Features of Statistical Parsers

Mark Johnson
Microsoft Research

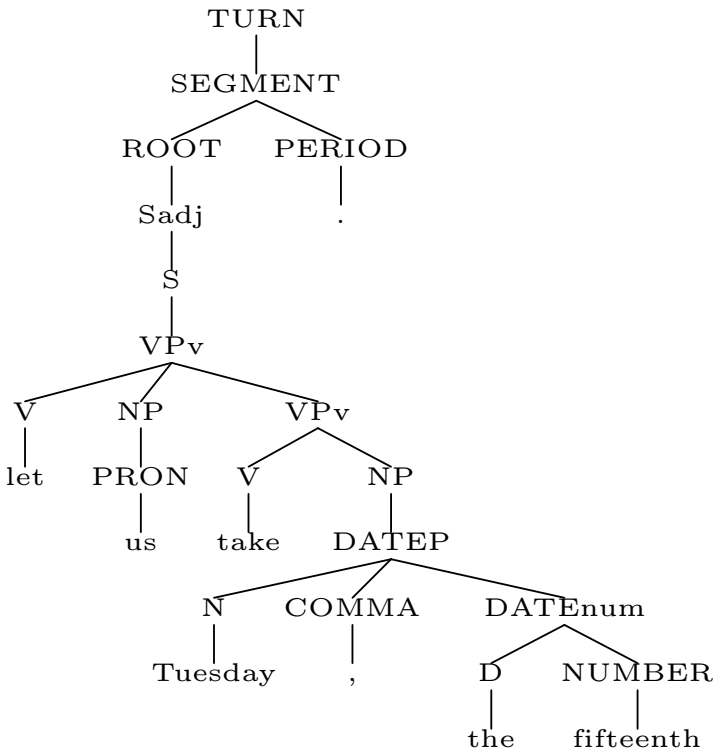
Joint work with
Eugene Charniak, Matt Lease and David McClosky
Brown University

MSR-UW Computational Linguistics Colloquium, October 2006

Talk outline

- Why rerank the output of generative parsers?
- Features of a reranking parser
- Reranking and self-training

LFG parse “Let us take Tuesday the 15th”



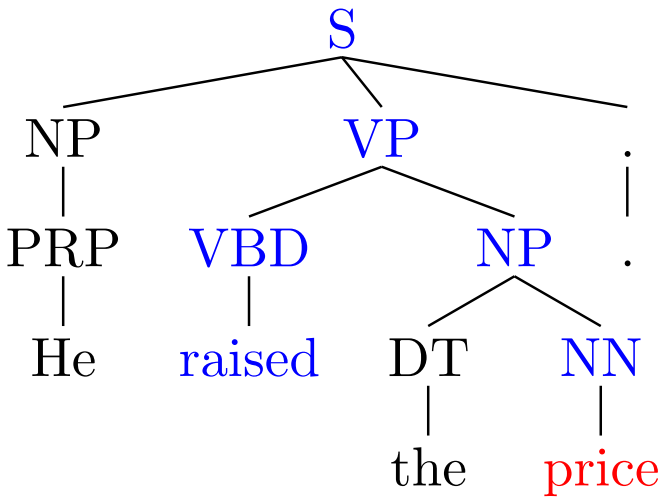
$$
\begin{bmatrix}
\text{SENTENCE\_ID} & \text{BAC002\_E} \\
\left\{ \begin{bmatrix}
\text{OBJ}_{9} & \begin{bmatrix} \text{ANIM} & + \\ \text{CASE} & \text{ACC} \\ \text{NUM} & \text{PL} \\ \text{PERS} & 1 \\ \text{PRED} & \text{PRO} \\ \text{PRON-FORM} & \text{WE} \\ \text{PRON-TYPE} & \text{PERS} \end{bmatrix} \\
\text{PASSIVE} & - \\
\text{PRED} & \text{LET}\langle 2,10 \rangle 9 \\
\text{STMT-TYPE} & \text{IMPERATIVE} \\
\text{SUBJ}_{2} & \begin{bmatrix} \text{PERS} & 2 \\ \text{PRED} & \text{PRO} \\ \text{PRON-TYPE} & \text{NULL} \end{bmatrix} \\
\text{TNS-ASP} & \begin{bmatrix} \text{MOOD} & \text{IMPERATIVE} \end{bmatrix} \\
\text{XCOMP}_{10} & \begin{bmatrix}
\text{OBJ} & \begin{bmatrix} \text{ANIM} & - \\ \text{APP} & \begin{bmatrix} \text{NTYPE} & \begin{bmatrix} \text{NUMBER} & \text{ORD} \\ \text{TIME} & \text{DATE} \end{bmatrix} \\ \text{NUM} & \text{SG} \\ \text{PRED} & \text{fifteen} \\ \text{SPEC} & \begin{bmatrix} \text{SPEC-FORM} & \text{THE} \\ \text{SPEC-TYPE} & \text{DEF} \end{bmatrix} \end{bmatrix} \\ \text{CASE} & \text{ACC} \\ \text{GEND} & \text{NEUT} \\ \text{NTYPE} & \begin{bmatrix} \text{GRAIN} & \text{COUNT} \\ \text{PROPER} & \text{DATE} \\ \text{TIME} & \text{DAY} \end{bmatrix} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \\ \text{PRED} & \text{TUESDAY} \end{bmatrix}_{13} \\
\text{PASSIVE} & - \\
\text{PRED} & \text{TAKE}\langle 9,13 \rangle \\
\text{SUBJ} & \boxed{9}
\end{bmatrix}
\end{bmatrix} \right\}
\end{bmatrix}
$$

Parsing in the late 1990s

- Parsers for hand-written grammars (LFG, HPSG, etc)
  - linguistically rich, detailed representations
  - uneven / poor coverage of English
  - even simple sentences are highly ambiguous
  - only ad hoc treatment of preferences
  - could not be learnt from data
- Generative probabilistic parsers
  - systematic treatment of preferences
  - *learnt from treebank corpora*
  - simple constituent structure representations
  - wide (if superficial) coverage of English
- *Could the two approaches be combined?*

Generative statistical parsers

- Generative statistical parsers (Bikel, Charniak, Collins) generate each new node in parse conditioned on the structure already generated: P(price|NN, NP, raised, VBD, VP, S)

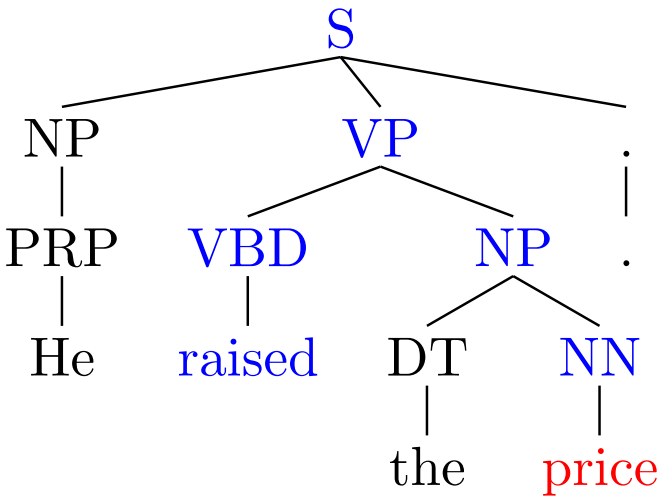


- They assume each node is independent of all existing structure except for nodes explicitly conditioned on ⇒ simple "relative frequency" estimators (smoothed$^{\diamond\diamond}$)
- Re-entrancies in LFG and HPSG *violate* these independence assumptions

Abandoning independence assumptions

- Mathematically straight-forward to define models in which nodes are not assumed independent (Abney 1997)
  - Maximum Entropy, log-linear, exponential, Gibbs, ...
- Once we have abandoned feature independence,
  - parses need not be trees
    - feature structures, minimalist derivations, ...
  - features can be *any computable function* of parses
- But simple "relative frequency" estimators no longer work
  - *estimating grammar from a corpus is a computationally very difficult problem*

Generative parsers as log-linear models



- Define a feature $f_{x,c}$ for all possible nodes x and conditioning contexts c

  $f_{(\text{price,NN,NP,raised,VBD,VP,S})}(t)$ is the number of times $(\text{price}, \text{NN}, \text{NP}, \text{raised}, \text{VBD}, \text{VP}, \text{S})$ appears in parse t

- Let weight $w_{x,c} = \log \text{P}(x|c)$

  $w_{(\text{price,NN,NP,raised,VBD,VP,S})} =$
  $\log \text{P}(\text{price}|\text{NN}, \text{NP}, \text{raised}, \text{VBD}, \text{VP}, \text{S})$

- Then weighted sum of features is log probability of parse

Conditional estimation

- Maximum likelihood joint estimation (used in generative parsers) adjusts weights to make corpus parses score higher than all other parses
- Without independence assumptions, requires summing over all possible parses of *all possible sentences* (partition function)

  $\Rightarrow$ estimation is computationally intractible$^{\diamond\diamond}$
- But for parsing we only need *conditional distribution* $\mathrm{P}(t|s)$ of parses given strings
  - "only" requires parses for strings in training corpus

  $\Rightarrow$ computationally tractable

Conditional estimation

s	$f(\hat{t}(s))$	feature vectors of other parses for s
sentence 1	$(1, 3, 2)$	$(2, 2, 3)$ $(3, 1, 5)$ $(2, 6, 3)$
sentence 2	$(7, 2, 1)$	$(2, 5, 5)$
sentence 3	$(2, 4, 2)$	$(1, 1, 7)$ $(7, 2, 1)$
...	...	...

- Treebank tells us correct parse $\hat{t}(s)$ for sentence s
- Parser produces all possible parses for each sentence s
- Adjust feature weights $w = (w_1, \ldots, w_m)$ to make $\hat{t}(s)$ score as high as possible relative to other parses for s

Conditional vs joint estimation

$$\mathrm{P}(t, s) \;=\; \mathrm{P}(t|s)\mathrm{P}(s)$$

- Joint MLE maximizes probability of training trees t *and strings* s
- Conditional MLE maximizes probability of trees given strings
  - Conditional estimation uses less information from the data
  - learns nothing from distribution of strings $\mathrm{P}(s)$
  - ignores unambiguous sentences (!)
- Joint estimation should be better (lower variance) if your model correctly relates $\mathrm{P}(t|s)$ and $\mathrm{P}(s)$
- Conditional estimation should be better if your model incorrectly relates $\mathrm{P}(t|s)$ and $\mathrm{P}(s)$

Linguistic representations and features

- Probability of a parse t is completely determined by its feature vector $(f_1(t), \ldots, f_m(t))$
- The actual linguistic representation of parse t is *irrelevant* as long as it is rich enough to calculate features $f(t)$
- Feature functions define the kinds of generalizations that the learner can extract
  - parses with the same feature values will be assigned the same probability
  - the choice of feature functions is as much a linguistic decision than the choice of representations
- Features can be arbitrary functions $\Rightarrow$ the linguistic properties they encode *need not be directly represented in the parse*

Reranking a generative parser's parses

- Parses only need to be rich enough to recover the features
  - WH-movement, raising and control need not be explicitly marked in parses, just so long as we can identify them if required
- LFG and similar parsers have problems with coverage and implementation
- Generative parsers are reliable, and their parses are rich enough to identify many linguistically interesting features

$\Rightarrow$ *Why not work with a generative parser's output instead?* (Collins 2000)

