Probabilistic parsing with a wide variety of features

Mark Johnson

Brown University

IJCNLP, March 2004

Joint work with Eugene Charniak (Brown) and Michael Collins (MIT)

Supported by NSF grants LIS 9720368 and IIS0095940

Talk outline

- Statistical parsing models
- Discriminatively trained reranking models
  - features for selecting good parses
  - estimation methods
  - evaluation
- Conclusion and future work

Approaches to statistical parsing

- Kinds of models: "Rationalist" vs. "Empiricist"
  - based on *linguistic theories* (CCG, HPSG, LFG, TAG, etc.)
    - typically use specialized representations
  - *models of trees* in a training corpus (Charniak, Collins, etc.)
- Grammars are typically *hand-written* or *extracted from a corpus* (or both?)
  - both methods *require linguistic knowledge*
  - each method is affected differently by
    - lack of linguistic knowledge (or resources needed to enter it)
    - errors and inconsistencies

Features in linear models

- (Statistical) features are *real-valued functions* of parses (e.g., in a PCFG, the number of times a rule is used in a tree)
- A model associates a *real-valued weight* with each feature (e.g., the log of the rule's probability)
- The *score* of a parse is the weighted sum of its feature values (the tree's log probability)
- Higher scoring parses are more likely to be correct
- Computational complexity of estimation (training) depends on *how these features interact*

Feature dependencies and complexity

- *"Generative models"* (features and constraints induce *tree-structured dependencies*, e.g., PCFGs, TAGs)
  - maximum likelihood estimation is computationally cheap (counting occurences of features in training data)
  - crafting a model with a given set of features can be difficult
- *"Conditional"* or *"discriminative models"* (features have arbitrary dependencies, e.g., SUBGs)
  - maximum likelihood estimation is computationally intractible (as far as we know)
  - *conditional estimation* is computationally feasible but expensive
  - features can be *arbitrary functions* of parses

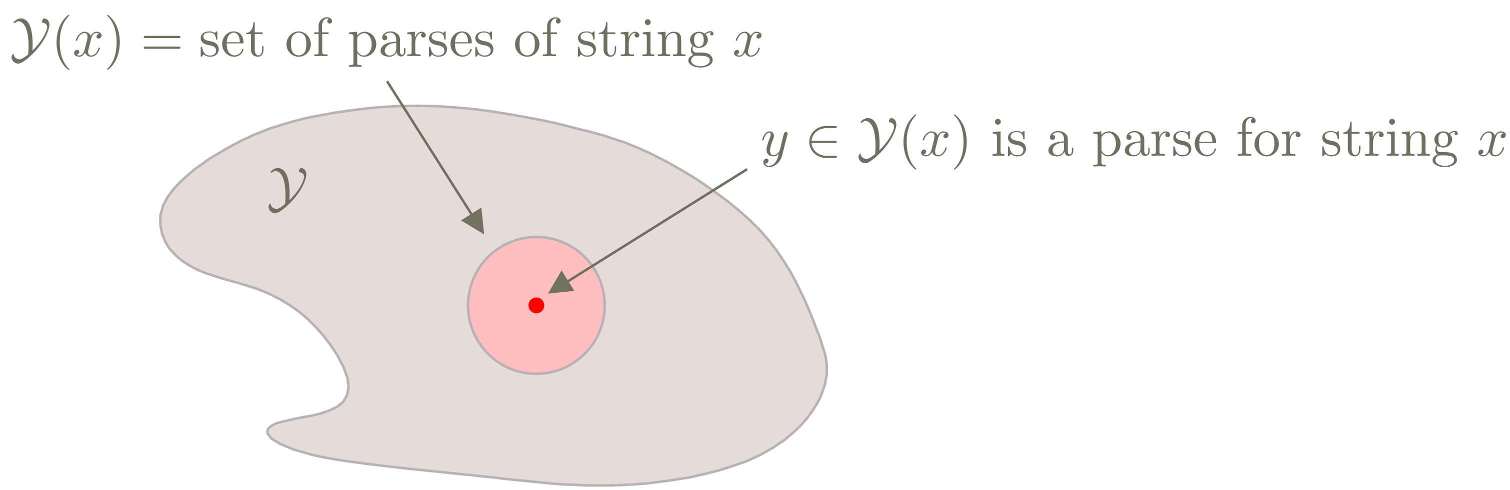
Why coarse-to-fine discriminative reranking?

- Question: *What are the best features for statistical parsing?*
- Intuition: *The choice of features matters more than the grammar formalism or parsing method*
- Are *global features* of the parse tree useful?

$\Rightarrow$ Choose a framework that makes experimenting with features as easy as possible

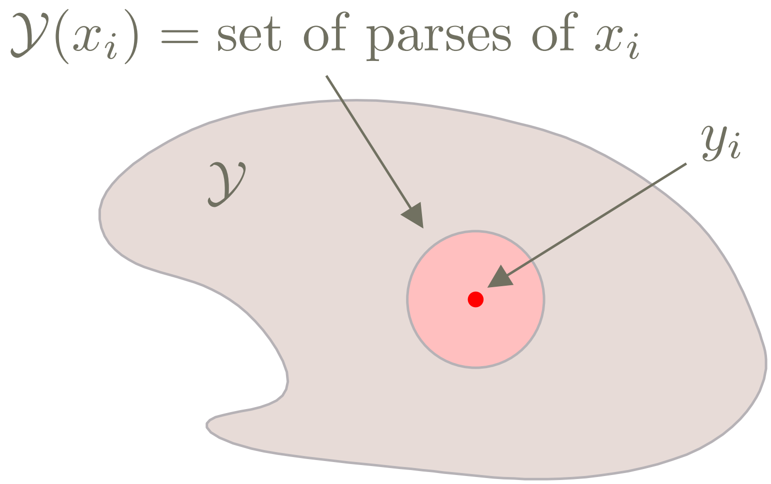
- *Coarse-to-fine discriminative reranking is such a framework*
  - features can be arbitrary functions of parse trees
  - computational complexity is manageable
- Why a Penn tree-bank parsing model?

The parsing problem



- $\mathcal{Y}$ = set of all parses, $\mathcal{Y}(x)$ = set of parses of string x
- $f = (f_1, \ldots, f_m)$ are real-valued *feature functions* (e.g., $f_{22}(y)$ = number of times an S dominates a VP in y)
- So $f(y) = (f_1(y), \ldots, f_m(y))$ is real-valued vector
- $w = (w_1, \ldots, w_m)$ is a *weight vector*, which we learn from training data
- $S_w(y) = w \cdot f(y) = \sum_{j=1}^{m} w_j f_j(y)$ is the *score* of a parse

Conditional training



- Labelled training data $D = ((x_1, y_1), \ldots, (x_n, y_n))$, where y_i is the correct parse for x_i
- Parsing: return the parse $y \in \mathcal{Y}(x)$ with the highest score
- Conditional training: Find a weight vector w so that the correct parse y_i scores "better" than any other parse in $\mathcal{Y}(x_i)$
- There are many different algorithms for doing this (MaxEnt, Perceptron, SVMs, etc.)

Another view of conditional training

	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	$[2, 4, 2]$	$[1, 1, 7]$ $[7, 2, 1]$
...	...	...

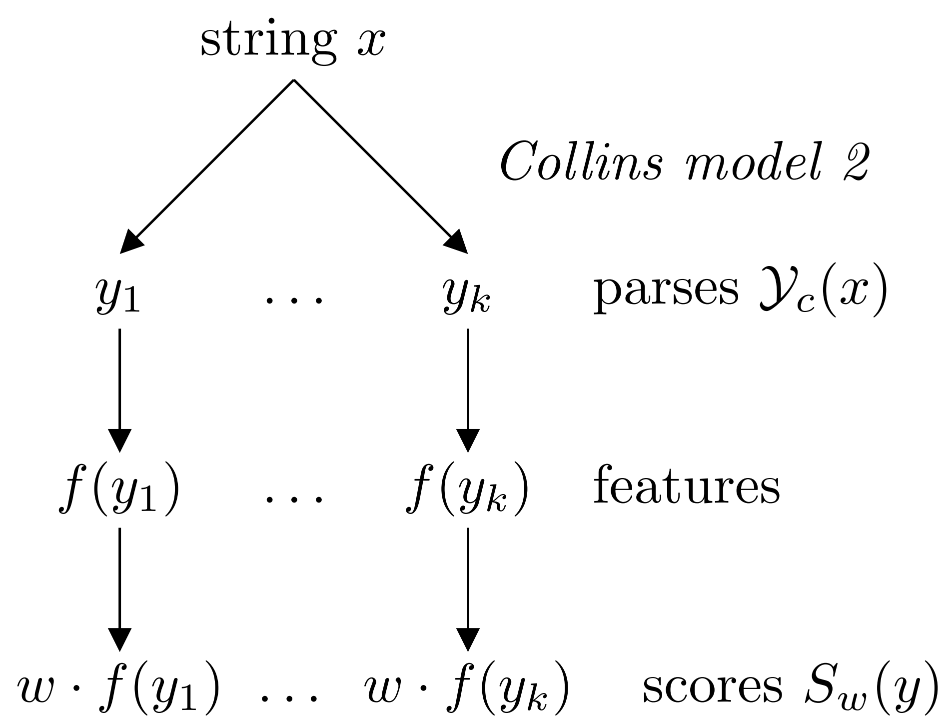
- Training data is *fully observed* (i.e., parsed data)
- Choose w to maximize score of *correct* parses relative to other parses
- Distribution of *sentences* is ignored
  - The models learnt by this kind of conditional training *can't be used as language models*
- *Nothing is learnt from unambiguous examples*

