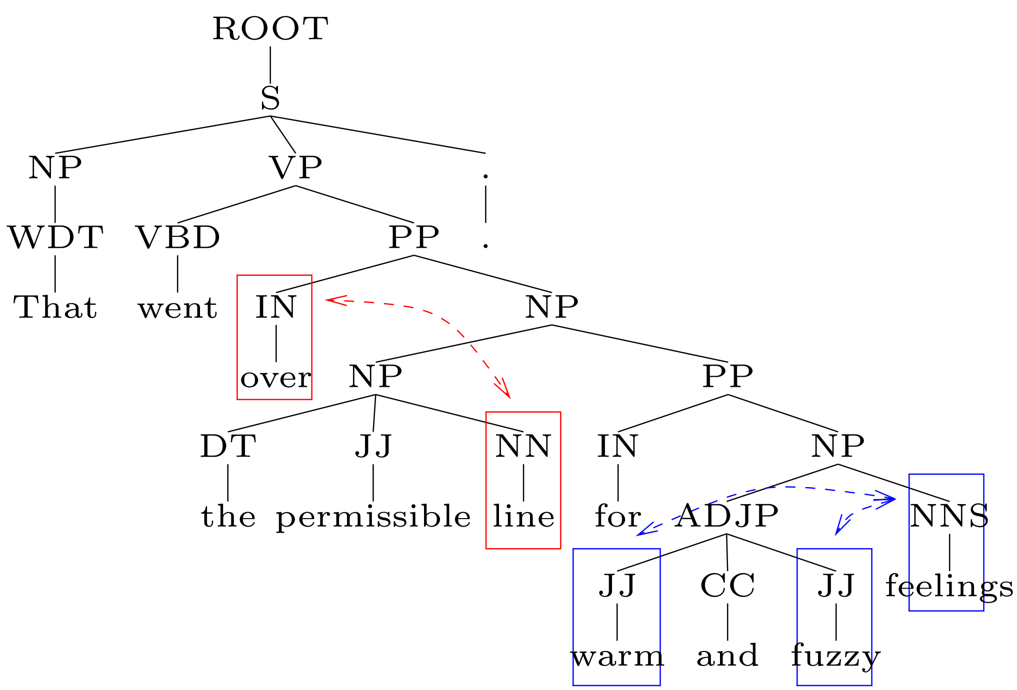
Lexicalized and parent-annotated rules

- *Lexicalization* associates each constituent with its head
- *Parent annotation* provides a little "vertical context"
- With all combinations, there are 158,890 rule features



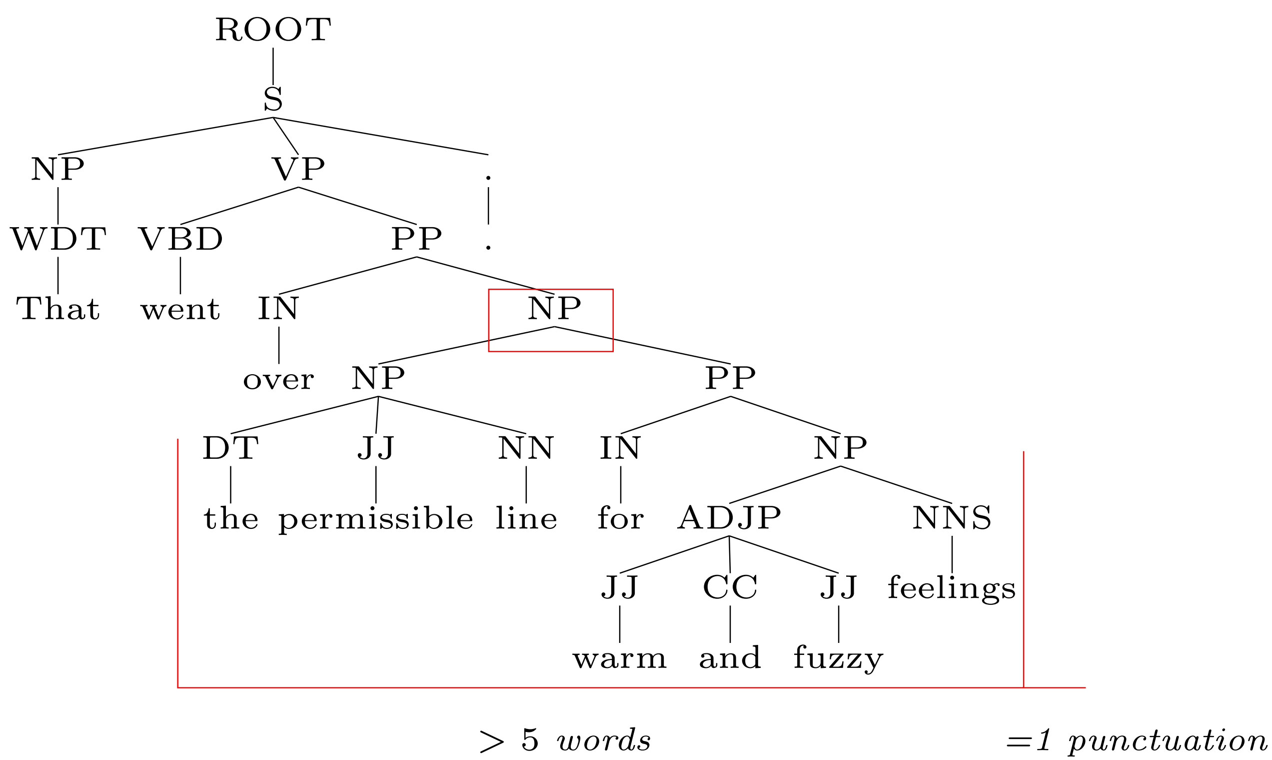
Head to head dependencies

- Head-to-head dependencies track the function-argument dependencies in a tree
- Co-ordination leads to phrases with multiple heads or functors
- With all combinations, there are 121,885 head-to-head features