Talk outline

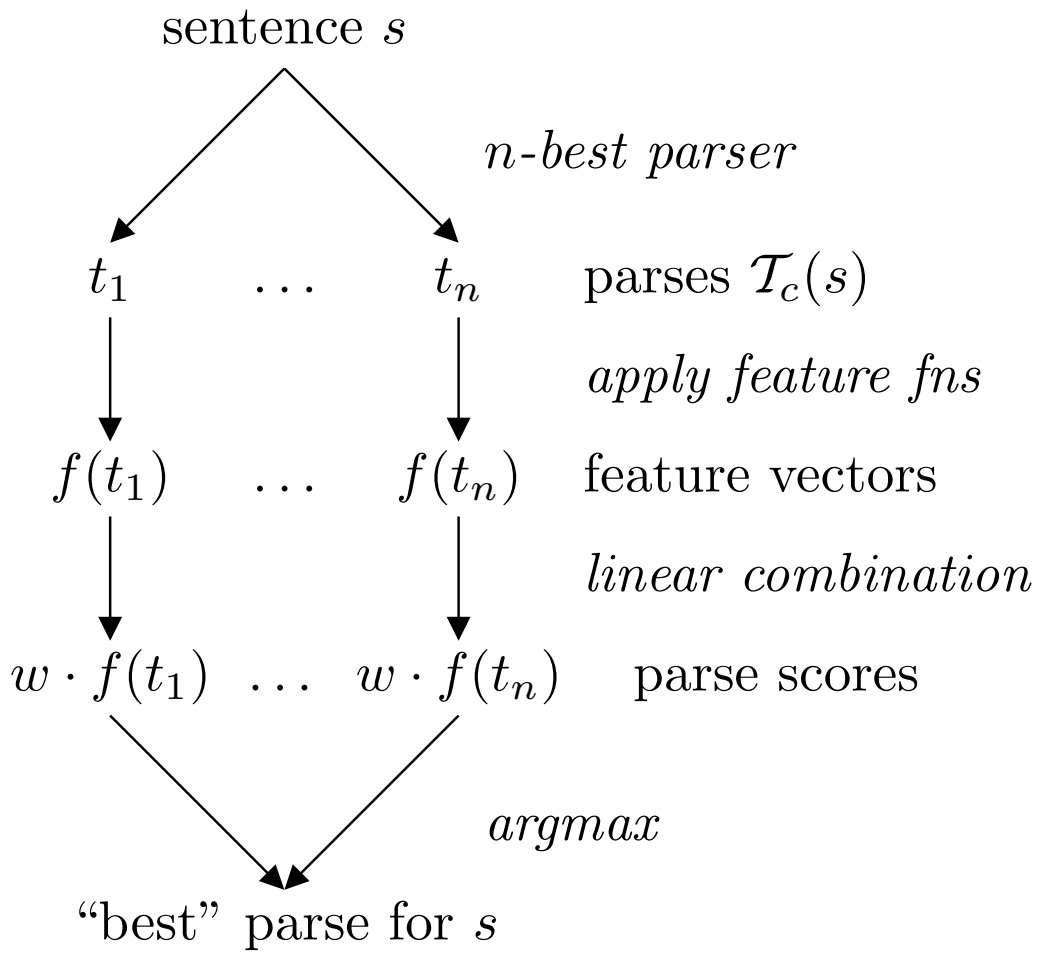
- Why rerank the output of generative parsers?
- Features of a reranking parser
- Reranking and self-training

Linear reranking framework

- Generative parser produces n *candidate parses* $\mathcal{T}_c(s)$ for each sentence s
- Map each parse $t \in \mathcal{T}_c(s)$ to a real-valued *feature vector* $f(t) = (f_1(t), \ldots, f_m(t))$
- Each feature f_j is associated with a *weight* w_j
- The *highest scoring* parse

$$\hat{t} = \underset{t \in \mathcal{T}_c(s)}{\operatorname{argmax}}\, w \cdot f(t)$$

is predicted correct



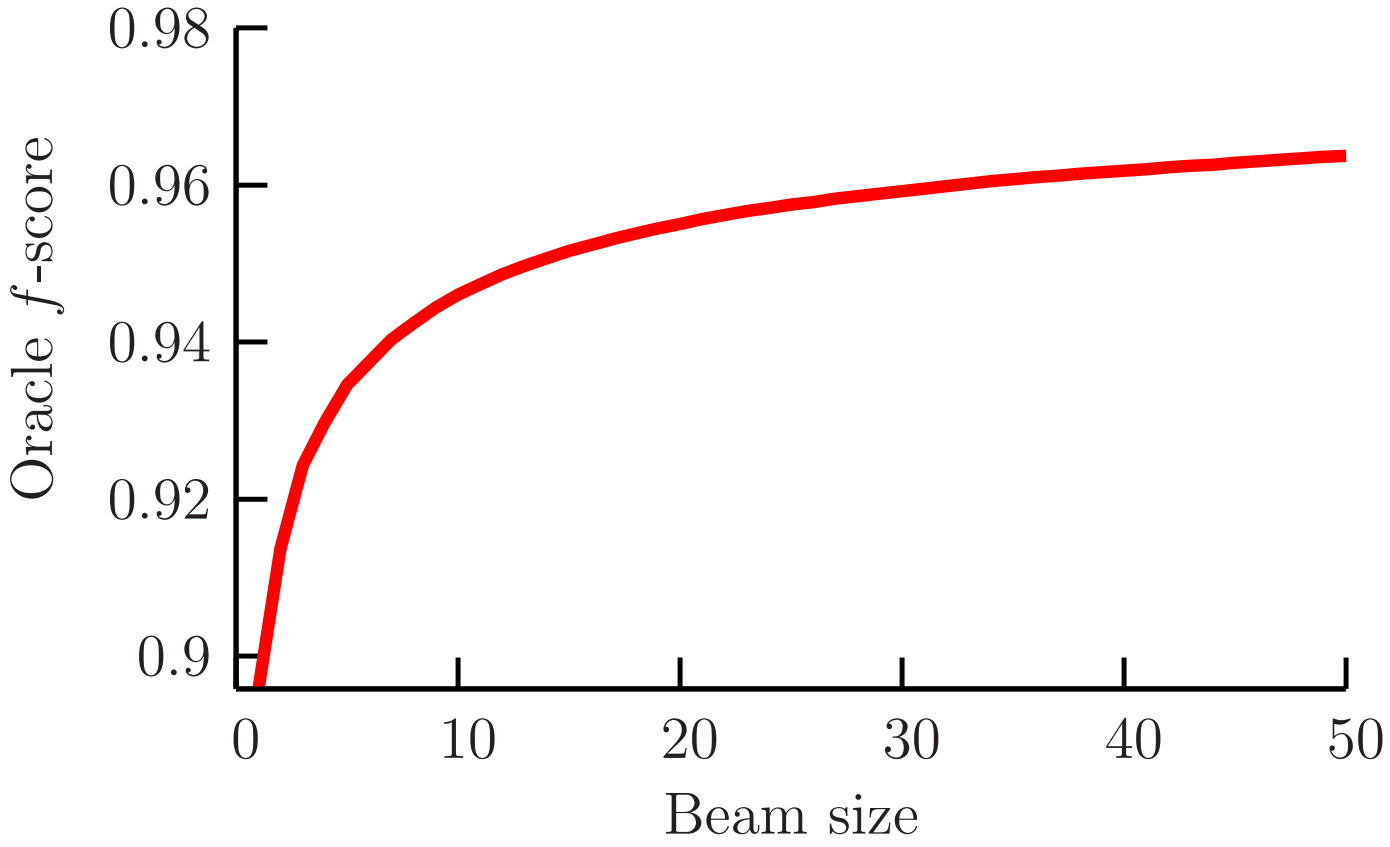
Features for ranking parses

- Features can be any real-valued function of parse trees
- In these experiments the features come in two kinds:
  - The logarithm of the tree's probability estimated by the Charniak parser
  - The number of times a particular configuration appears in the parse
- *Which ones improve parsing accuracy the most?* (can you guess?)

Experimental setup

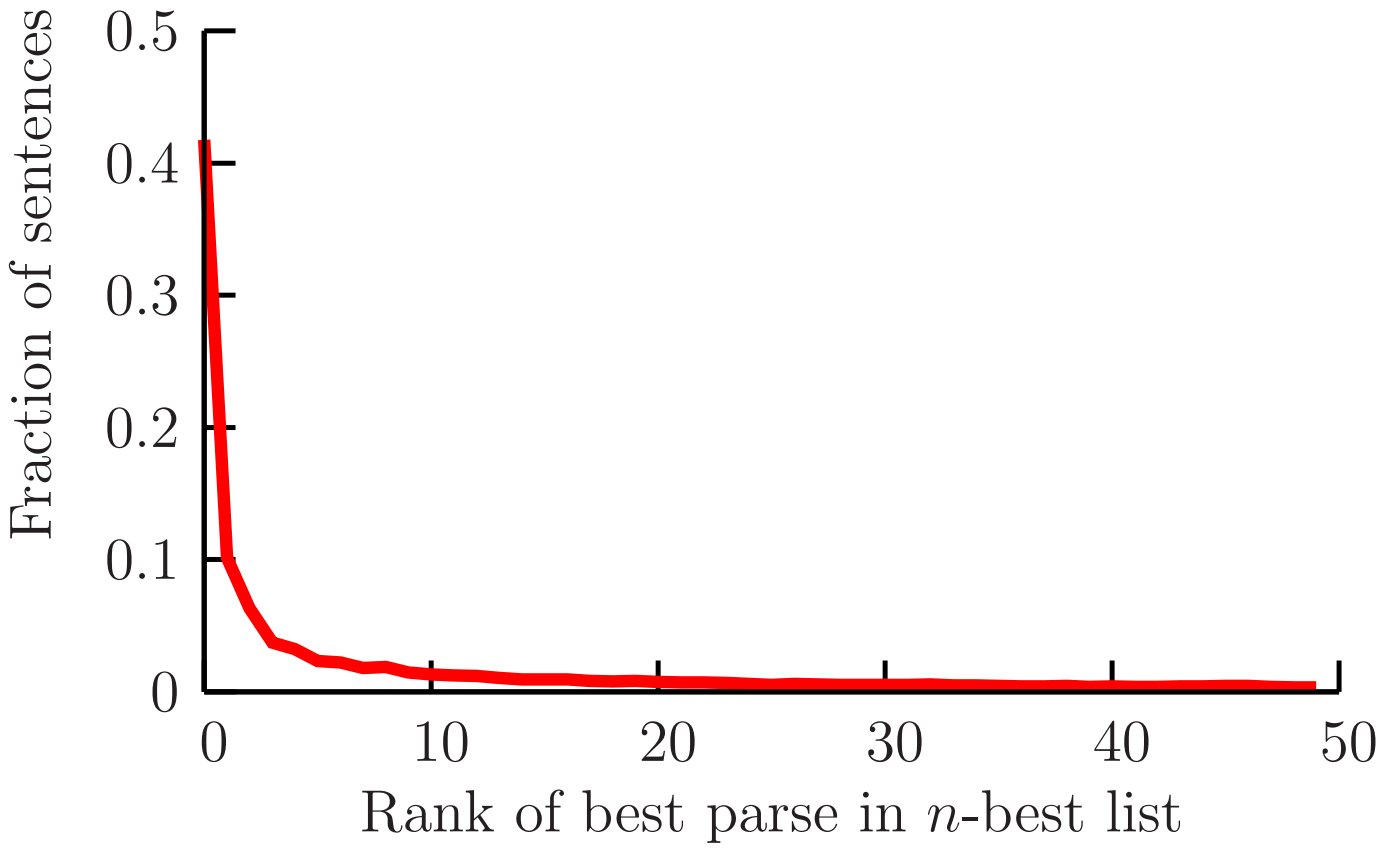
- Feature tuning experiments done using Collins' split: sections 2-19 as train, 20-21 as dev and 22 as test
- $\mathcal{T}_c(s)$ computed using Charniak 50-best parser
- Features which vary on less than 5 sentences pruned
- Optimization performed using LMVM optimizer from Petsc/TAO optimization package
- Regularizer constant c adjusted to maximize f-score on dev

f-score vs. n-best beam size



- F-score of Charniak's most probable parse = 0.896
- Oracle f-score (f-score of best parse in beam) of Charniak's 50-best parses = 0.965 (66% redn)