A coarse to fine approximation

- The set of parses $\mathcal{Y}(x)$ can be huge!
- Collins Model 2 parser produces a *set of candidate parses* $\mathcal{Y}_c(x)$ for each sentence x
- The score for each parse is $S_w(y) = w \cdot f(y)$
- The highest scoring parse

$$y^\star = \operatorname*{argmax}_{y \in \mathcal{Y}_c(x)} S_w(y)$$

is predicted correct



(Collins 1999 “Discriminative reranking”)

Advantages of this approach

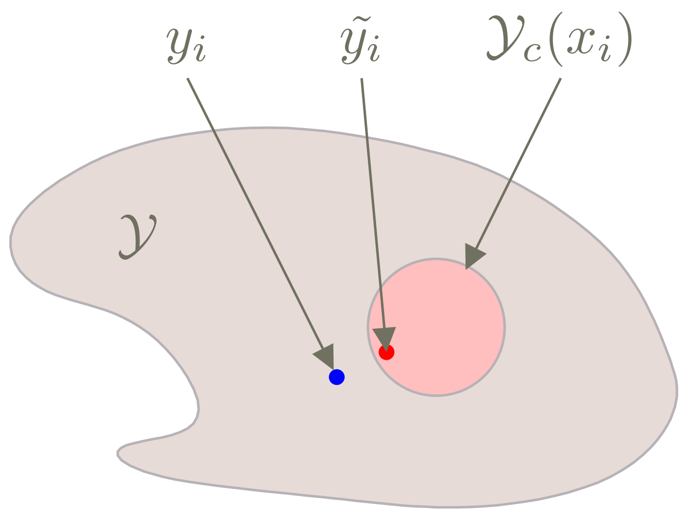
- The Collins parser only uses features for which there is a fast dynamic programming algorithm
- The set of parses $\mathcal{Y}_c(x)$ it produces is small enough that dynamic programming is not necessary
- This gives us almost complete freedom to formulate and explore possible features
- We're already starting from a good baseline ...
- ... but we only produce Penn treebank trees (instead of something deeper)
- and parser evaluation with respect to the Penn treebank is standard in the field

A complication

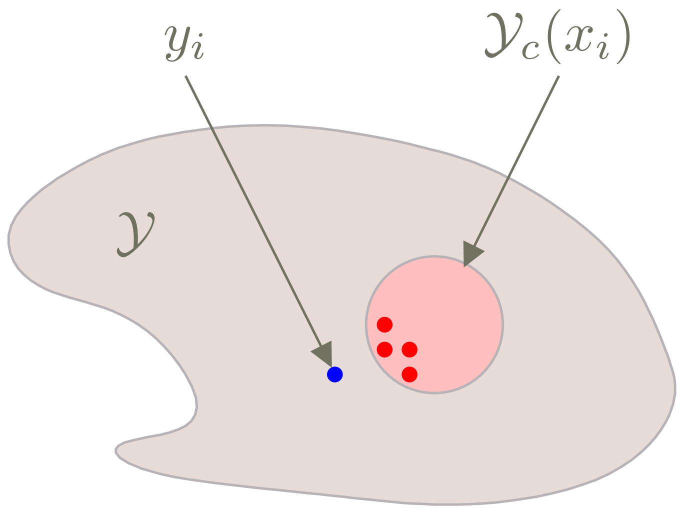
- Intuition: the discriminative learner should learn the common error modes of Collins parser
- Obvious approach: parse the training data with the Collins parser
- When parsed on the training section of the PTB, the Collins parser does much better on training section than it does on other text!
- Train the discriminative model from parser output on text parser was not trained on
- Use *cross-validation paradigm* to produce discriminative training data (divide training data into 10 sections)
- Development data described here is from PTB sections 20 and 21

Another complication

- Training data $((x_1, y_1), \ldots, (x_n, y_n))$
- Each string x_i is parsed using Collins parser, producing a set $\mathcal{Y}_c(x_i)$ of parse trees
- *The correct parse y_i might not be in the Collins parses $\mathcal{Y}_c(x_i)$*
- Let $\tilde{y}_i = \operatorname{argmax}_{y \in \mathcal{Y}_c(x_i)} F_{y_i}(y)$ be the *best Collins parse*, where $F_{y'}(y)$ measures parse accuracy
- Choose w to discriminate $\tilde{y}_i$ from the other $\mathcal{Y}_c(x_i)$



Multiple best parses



- There can be several Collins parses equally close to the correct parse: which one(s) should we declare to be the best parse?
- Weighting all close parses equally does not work as well (0.9025) as . . .
- picking the parse with the highest Collins parse probability (0.9036), but . . .
- letting the model pick its own winner from the close parses (EM-like scheme in Riezler '02) works best of all (0.904)

Baseline and oracle results

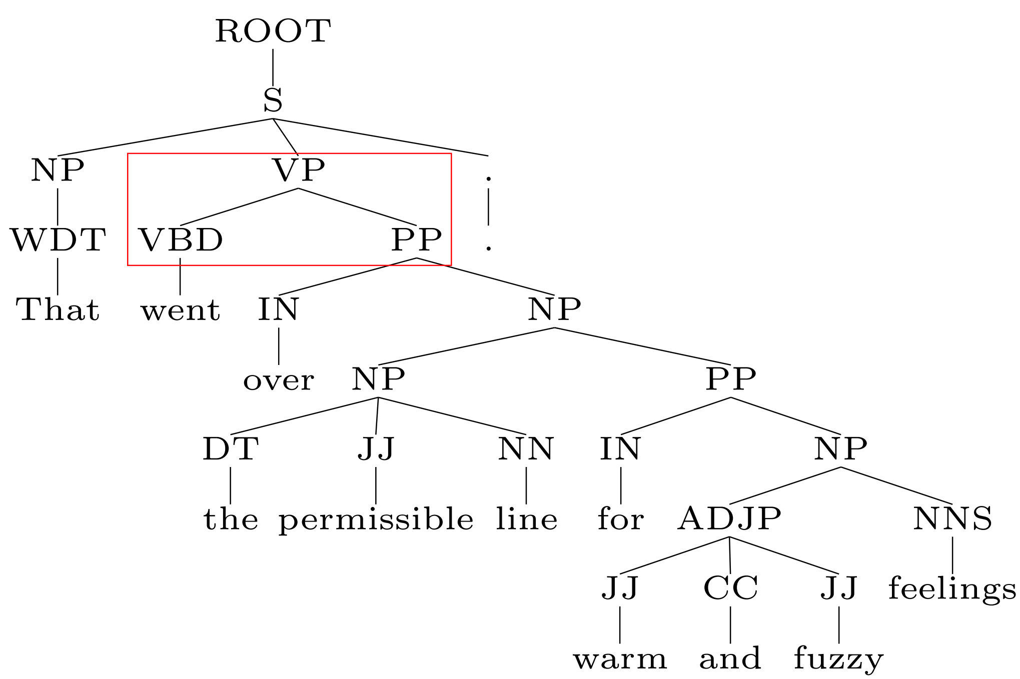
- Training corpus: 36,112 Penn treebank trees from sections 2–19, development corpus 3,720 trees from sections 20–21
- Collins Model 2 parser failed to produce a parse on 115 sentences
- Average $|\mathcal{Y}(x)| = 36.1$
- Model 2 f-score $= 0.882$ (picking parse with highest Model 2 probability)
- Oracle (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

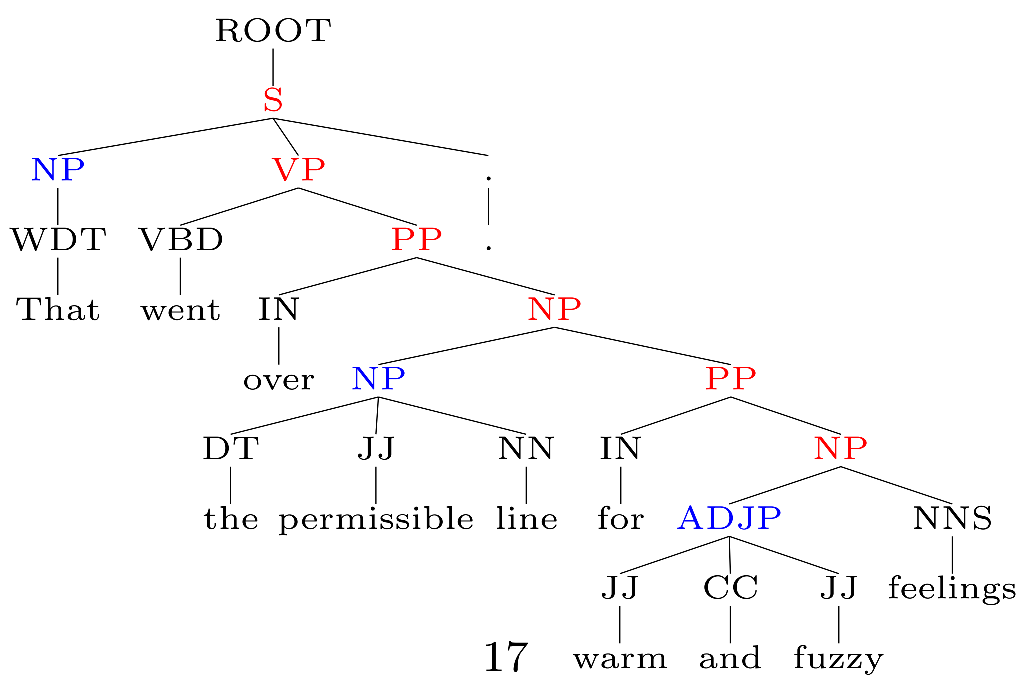
- Features: (1) *log Model 2 probability*, (9717) local tree features
- Model 2 already conditions on local trees!
- Feature selection: features must vary on 5 or more sentences
- Results: f-score $= 0.886$; $\approx 4\%$ error reduction