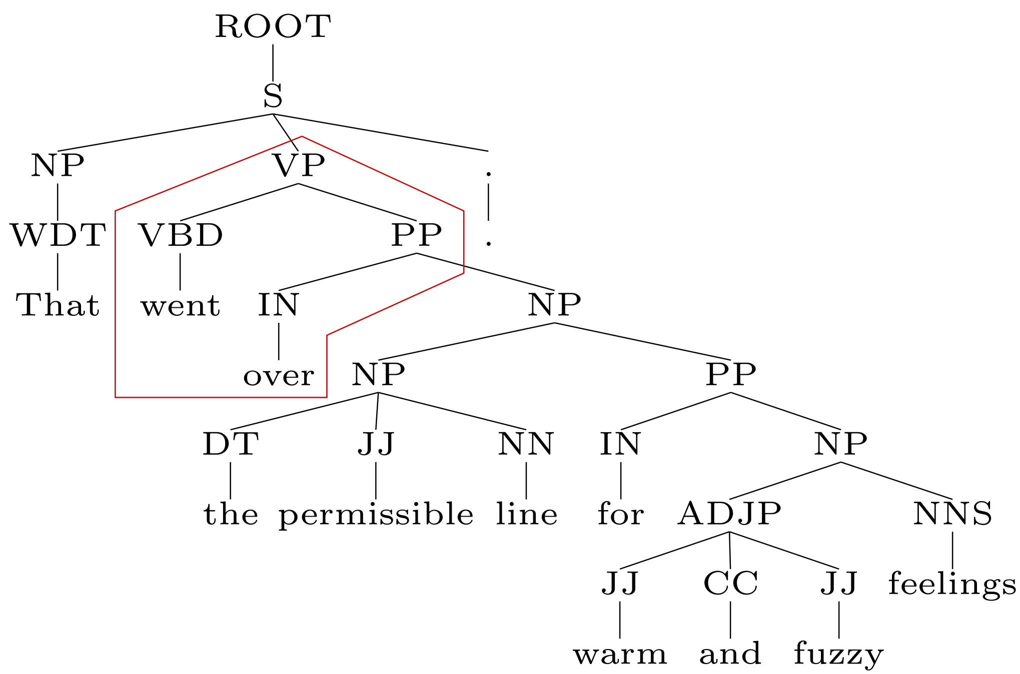
Constituent Heavyness and location

- Heavyness measures the constituent's category, its (binned) size and (binned) closeness to the end of the sentence
- There are 984 Heavyness features



Tree n-gram

- A tree n-gram are tree fragments that connect sequences of adjacent n words
- There are 62,487 tree n-gram features



Regularization

- General form of regularizer: $c \sum_j |w_j|^p$
- $p = 1$ leads to sparse weight vectors.
  - If $|\partial L / \partial w_j| < c$ then $w_j = 0$
- Experiment on small feature set:
  - 164,273 features
  - $c = 2, p = 2$, f-score $= 0.898$
  - $c = 4, p = 1$, f-score $= 0.896$, only 5,441 non-zero features!
  - Earlier experiments suggested that optimal performance is obtained with $p \approx 1.5$

Experimental results with all features

- 692,708 features
- regularization term: $4 \sum_j |w_j|^2$
- dev set results: *f-score* $= 0.9024$ (17% error reduction)

Cross-validating regularizer weights

- The features are naturally divided into classes
- Each class can be associated with its own regularizer constant c
- These regularizer classes can be adjusted to maximize performance on the dev set
- Evaluation is still running ...

Evaluating feature classes

Δ f-score	$\Delta - logL$	**features**	**zeroed class**
-0.0201874	1972.17	1	LogProb
-0.00443139	291.523	59567	NGram
-0.00434744	223.566	108933	Rule
-0.00359524	203.377	2	RightBranch
-0.00248663	62.5268	984	Heavy
-0.00220132	49.6187	195244	Heads
-0.00215015	71.6588	32087	Neighbours
-0.00162792	92.557	169903	NGramTree
-0.00119068	164.441	37068	Word
-0.000203843	-0.155993	1820	SynSemHeads
-1.42435e-05	-1.39079	18488	RHeads
9.98055e-05	0.237878	16140	LHeads

Other ideas we've tried

- Optimizing exp-loss instead of log-loss
- Averaged perceptron classifier with cross-validated feature class weights
- 2-level neural network classifier

Conclusion and directions for future work

- Discriminatively trained parsing models can perform better than standard generative parsing models
- Features can be arbitrary functions of parse trees
  - Are there techniques to help us explore the space of possible feature functions?
- Can these techniques be applied to problems that now require generative models?
  - speech recognition
  - machine translation