Rank of best parse



- Charniak parser's most likely parse is the best parse 41% of the time
- Reranker picks Charniak parser's most likely parse 58% of the time

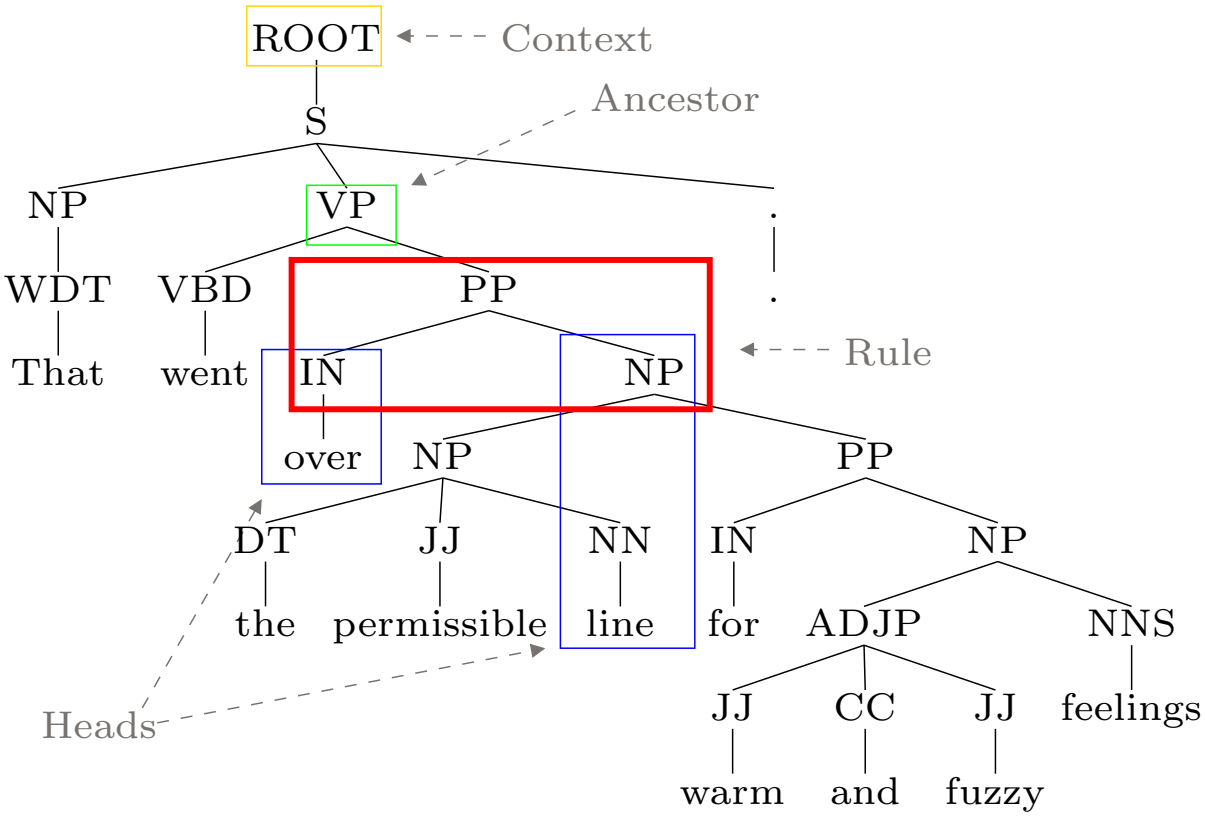
Evaluating features

- The feature weights are not that indicative of how important a feature is
- The MaxEnt ranker with regularizer tuning takes approx 1 day to train
- The *averaged perceptron* algorithm takes approximately 2 minutes

$\Rightarrow$ used in feature-comparison experiments here

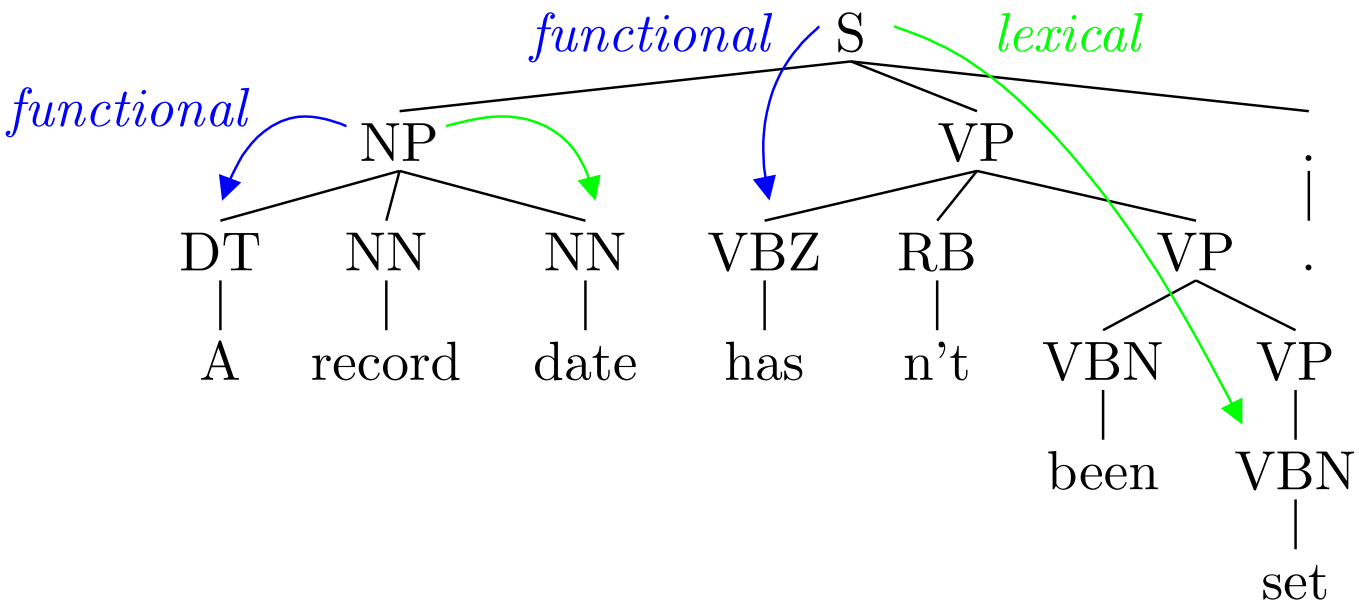
Lexicalized and parent-annotated rules

- Rule features largely replicate features already in generative parser
- A typical Rule feature might be (PP IN NP)



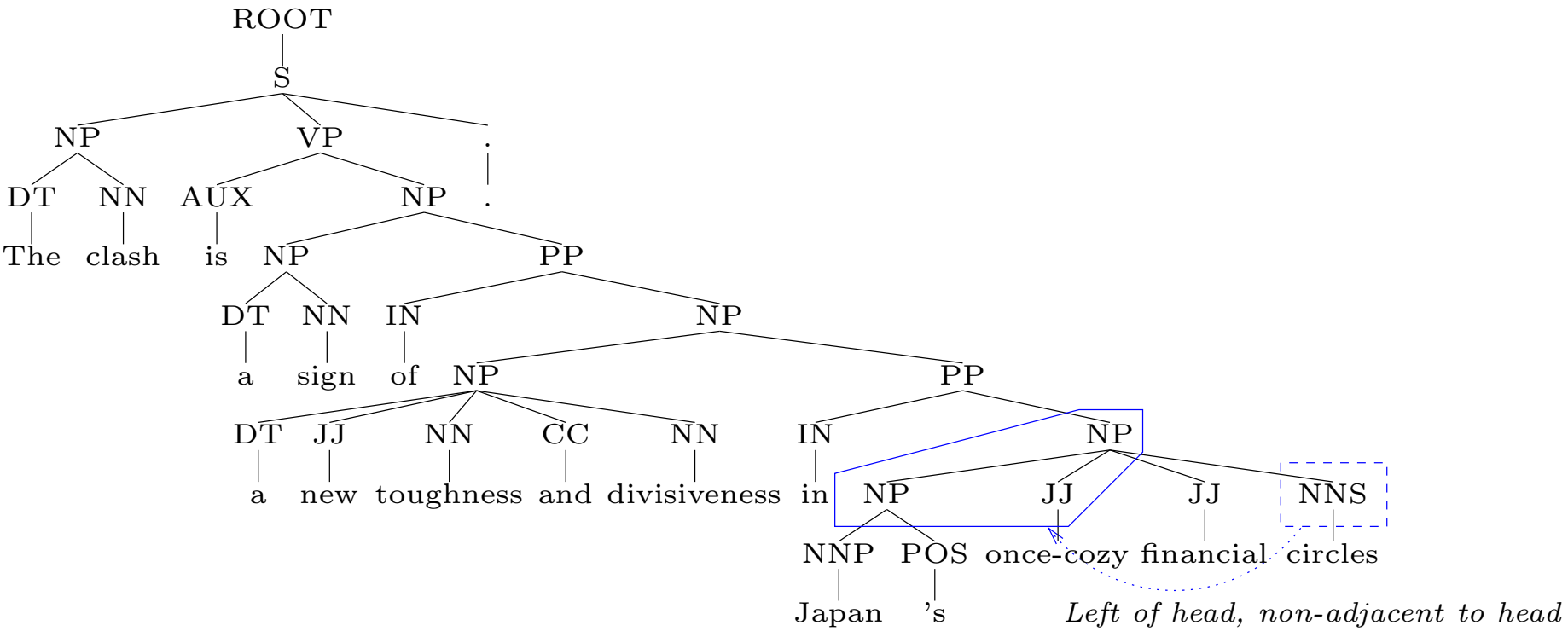
Functional and lexical heads

- There are at least two sensible notions of head (c.f., Grimshaw)
  - *Functional heads:* determiners of NPs, auxilary verbs of VPs, etc.
  - *Lexical heads:* rightmost Ns of NPs, main verbs in VPs, etc.
- In a log-linear model, it is easy to use both!



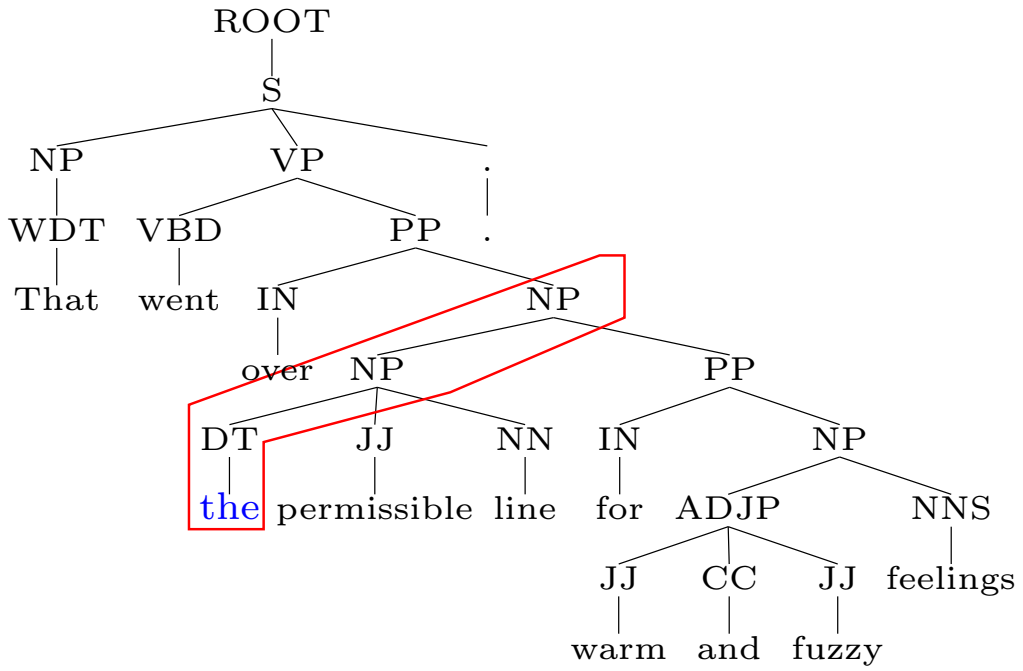
n-gram rule features generalize rules

- Breaks up long treebank constituents into shorter (phrase-like?) chunks
- Also includes *relationship to head* (e.g., adjacent? left or right?)