$\Rightarrow$ *discriminative training alone can improve accuracy*



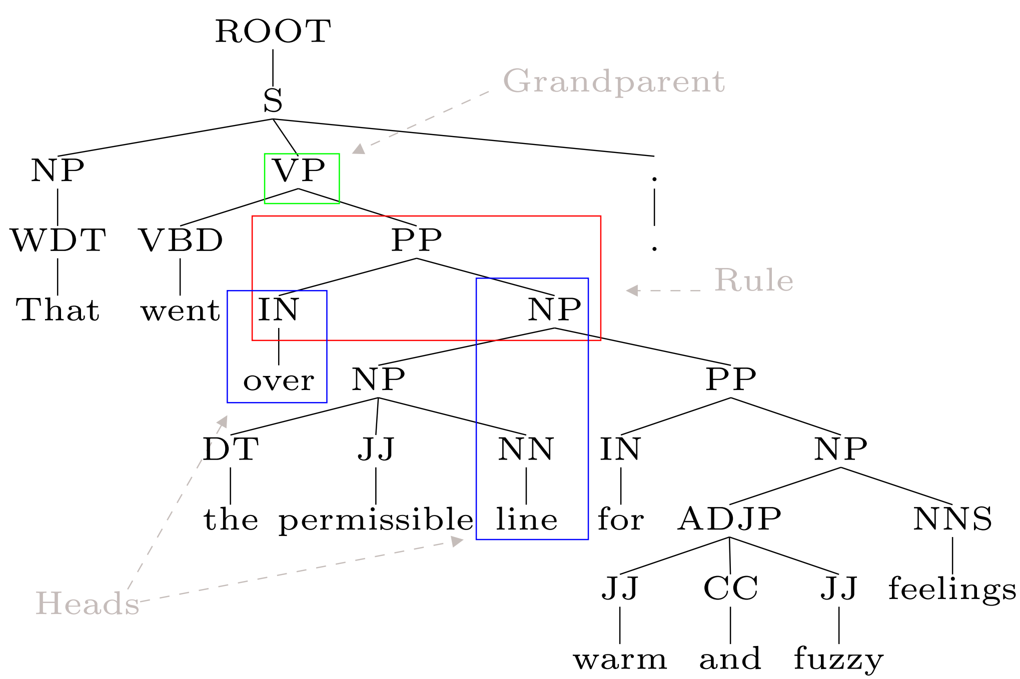
Expt 2: Rightmost branch bias

- 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



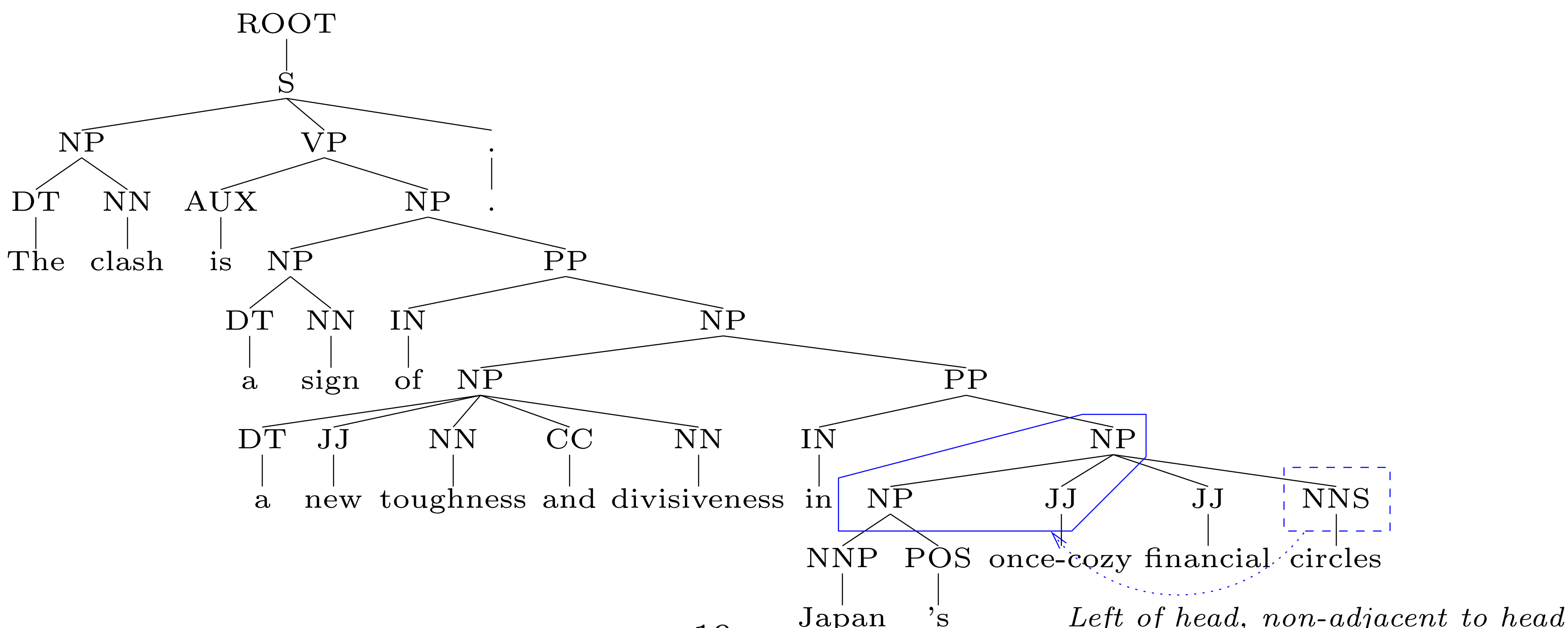
Lexicalized and parent-annotated rules

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



n-gram rule features generalize rules

- Collects adjacent constituents in a local tree
- Also includes relationship to head
- Constituents can be ancestor-annotated and lexicalized
- 5,143 unlexicalized rule bigram features, 43,480 lexicalized rule bigram features

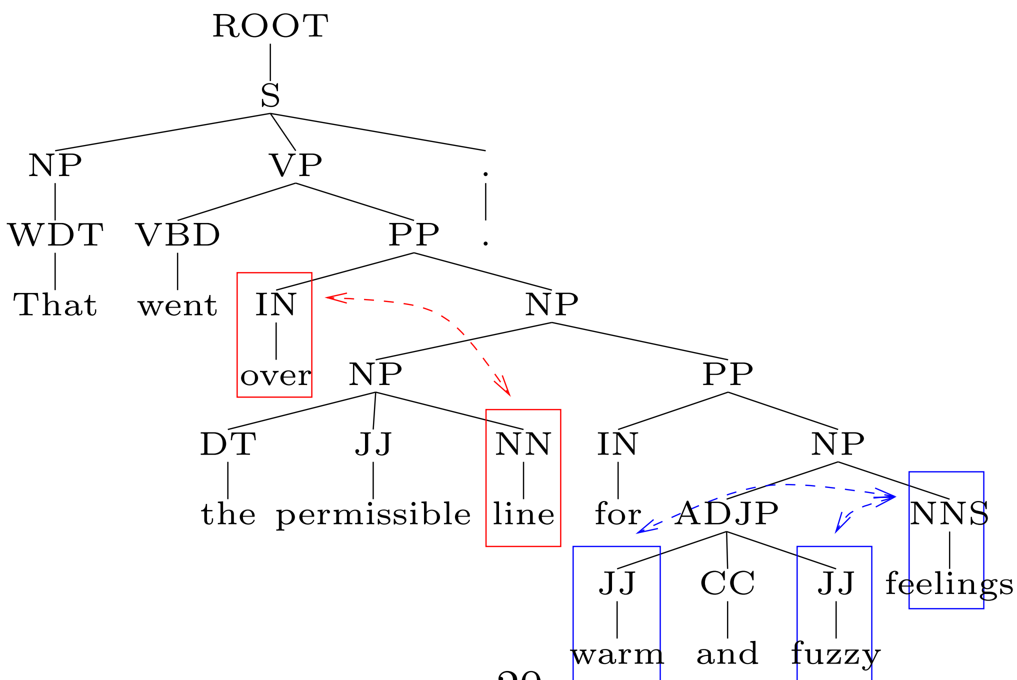


Head to head dependencies

- Head-to-head dependencies track the function-argument dependencies in a tree

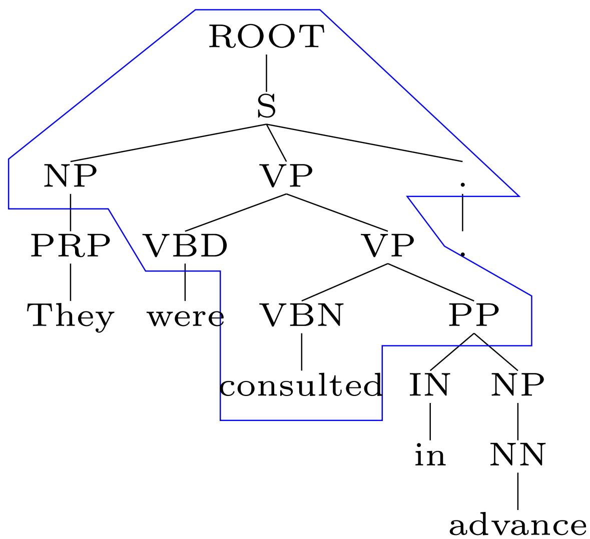
- *Co-ordination leads to phrases with multiple heads and arguments*