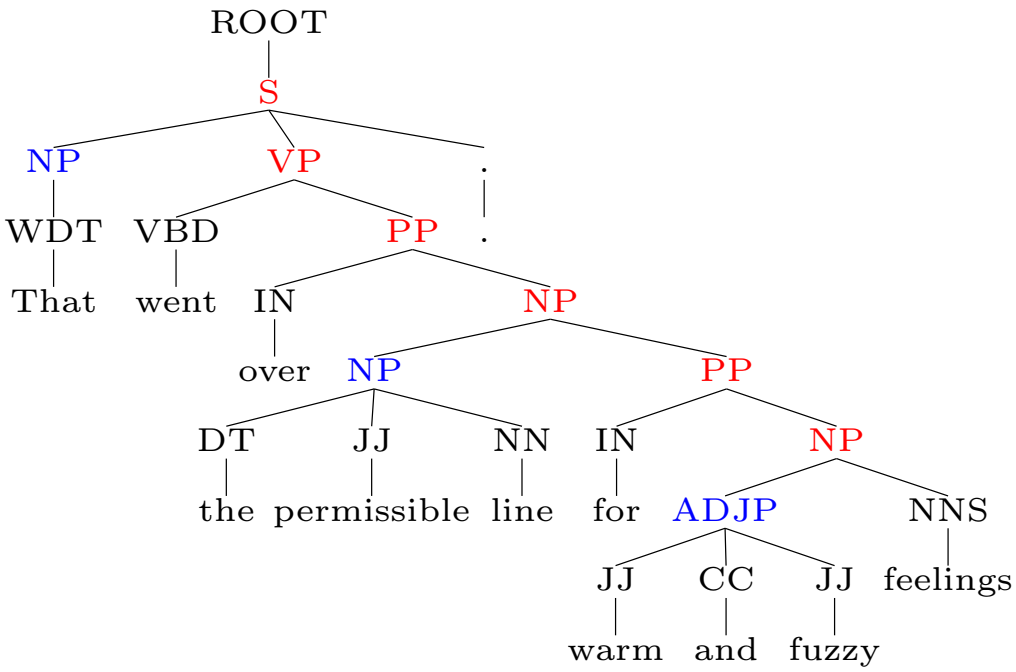
Word and WProj features

- A Word feature is a word plus n of its parents (c.f., Klein and Manning's non-lexicalized PCFG)
- A WProj feature is a word plus all of its (maximal projection) parents, up to its governor's maximal projection



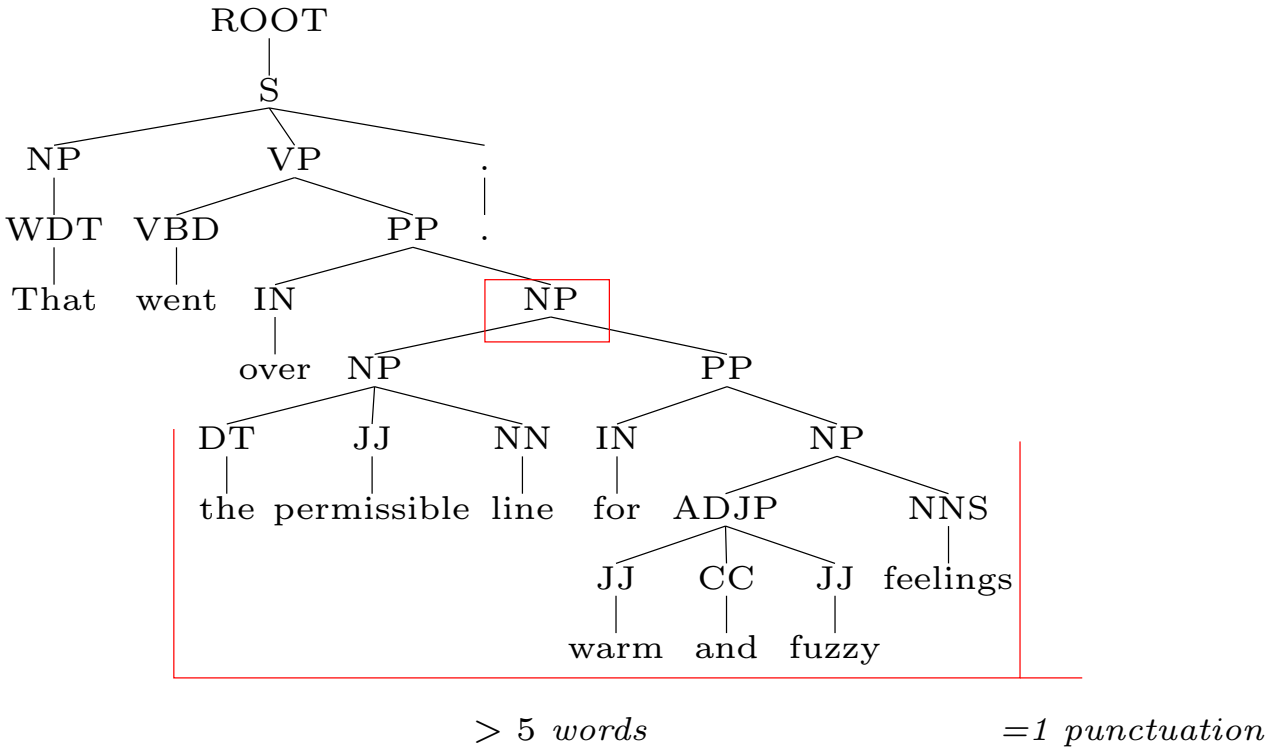
Rightmost branch bias

- The RightBranch feature's value is the number of nodes on the right-most branch (ignoring punctuation) (c.f., Charniak 00)
- Reflects the tendancy toward right branching in English
- Only 2 different features, but very useful in final model!