- With all combinations, there are 121,885 head-to-head features



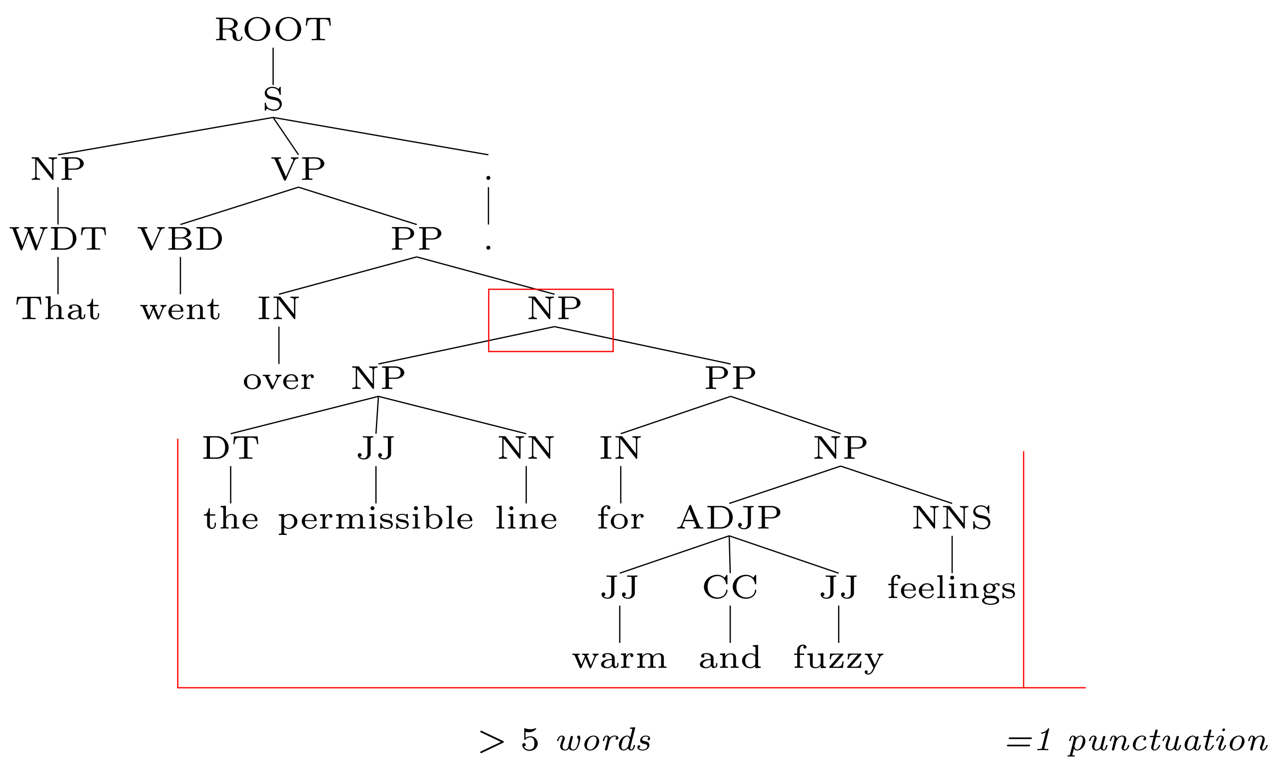
Head trees record all dependencies

- Head trees consist of a (lexical) head, all of its projections and (optionally) all of the siblings of these nodes
- These correspond roughly to TAG elementary trees



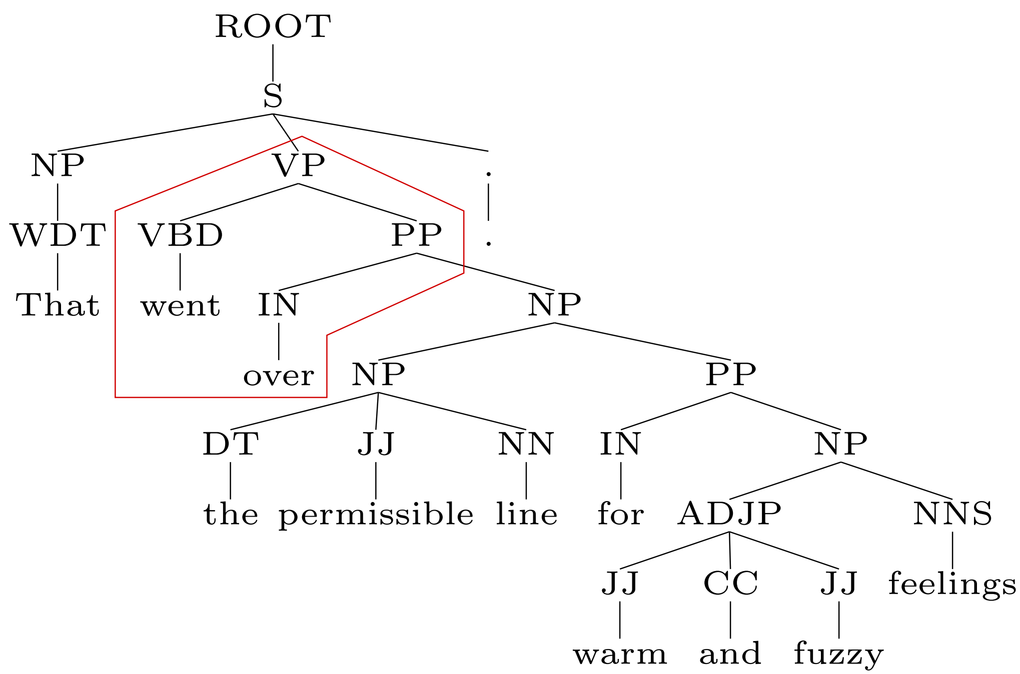
Constituent Heavyness and location

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



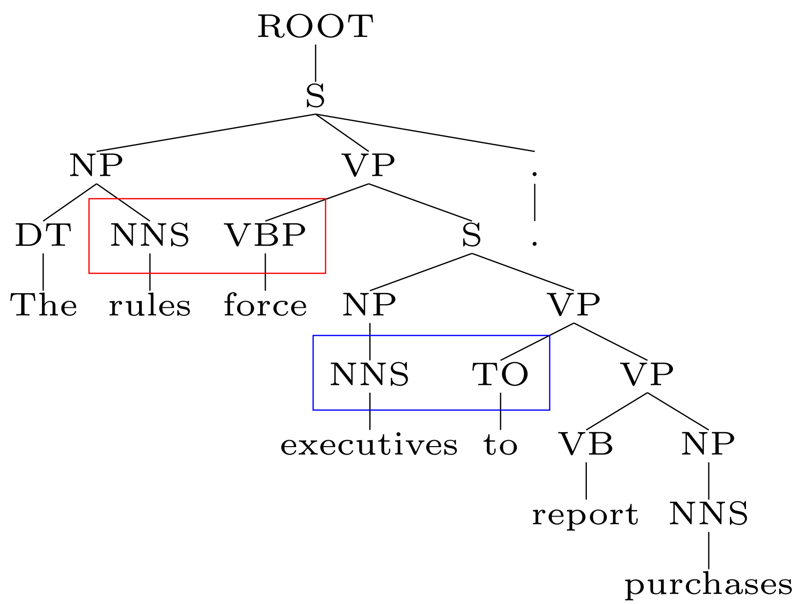
Tree n-gram

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



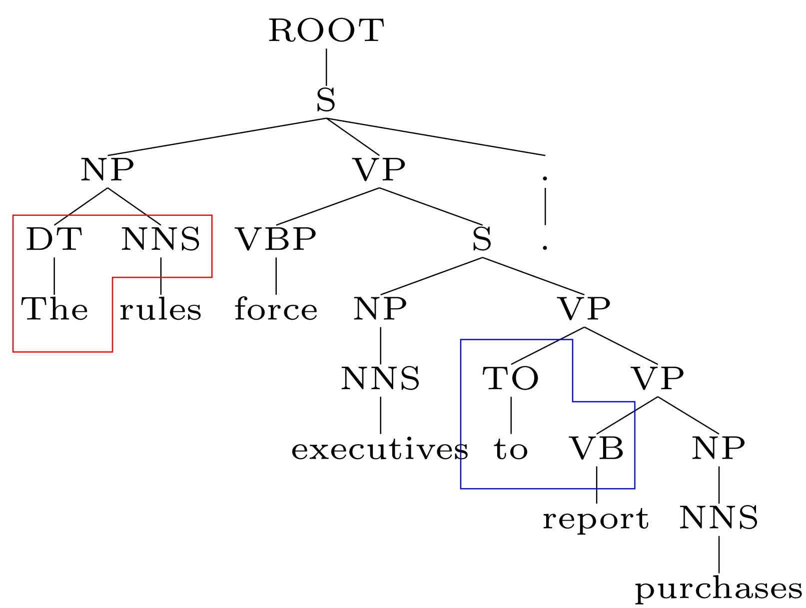
Subject-Verb Agreement

- The SubjVerbAgr features are the POS of the subject NP's lexical head and the VP's functional head
- There are 200 SubjVerbAgr features



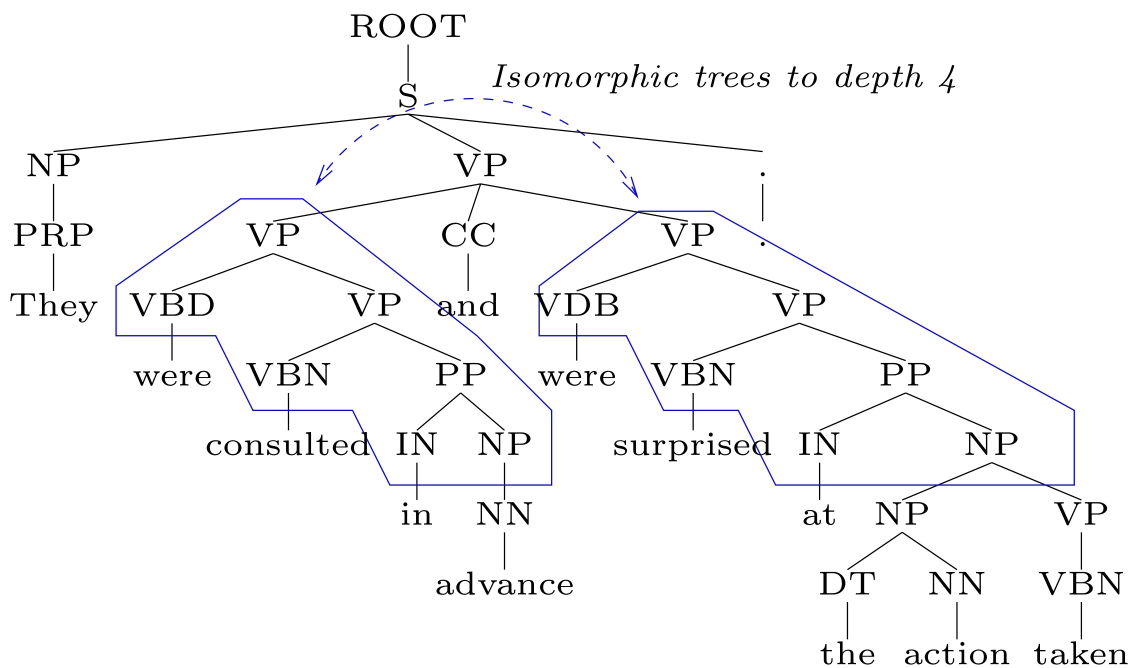
Functional-lexical head dependencies

- The SynSemHeads features collect pairs of functional and lexical heads of phrases (Grimshaw)
- This captures number agreement in NPs and aspects of other head-to-head dependencies
- There are 1,606 SynSemHeads features



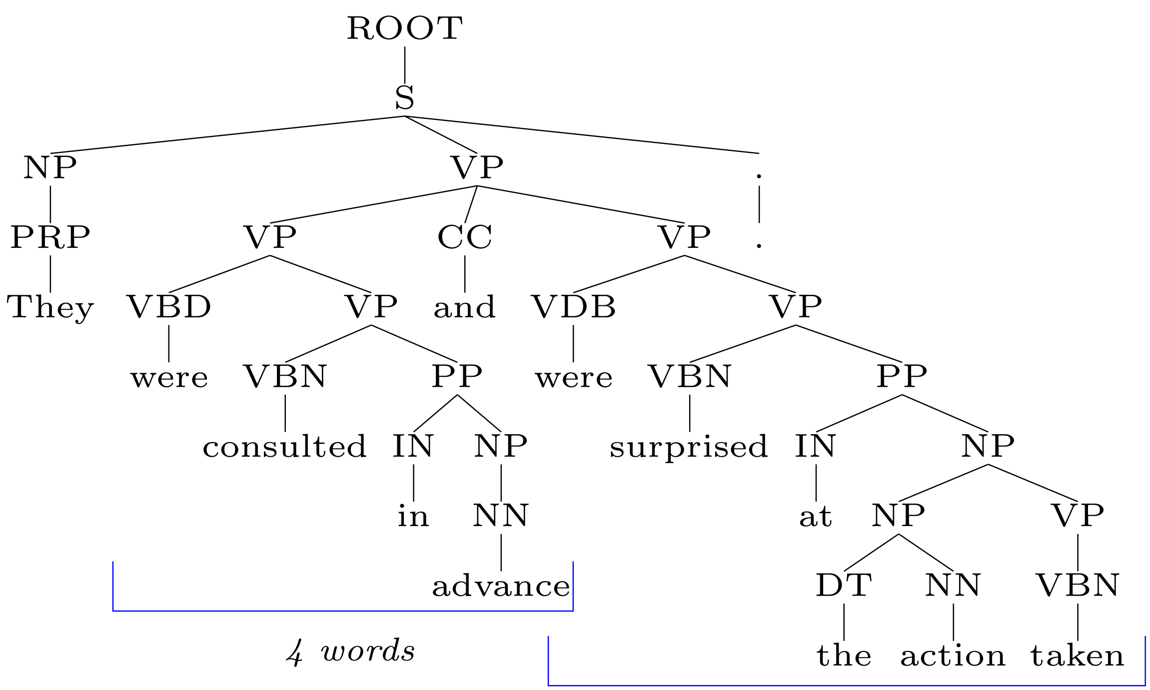
Coordination parallelism (1)

- The CoPar feature indicates the depth to which adjacent conjuncts are parallel
- There are 9 CoPar features



Coordination parallelism (2)

- The CoLenPar feature indicates the difference in length in adjacent conjuncts and whether this pair contains the last conjunct.
- There are 22 CoLenPar features



CoLenPar feature: (2,true)

Experimental results with all features

- Feature selection: features must vary on parses of at least 5 sentences in training data (a cutoff of 2 improves results)
- In this experiment, 883,936 features
- *log loss* with Gaussian regularization term: $11 \sum_j w_j^2$
  - dev set results: f-score = 0.903–0.904
  - *section 23 results: f-score = 0.9039 (≈20% error reduction), 47% of sentences have f-score = 1*
- *exp loss* with Gaussian regularization term: $50 \sum_j w_j^2$
  - dev set results: f-score = 0.902
- *averaged perceptron classifier* (very fast!)
  - dev set results: f-score = 0.902 (with feature class tuning)

Which kinds of features are best?

	# of features	f-score
Treebank trees	375,646	0.901
Correct parses	271,267	0.902
Incorrect parses	876,339	0.903
Correct & incorrect parses	883,936	0.903

- Features from incorrect parses characterize failure modes of Collins parser
- There are far more ways to be wrong than to be right!

Feature classes overview

# of feat.	av. value	s.d.	feat. class
1	0.416674	–	LogProb
2	-0.376498	0.000265398	RightBranch
9	0.117017	0.0371904	CoPar
22	0.0133718	0.0196021	CoLenPar
200	-0.000552325	0.00364032	SubjVerbAgr
984	-0.00118015	0.00613362	Heavy
1606	0.00145433	0.00196207	SynSemHeads
37068	0.000505719	0.000953109	Word
48623	6.68076e-05	0.00145942	NGram
122189	0.000623527	0.000679083	WProj
160582	0.00063112	0.000969829	Heads
203979	0.000393769	0.000832161	NGramTree
223354	0.000344003	0.000813581	Rule

Evaluating feature classes

Δ f-score	Δ $-\log$ CP	Δ correct	Δ best poss.	zeroed class
-0.00909743	3042.76	-123	-132	LogProb
-0.0034855	-107.341	17	-42	Rule
-0.00316443	120.551	-31	-64	NGram
-0.00292884	50.4752	-20	-44	Heads
-0.00248576	73.3785	-18	-25	Heavy
-0.00239372	251.753	-74	-27	RightBranch
-0.00208603	157.478	-19	-31	NGramTree
-0.00199449	130.832	-28	-36	WProj
-0.000761952	11.0709	5	-4	Word
-0.000422497	7.1691	6	-5	CoLenPar
-0.000368866	-14.2518	1	2	SynSemHeads
-0.000230322	11.3504	-9	-4	CoPar
-0.000100725	-14.7814	-2	0	SubjVerbAgr

Informal error analysis

- Manual examination of first 100 sentences of development data
- Preliminary classification of “type” of parser error
- Multiple errors per sentence were found

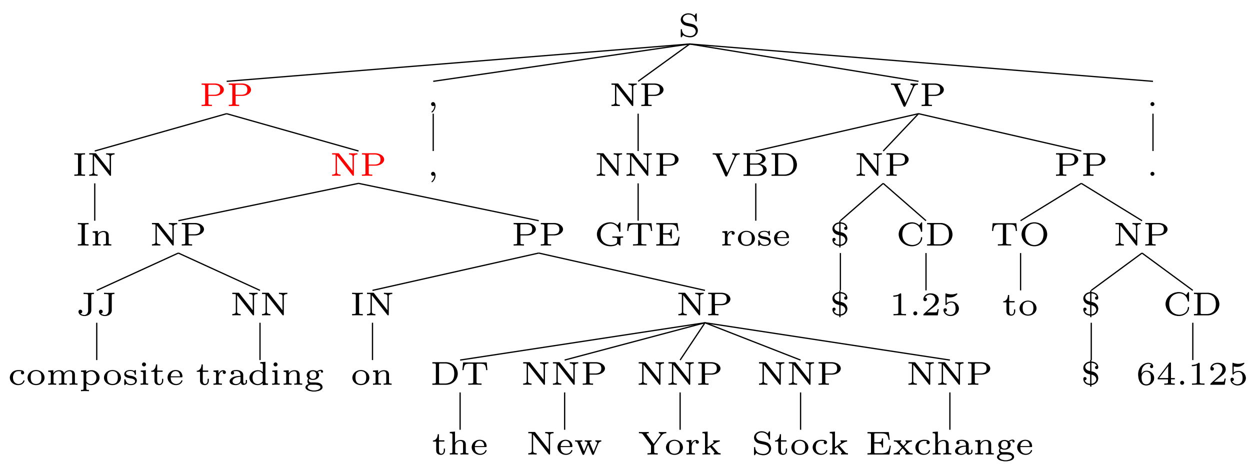
Error type	Reranker	Coarse parser
PP attach	19	3
Coordination	8	2
Category misanalysis	7	1
Other attachment	4	9
Compounding	2	3
Other errors	2	4