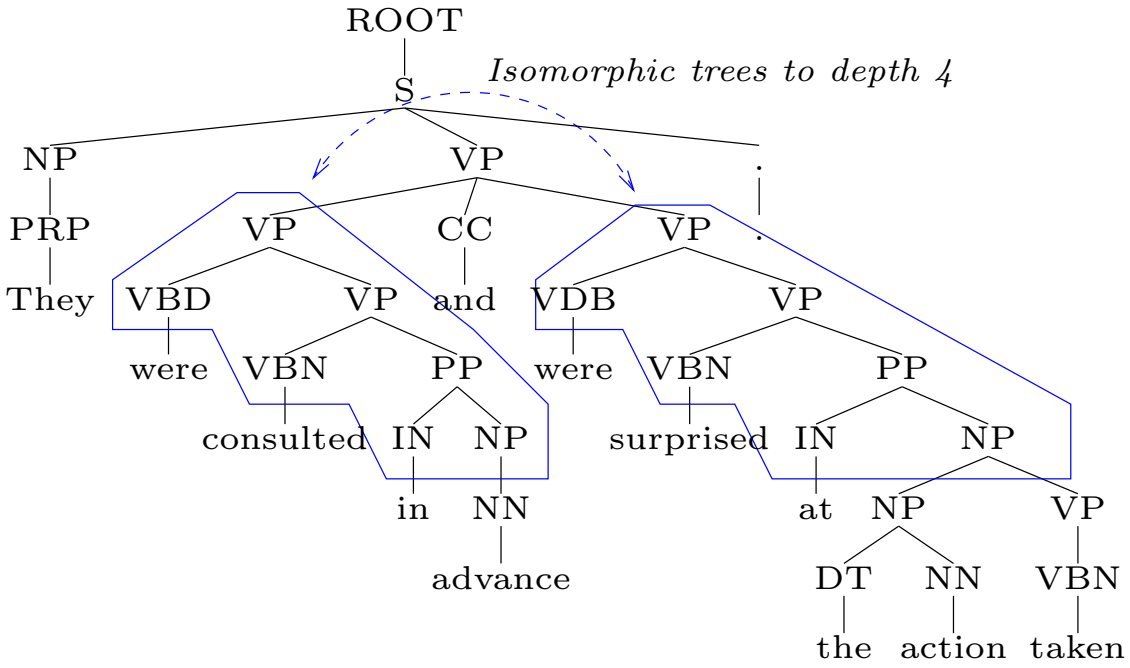
Constituent Heavyness and location

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



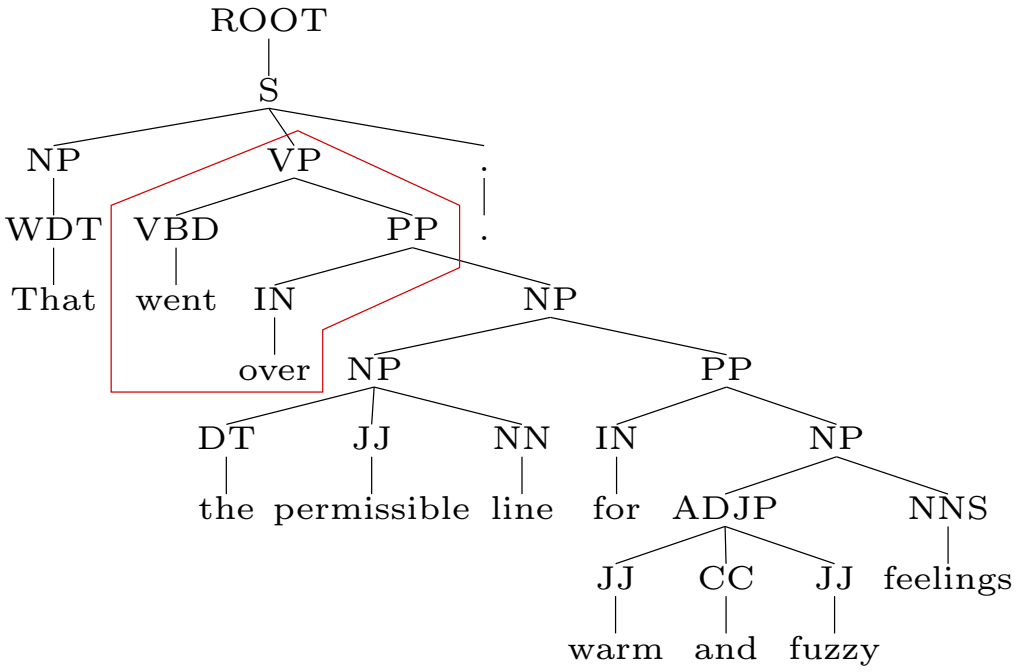
Coordination parallelism

- A CoPar feature indicates the depth to which adjacent conjuncts are parallel



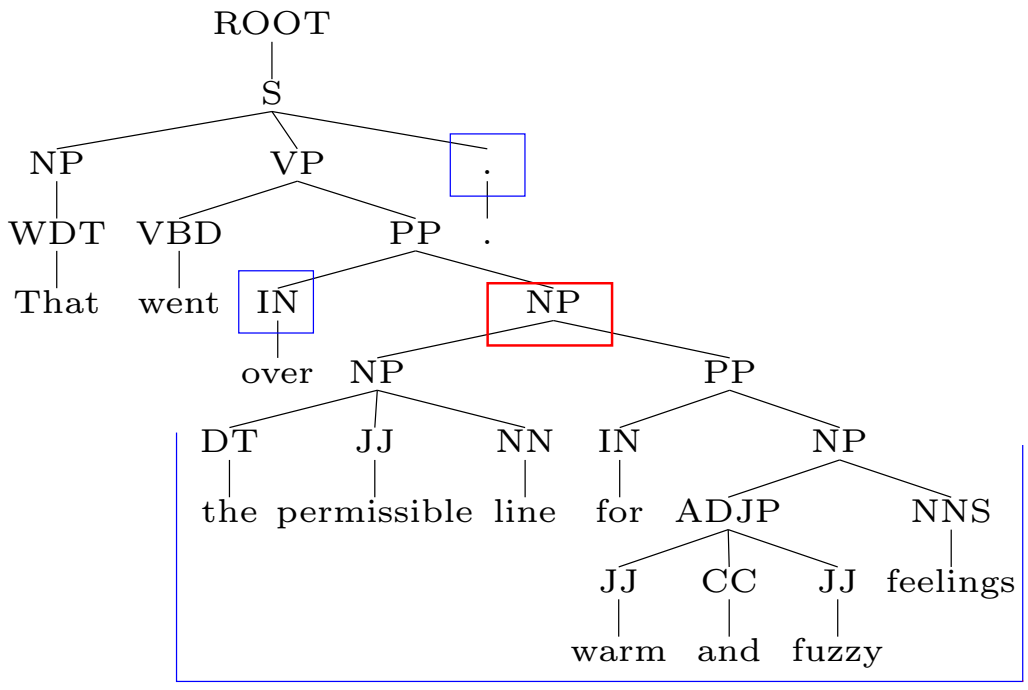
Tree n-gram

- A tree n-gram feature is a tree fragment that connect sequences of adjacent n words, for $n = 2, 3, 4$ (c.f. Bod's DOP models)
- lexicalized and non-lexicalized variants



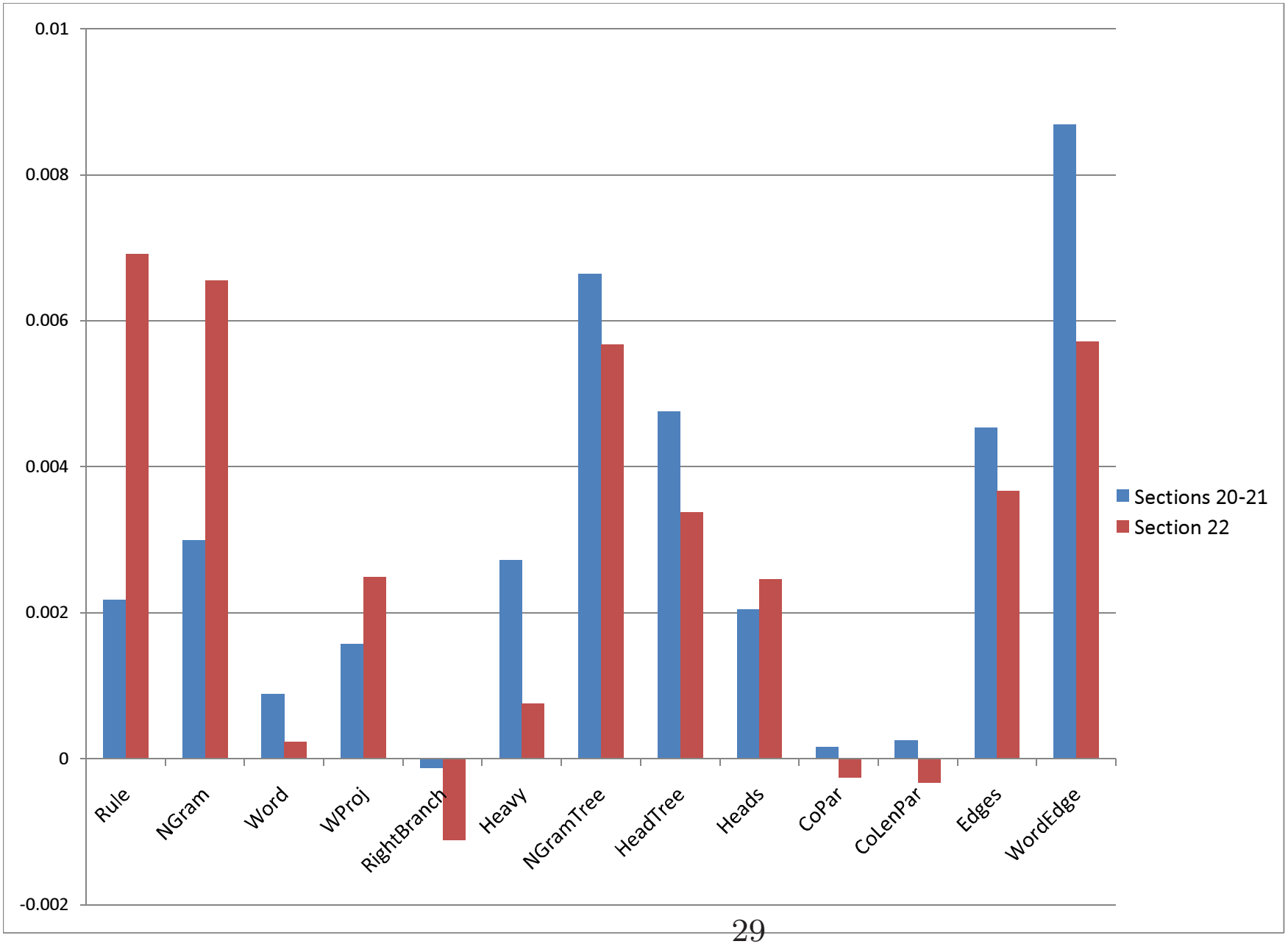
Edges and WordEdges

- A Neighbours feature indicates the node's category, its binned length and j left and k right lexical items and/or POS tags for $j, k \leq 2$