14 PTB errors, 7 PTB ambiguities

(Suggested by Yusuke Miyao)

Sample PP attachment error (1/2)

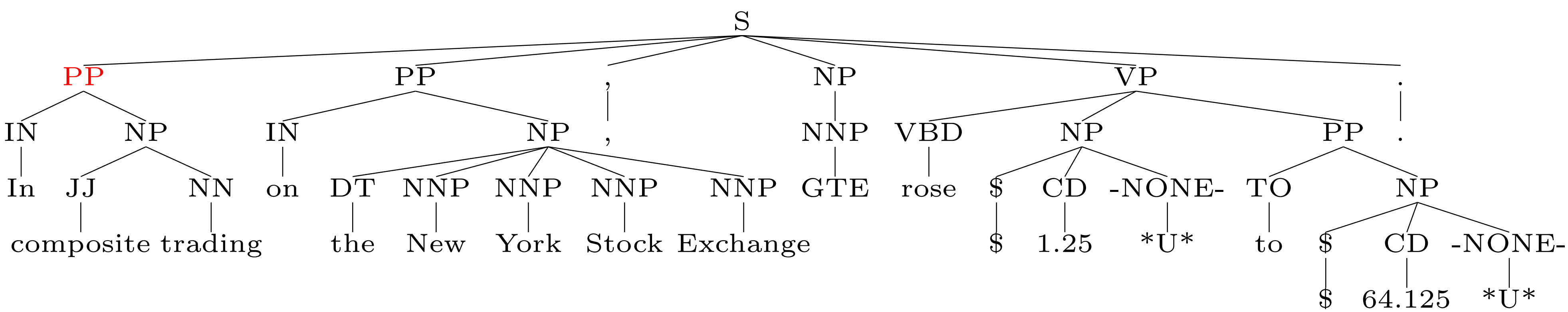
In composite trading on the New York Stock Exchange, GTE rose $1.25 to $64.125.



Parse tree

Sample PP attachment error (2/2)

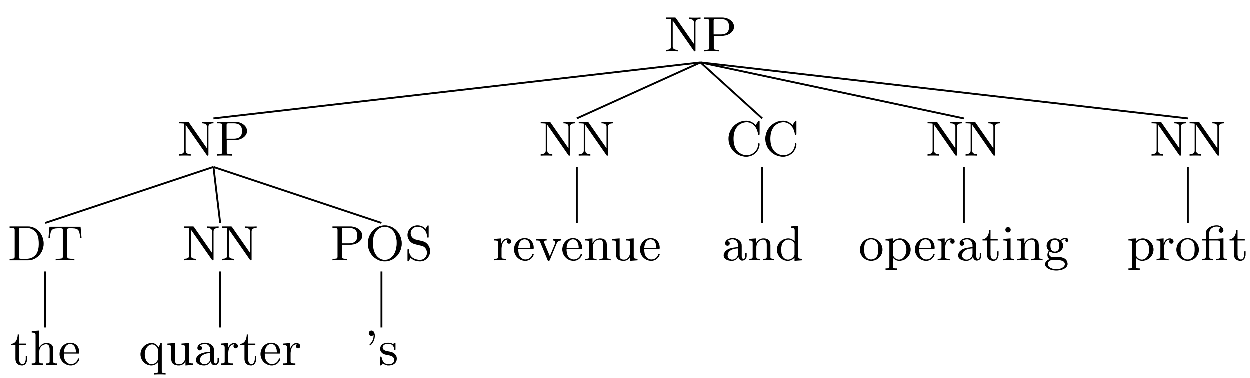
In composite trading *on the New York Stock Exchange*, *GTE rose \$1.25 to \$64.125.*



Gold (treebank) tree

Coordination error (1/2)

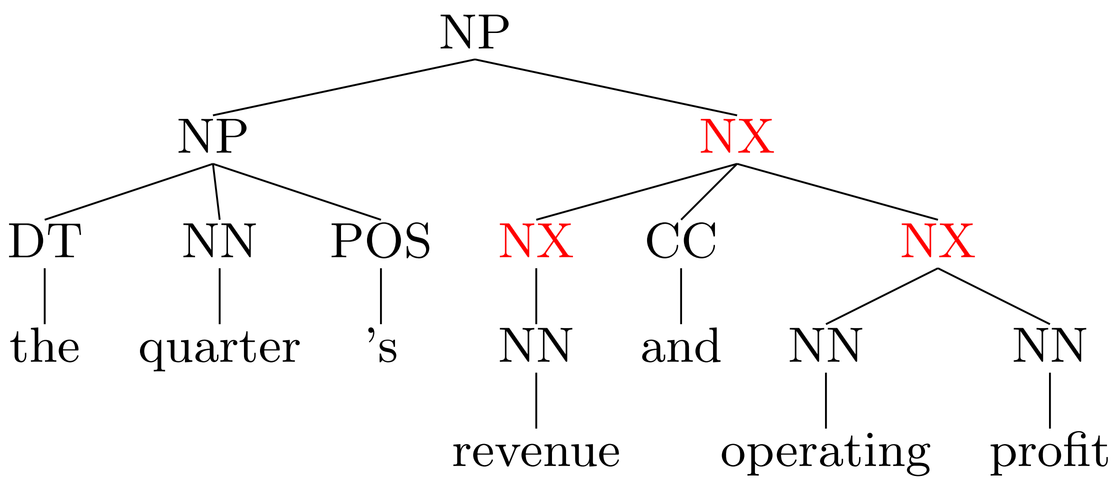
Earlier rate reductions in Texas and California reduced the quarter's revenue and operating profit $55 million; a year earlier, operating profit in telephone operations was reduced by a similar amount as a result of a provision for a reorganization.



Parse tree

Coordination error (2/2)

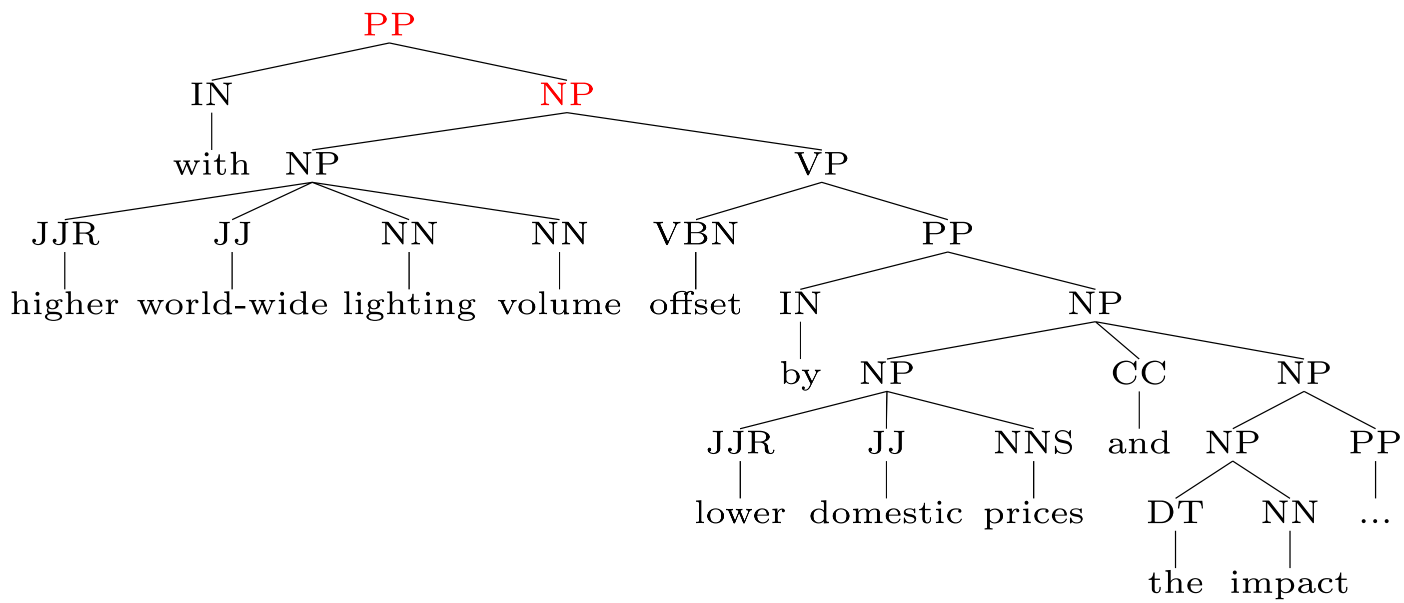
Earlier rate reductions in Texas and California reduced the quarter's revenue and operating profit $55 million; a year earlier, operating profit in telephone operations was reduced by a similar amount as a result of a provision for a reorganization.



Gold (treebank) tree

Category misanalysis error (1/2)

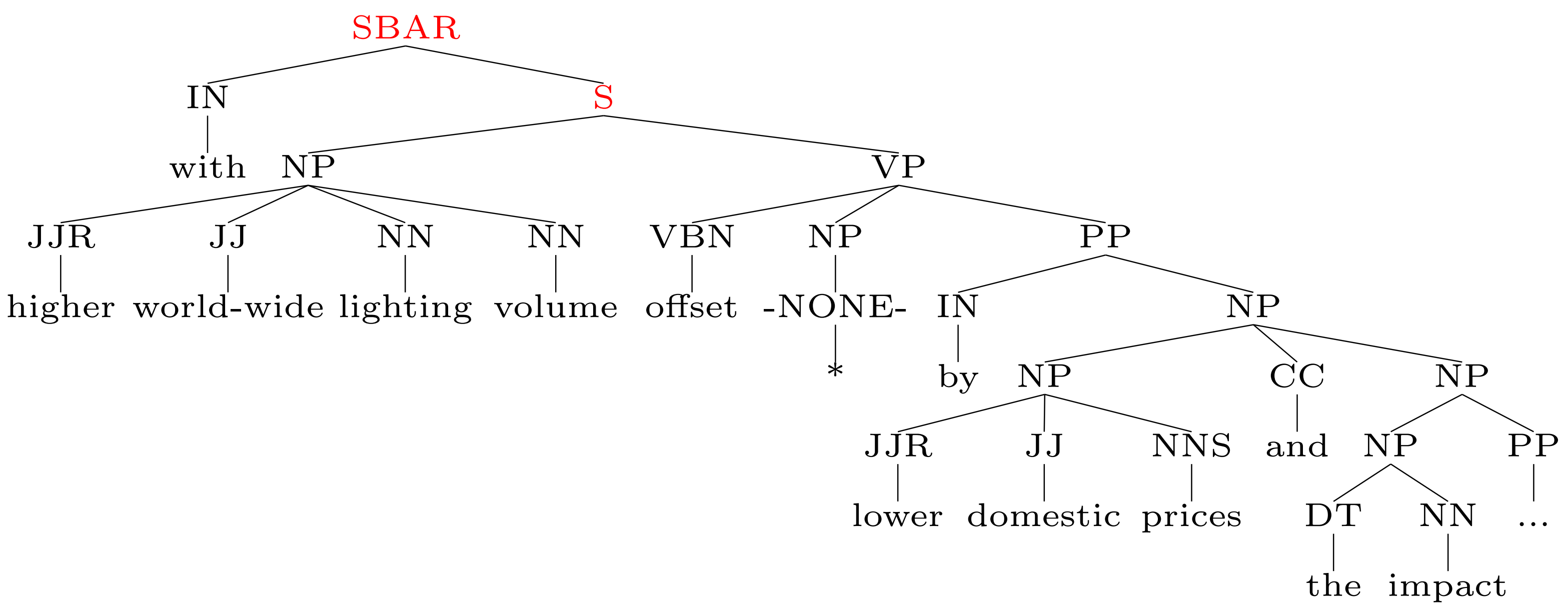
Electrical products' sales fell to \$496.7 million from \$504.5 million with higher world-wide lighting volume offset by lower domestic prices and the impact of weaker currencies in Europe and South America.



Parse tree

Category misanalysis (2/2)

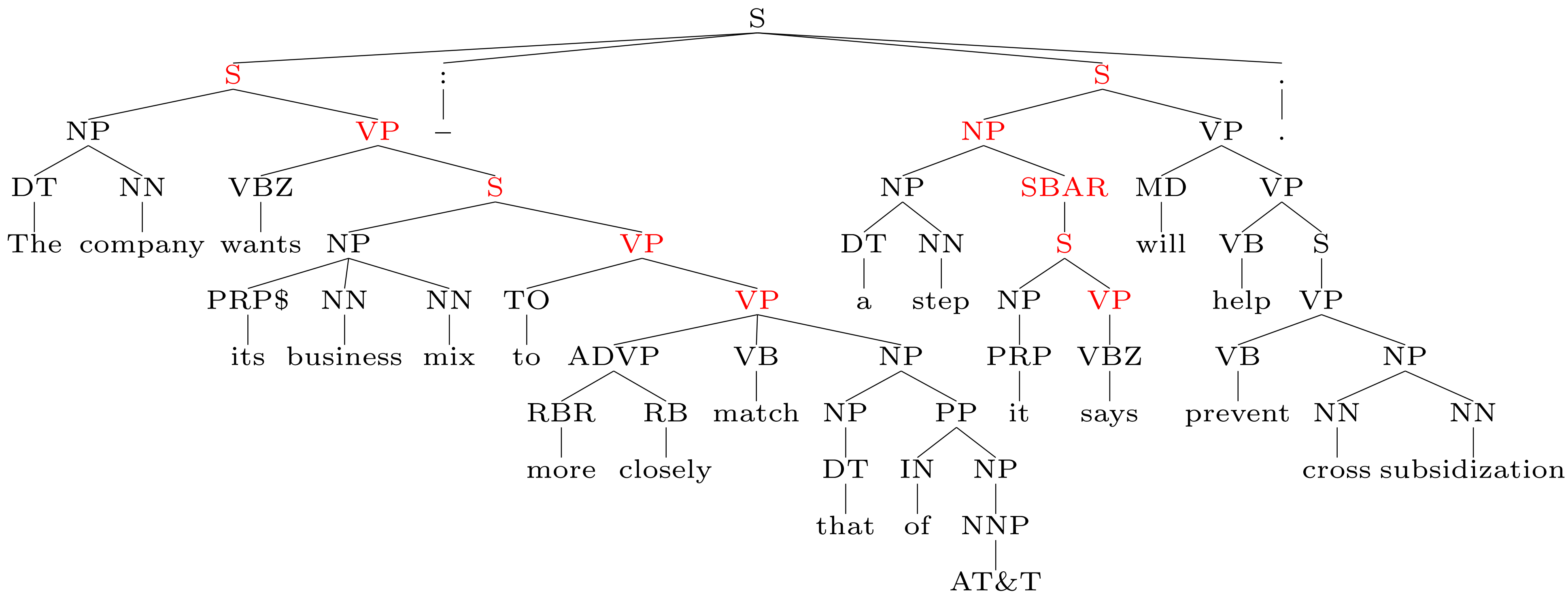
Electrical products' sales fell to \$496.7 million from \$504.5 million with higher world-wide lighting volume offset by lower domestic prices and the impact of weaker currencies in Europe and South America.



Gold (treebank) tree

Multiple attachment errors (1/3)

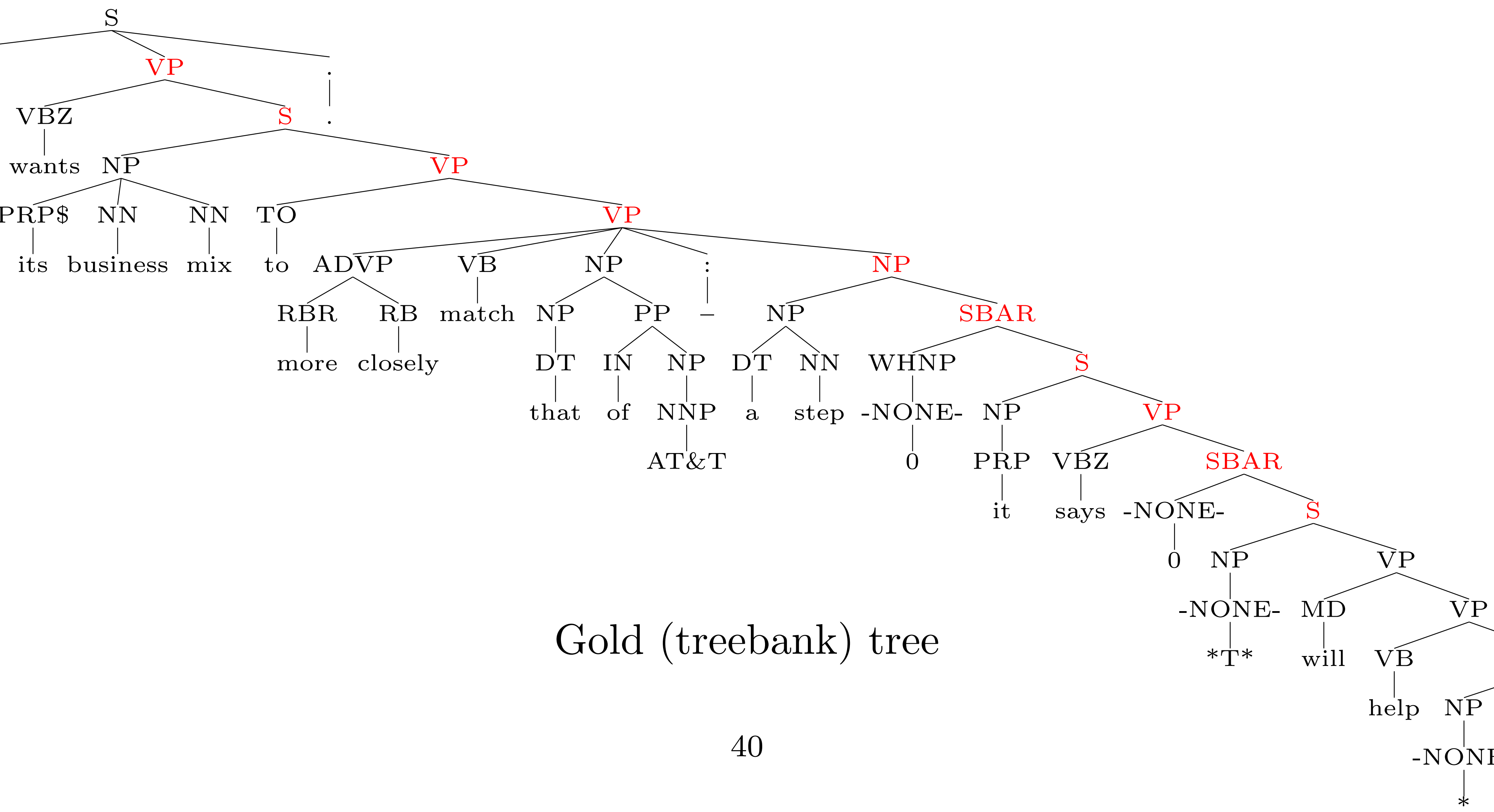
The company wants its business mix to more closely match that of AT&T – a step it says will help prevent cross subsidization.