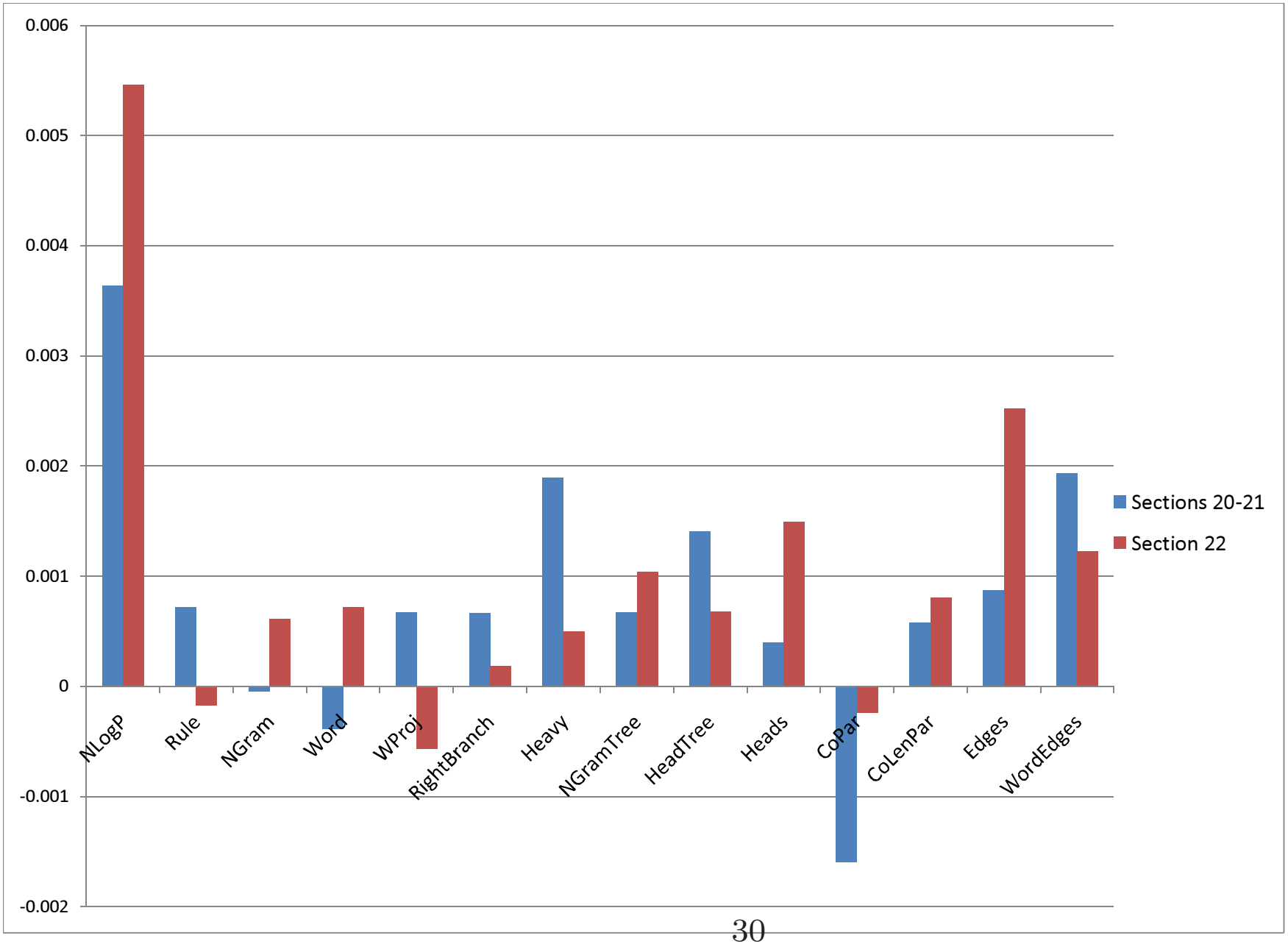


> 5 *words*

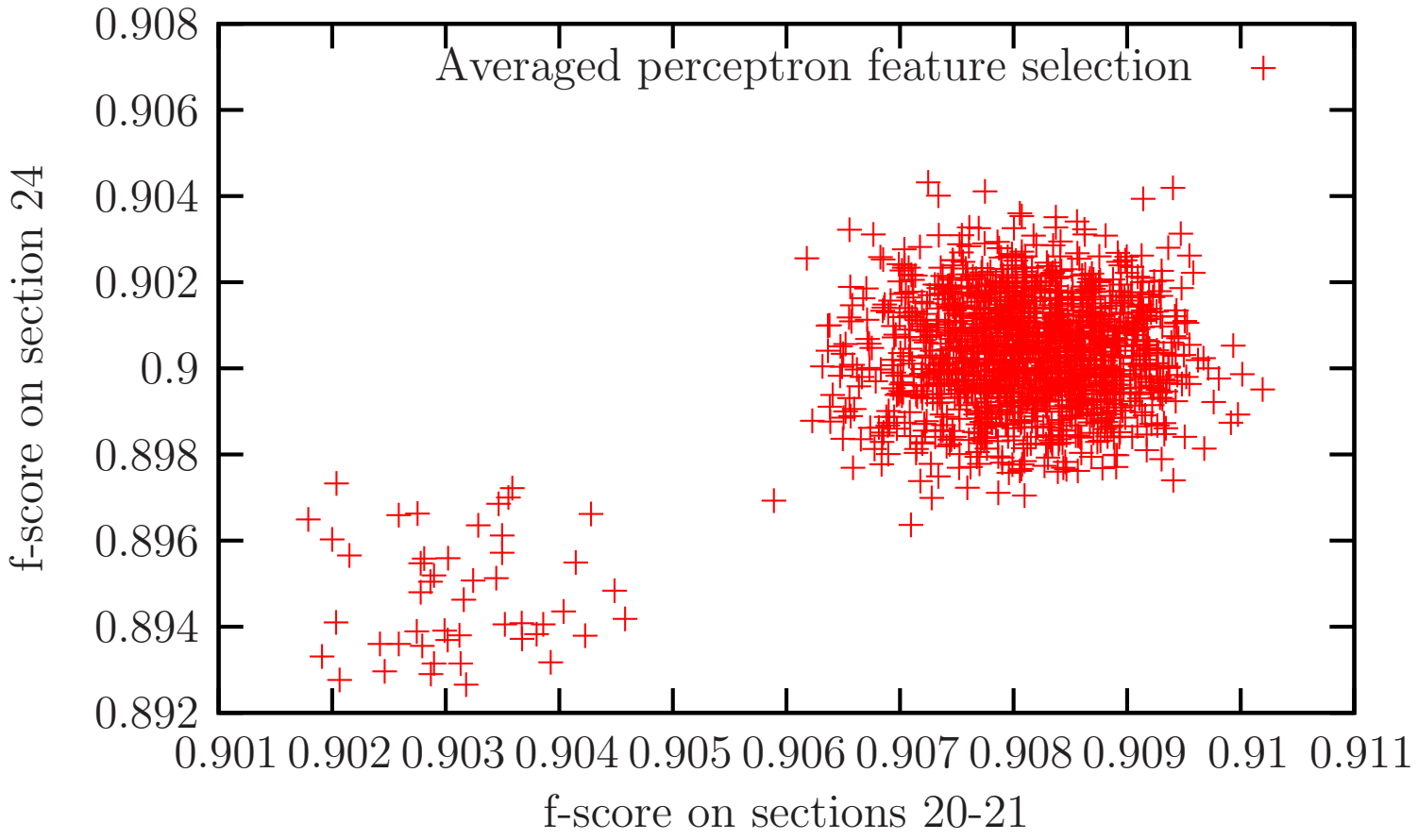
Adding one feature class to baseline parser



Removing one feature class from reranker



Feature selection is hard



- Greedy feature selection using *averaged perceptron* optimizing f-score on sec 20–21
- All models also evaluated on section 24

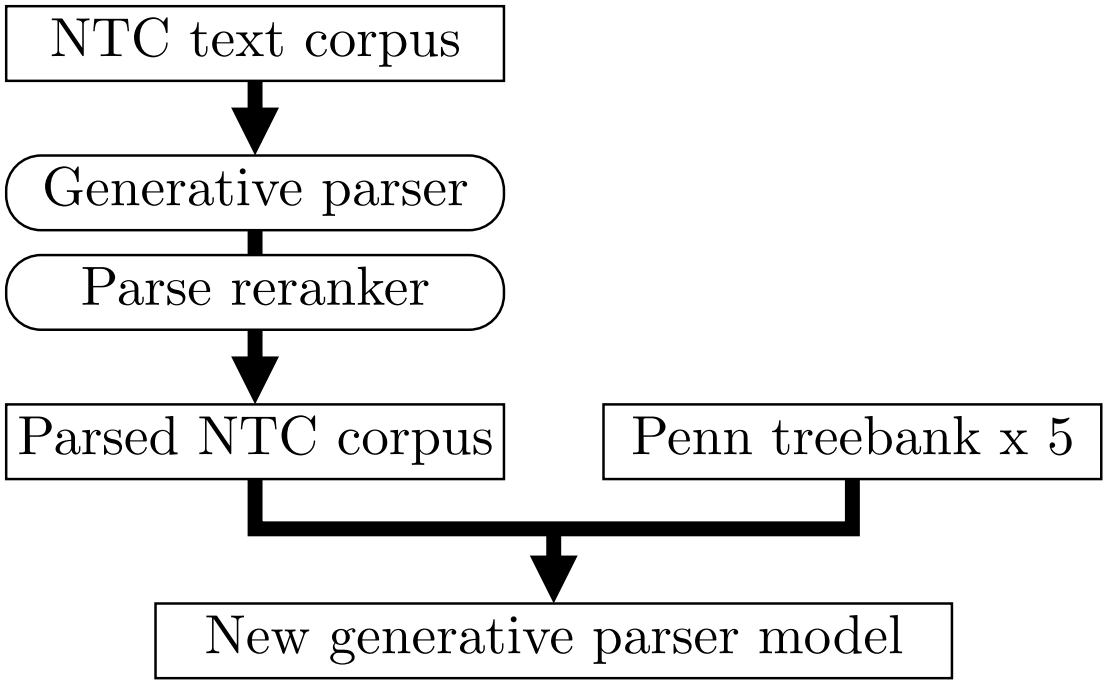
Results on all training data

- Features must vary on parses of at least 5 sentences in training data
- In this experiment, 1,333,863 features
- Exponential model trained on sections 2-21
- Gaussian regularization $p = 2$, constant selected to optimize f-score on section 22
- On section 23: recall = 91.0, precision = 91.8, f-score = 91.4
- Available from www.cog.brown.edu

Talk outline

- Why rerank the output of generative parsers?
- Features of a reranking parser
- Reranking and self-training

Self-training for discriminative parsing



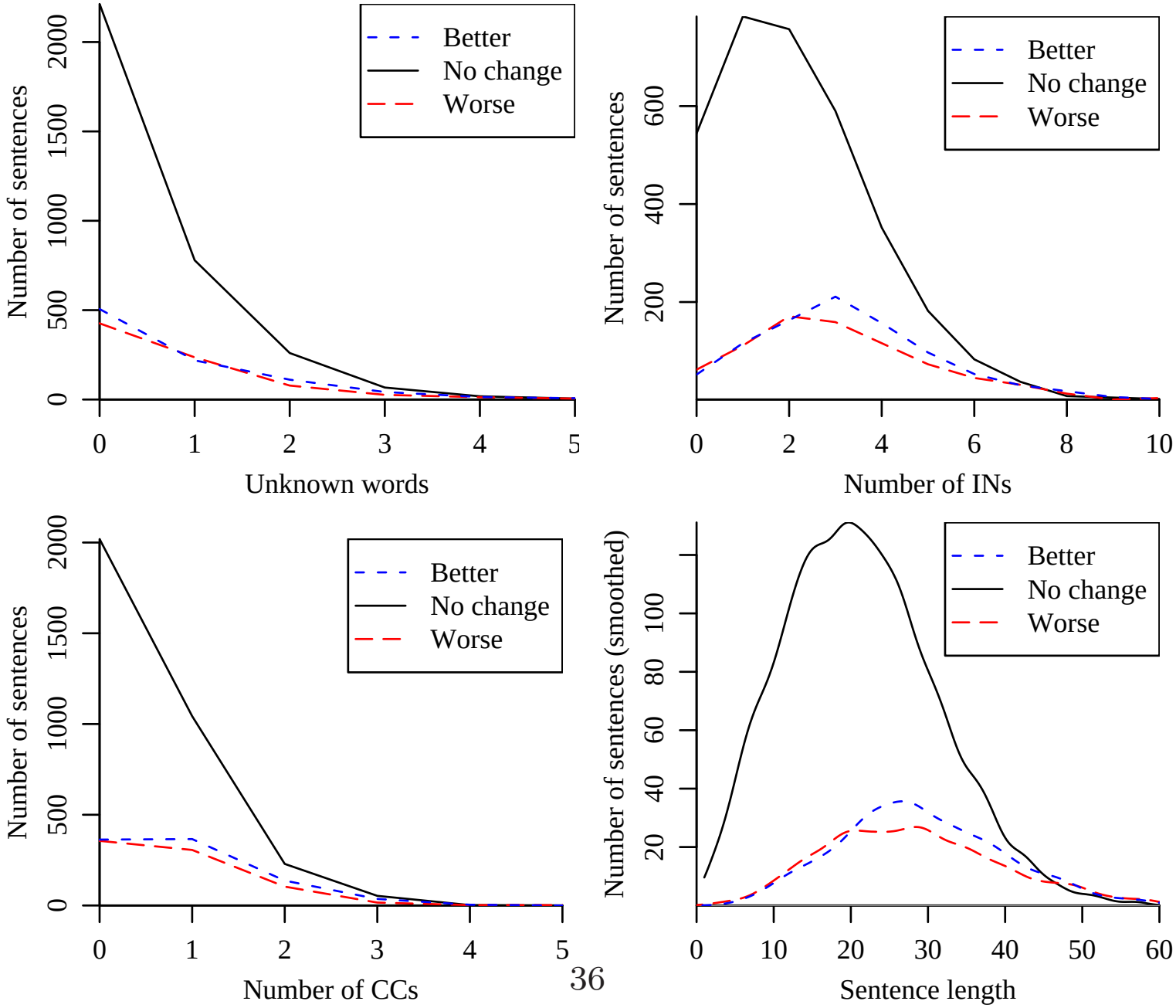
- *Improves performance* from 91.3 to **92.1 f-score**
- Self-training without the reranker does not improve performance
- Retraining the reranker on new first-stage model does not further improve performance[◇◇]
- Would reparsing the NTC with improved parser further improve performance?

First-stage oracle scores

Model	1-best	10-best	50-best
Baseline	89.0	94.0	95.9
WSJ×1 + 250k	89.8	94.6	96.2
WSJ×5 + 1,750k	90.4	94.8	96.4

- Self-training improves first-stage generative parser's oracle scores
- First-stage parser also became more decisive: mean of $\log_2$(P(1-best) / P(50th-best)) increased from 11.959 for the baseline parser to 14.104 for self-trained parser

Which sentences improve?



Self-trained WSJ parser on Brown

Sentences added	Parser	WSJ-reranker
Baseline Brown	86.4	87.4
Baseline WSJ	83.9	85.8
WSJ+50k	84.8	86.6
WSJ+250k	85.7	87.2
WSJ+1,000k	86.2	87.3
WSJ+2,500k	86.4	87.7

- Adding NTC data greatly improves performance on Brown corpus (to a lesser extent on Switchboard)

Self-training vs in-domain training

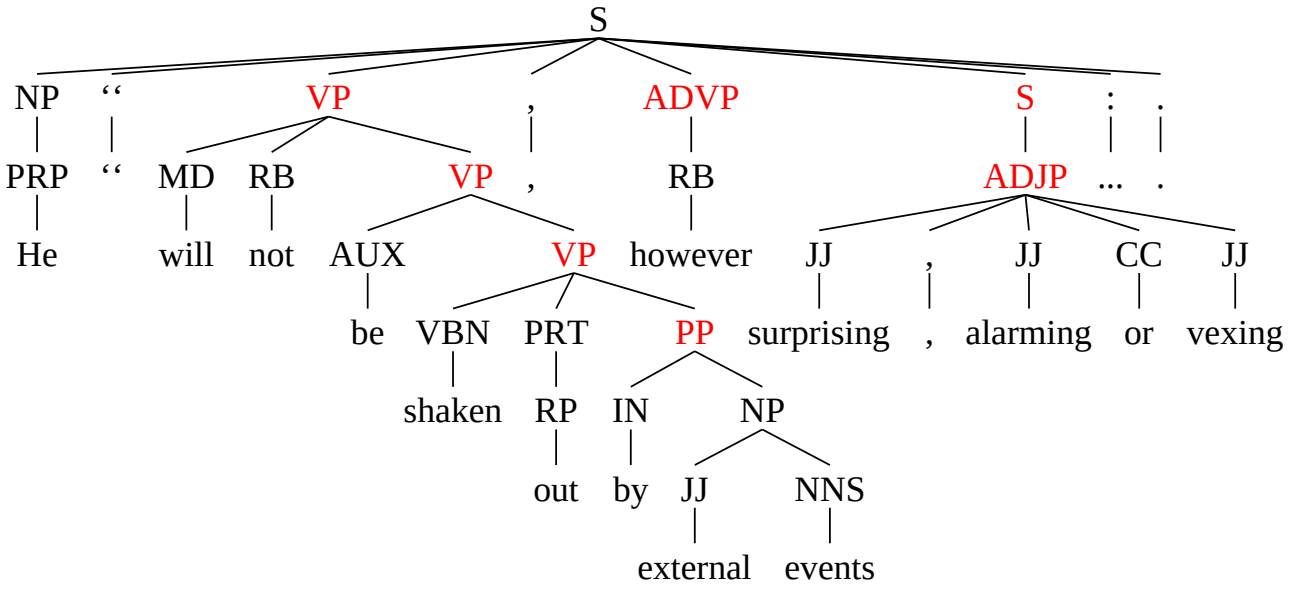
First-stage	First stage alone	WSJ-reranker	Brown-reranker
WSJ	82.9	85.2	85.2
WSJ+NTC	87.1	87.8	87.9
Brown	86.7	88.2	88.4

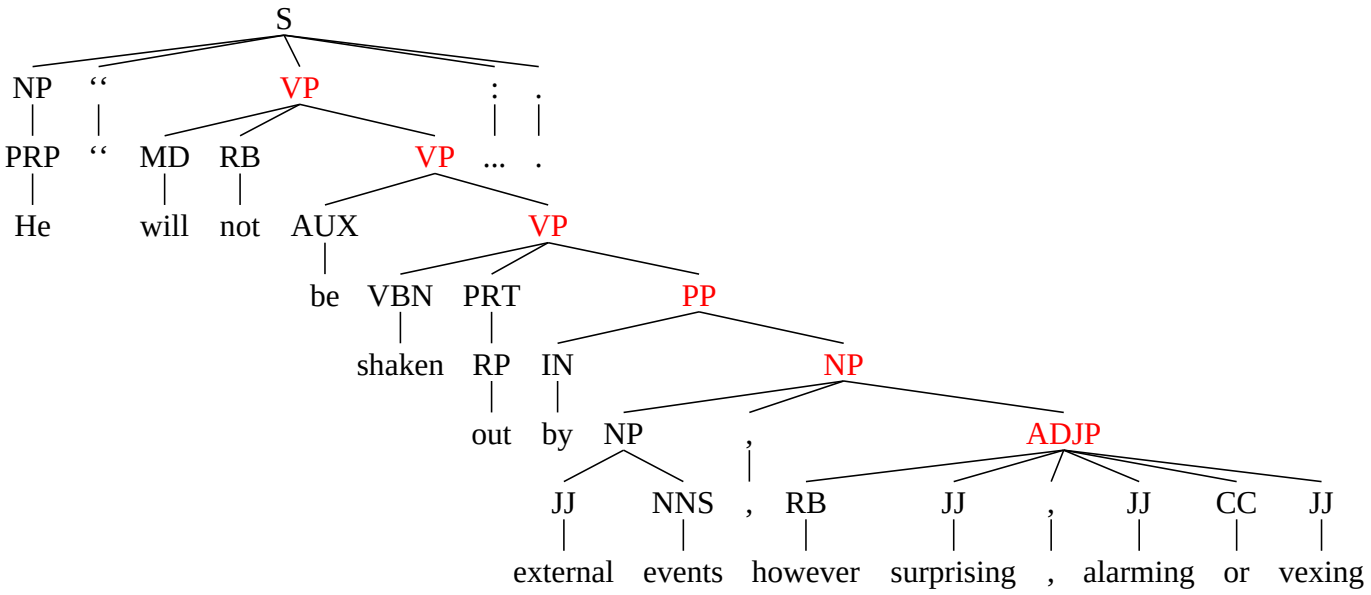
- Both reranking and self-training are surprisingly domain-independent
- Self-trained NTC parser with WSJ reranker is almost as good as a parser/reranker completely trained on Brown$^{\diamond\diamond}$

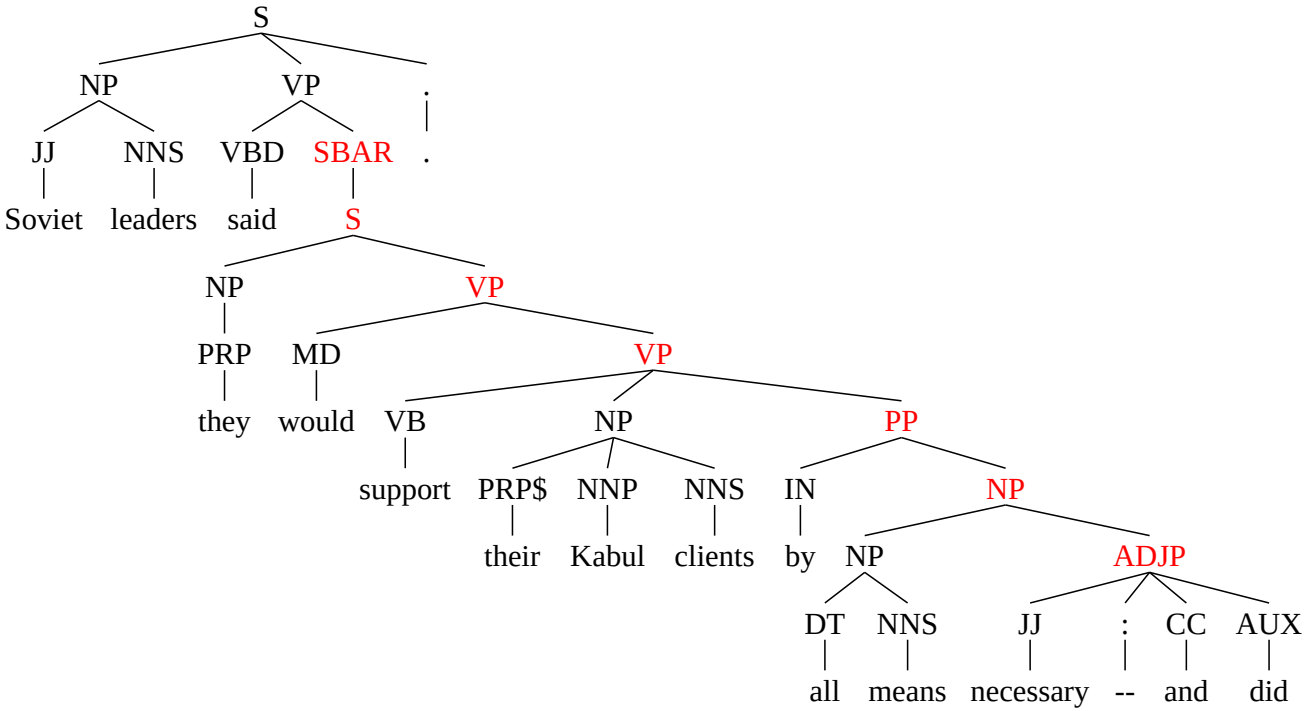
Summary and conclusions

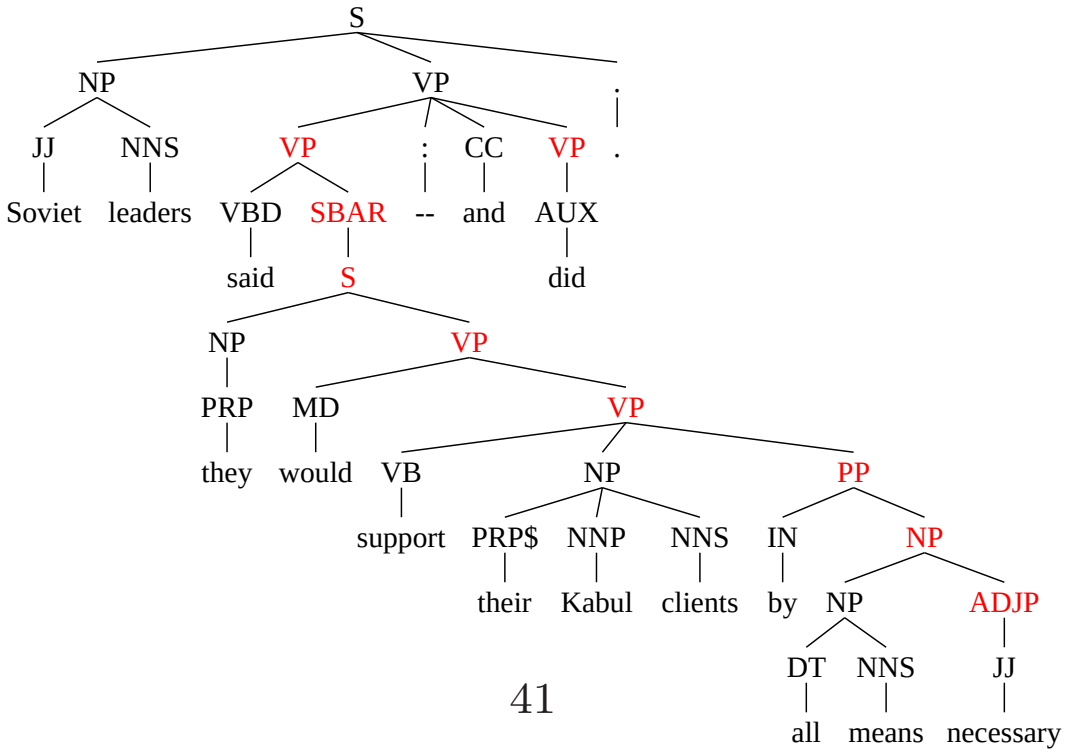
- (Re)ranking parsers can work with just about any features
- The details of linguistic representations don't matter so long as they are rich enough to compute your features from
- The choice of features is extremely important, and needs linguistic insight
- Self-training works with reranking parsers (why?)
- Both reranking and self-training is (surprisingly) domain-independent