Parse tree

Multiple attachment errors (2/3)

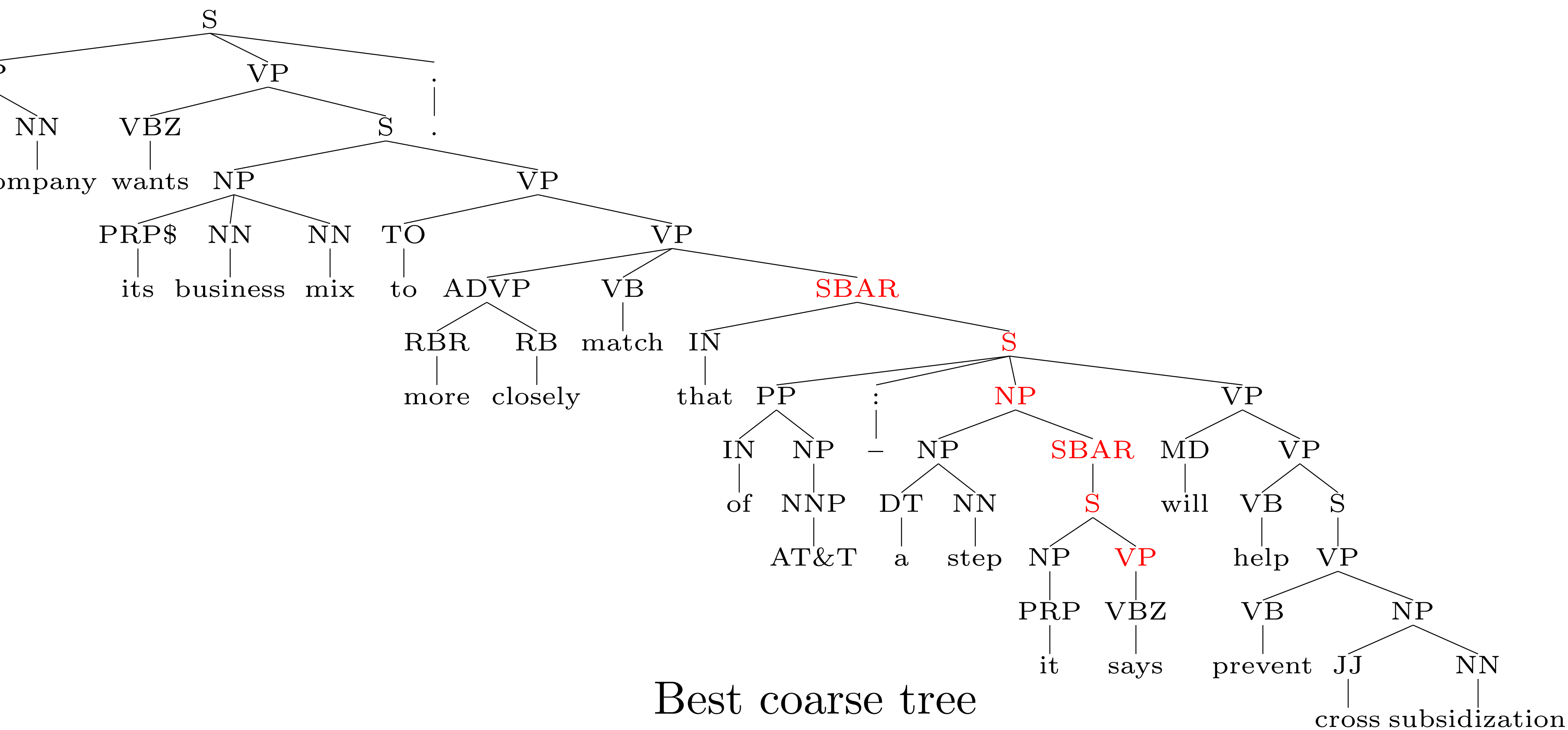
The company wants its business mix to more closely match that of AT&T – a step it says will help prevent cross subsidization.



Gold (treebank) tree

Multiple attachment errors (3/3)

The company wants its business mix to more closely match that of AT&T – a step it says will help prevent cross subsidization.



Best coarse tree

Technical summary

- Generative and discriminative parsers both identify the likely parse y of a string x, e.g., by estimating $\mathrm{P}(y|x)$
- *Generative parsers also define language models*, estimate $\mathrm{P}(x)$
- *Discriminative estimation doesn't require feature independence*
  - suitable for models without tree-structured feature dependencies
- *Parsing is equally complex* for generative and discriminative parsers
  - *depends on features used*
  - *coarse-to-fine* approaches use one parser to narrow the search space for another
- *Estimation* is *computationally inexpensive for generative parsers*, but *expensive for discriminative parsers*
- Because a discriminative parser can use the generative model's probability estimate as a feature, *discriminative parsers almost never do worse* than the generative model, and often do substantially better.

Conclusions

- Discriminatively trained parsing models can perform better than standard generative parsing models
- *Features can be arbitrary functions of parse trees*
  - Non-local features can make a big difference!
  - Difficult to tell which features are most useful
  - Better evaluation (maybe requires real parsing applications?)
- Coarse-to-fine results in (moderately) efficient algorithms
- The parser's errors are often recognizable as certain types of mistakes
  - PP attachment is still a serious issue!

Future directions

- More features (fix those PP attachments!)
- Additional languages (Chinese)
- Richer linguistic representations (WH-dependencies)
- More efficient computational procedures for search and estimation
  - Dynamic programming, approximation methods (variational methods, best-first or beam search)
- Apply discriminative techniques to applications such as speech recognition and machine translation

Discriminative learning in other settings

- Speech recognition
  - Take x to be the acoustic signal, $\mathcal{Y}(x)$ all strings in recognizer lattice for x
  - Training data: $D = ((y_1, x_1), \ldots, (y_n, x_n))$, where y_i is correct transcript for x_i
  - Features could be n-grams, log parser prob, cache features
- Machine translation
  - Take x to be input language string, $\mathcal{Y}(x)$ a set of target language strings (e.g., generated by an IBM-style model)
  - Training data: $D = ((y_1, x_1), \ldots, (y_n, x_n))$, where y_i is correct translation of x_i
  - Features could be n-grams of target language strings, word and phrase correspondences, ...

Regularizer tuning in Max Ent models

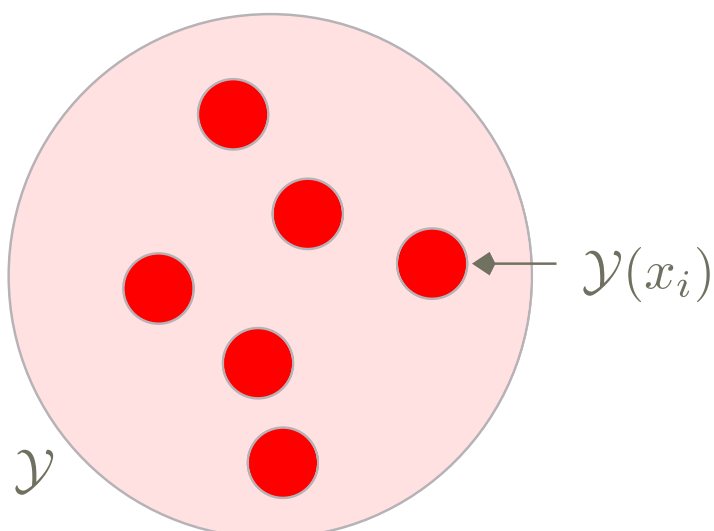
- Associate each feature f_j with bin $b(j)$
- Associate regularizer constant β_k with feature bin k
- Optimize feature weights $\alpha = (\alpha_1, \ldots, \alpha_m)$ on main training data M
- Optimize regularizer constants β on held-out data H

$$
\begin{aligned}
L_D(\alpha) &= \prod_{i=1}^{n} \mathrm{P}_\alpha(y_i|x_i), \text{ where } D = ((y_1, x_1), \ldots, (y_n, x_n)) \\
\hat{\alpha}(\beta) &= \operatorname*{argmax}_{\alpha} \log L_M(\alpha) - \sum_{j=1}^{m} \beta_{b(j)} \alpha_j^2 \\
\hat{\beta} &= \operatorname*{argmax}_{\beta} \log L_H(\hat{\alpha}(\beta))
\end{aligned}
$$

Expectation maximization for PCFGs

- Hidden training data: $D = (x_1, \ldots, x_n)$, where x_i is a string
- The Inside-Outside algorithm is an Expectation-Maximization algorithm for PCFGs

$$\begin{aligned} \hat{p} &= \operatorname*{argmax}_p L_D(p), \text{ where} \\ L_D(p) &= \prod_{i=1}^{n} \mathrm{P}_p(x_i) = \operatorname*{argmax}_p \prod_{i=1}^{n} \sum_{y \in \mathcal{Y}(x_i)} \mathrm{P}(y) \end{aligned}$$



Why there is no conditional ML EM

- Conditional ML conditions on the string x
- Hidden training data: $D = (x_1, \ldots, x_n)$, where x_i is a string
- The likelihood is the probability of predicting the string x_i given the string x_i, a *constant function*

$$\begin{aligned} \hat{p} &= \operatorname*{argmax}_p L_D(p), \text{ where} \\ L_D(p) &= \prod_{i=1}^{n} \mathrm{P}_p(x_i | x_i) \end{aligned}$$