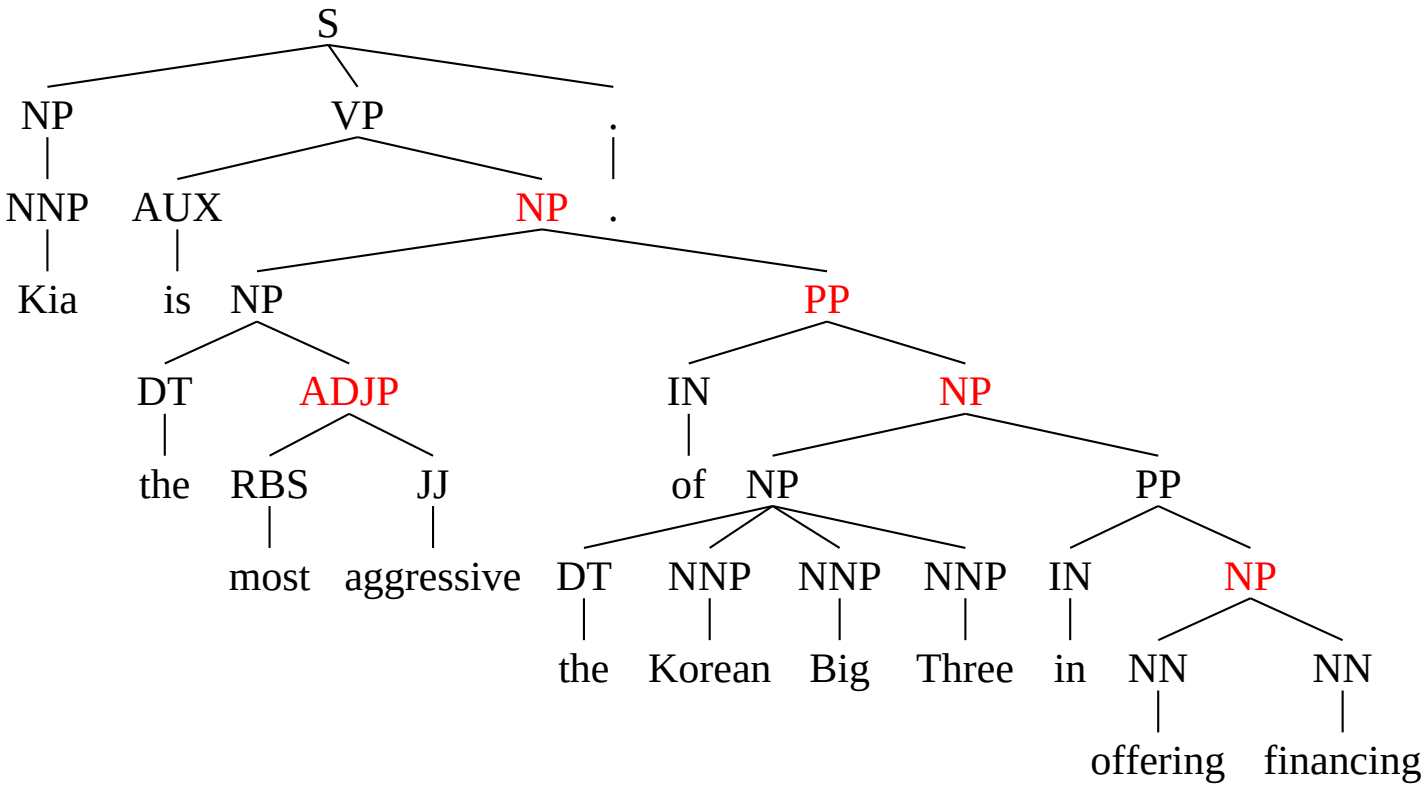
Sample parser errors

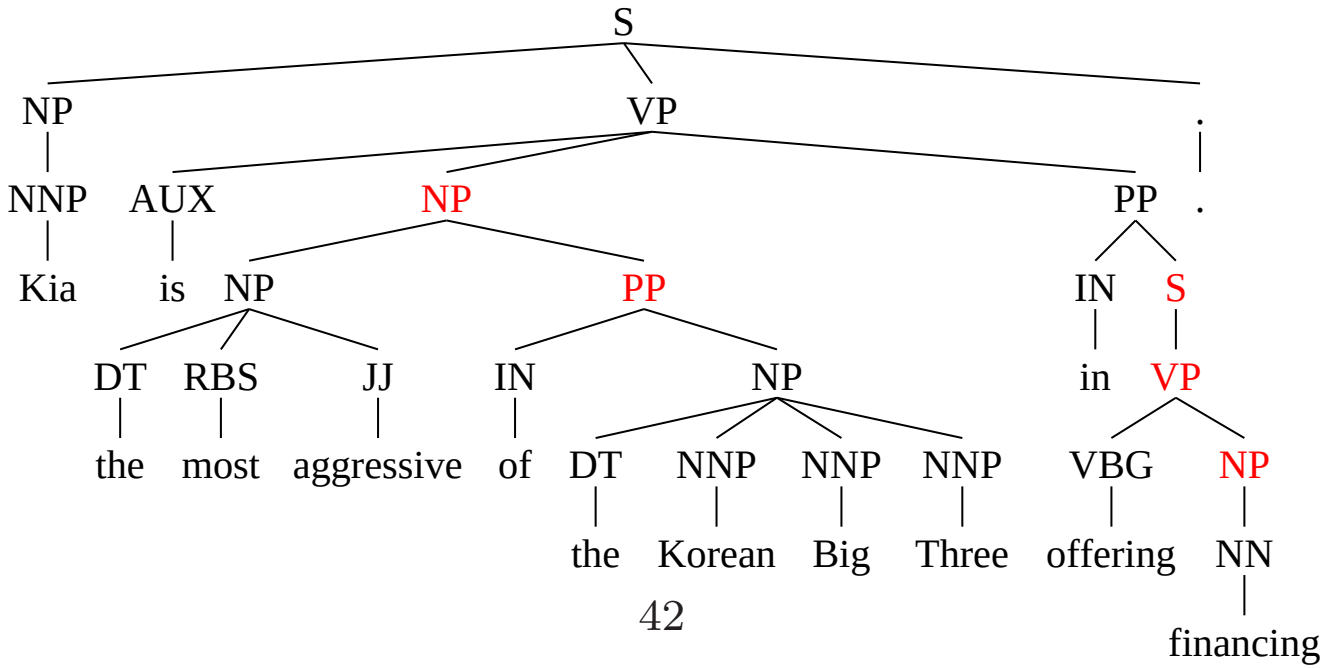


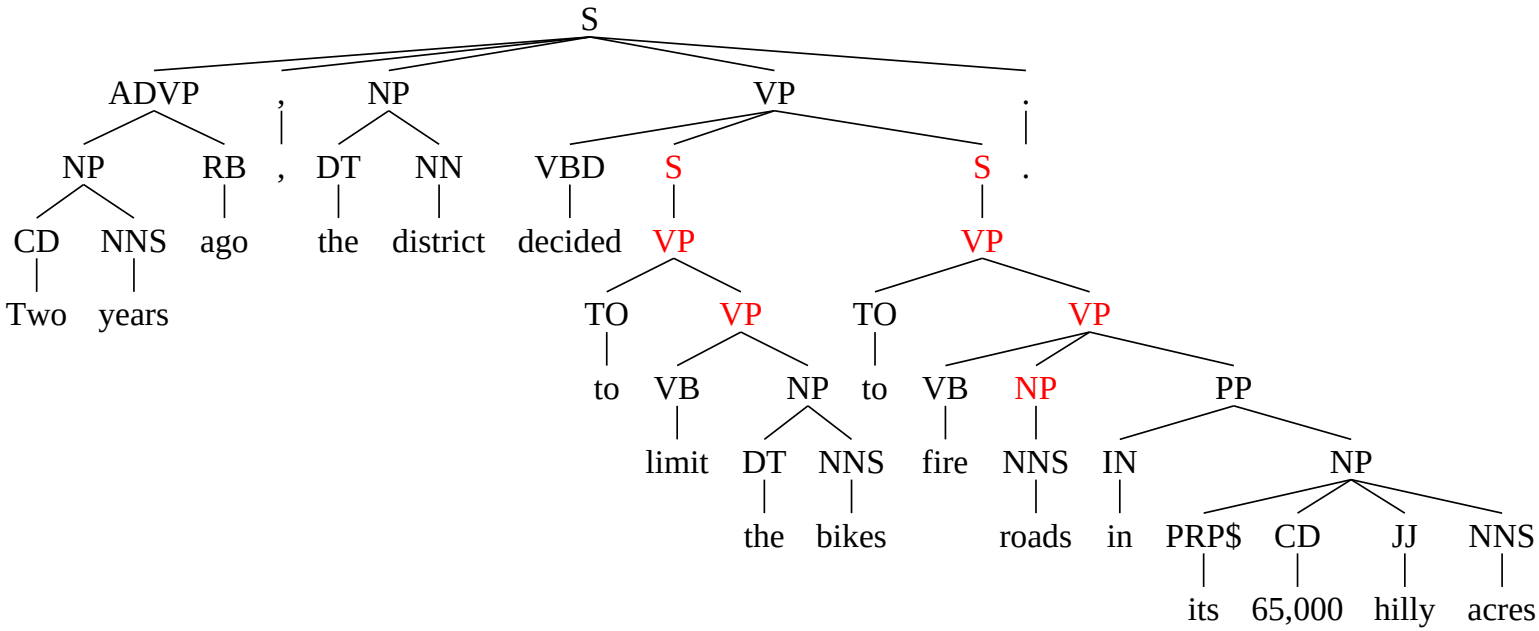


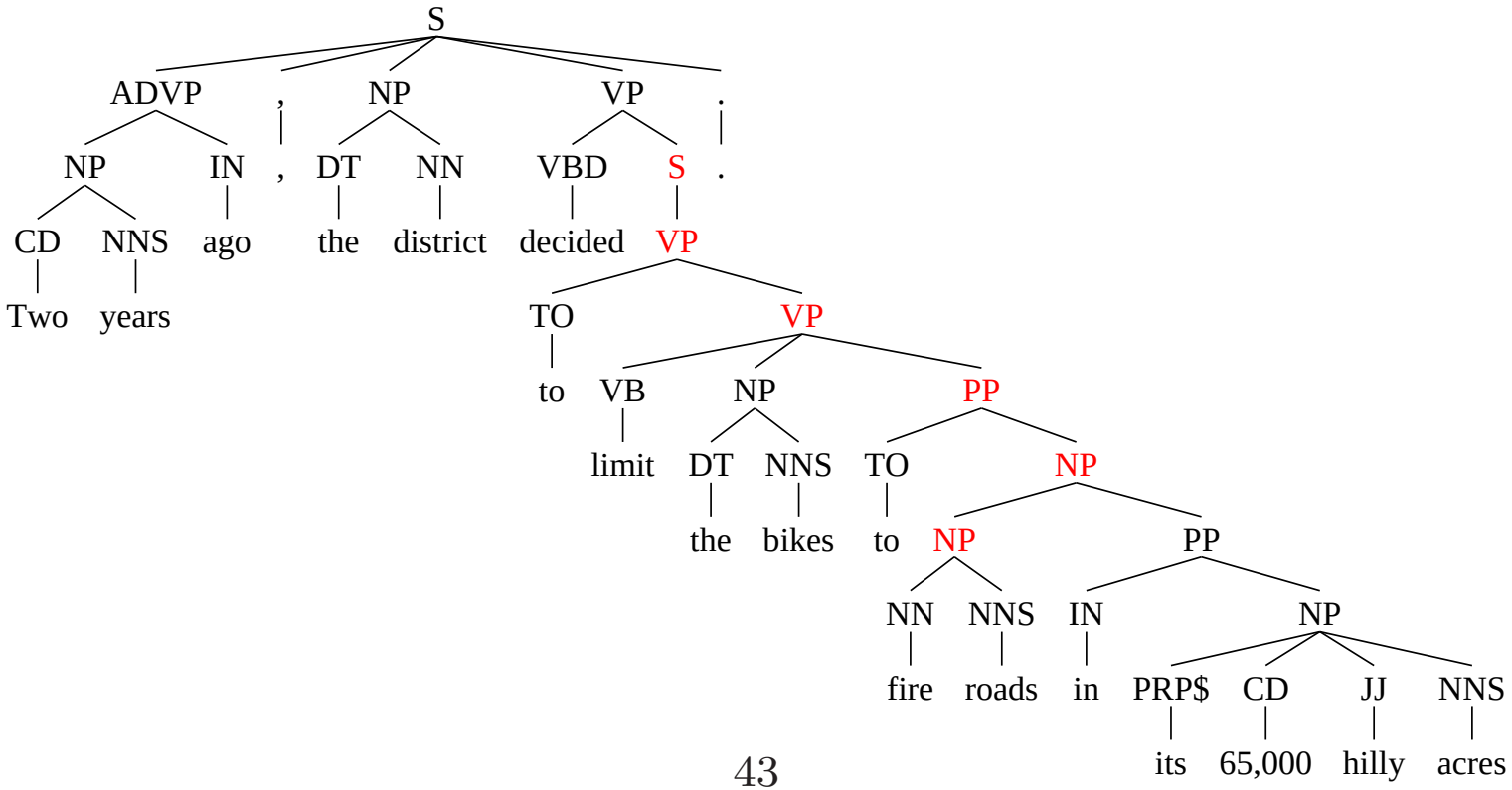

S
NP
VP
.
JJ
NNS
VBD
SBAR
.
Soviet
leaders
said
S
NP
VP
PRP
MD
VP
they
would
VB
NP
PP
support
PRP$
NNP
NNS
IN
NP
their
Kabul
clients
by
NP
ADJP
DT
NNS
JJ
:
CC
AUX
all
means
necessary
--
and
did

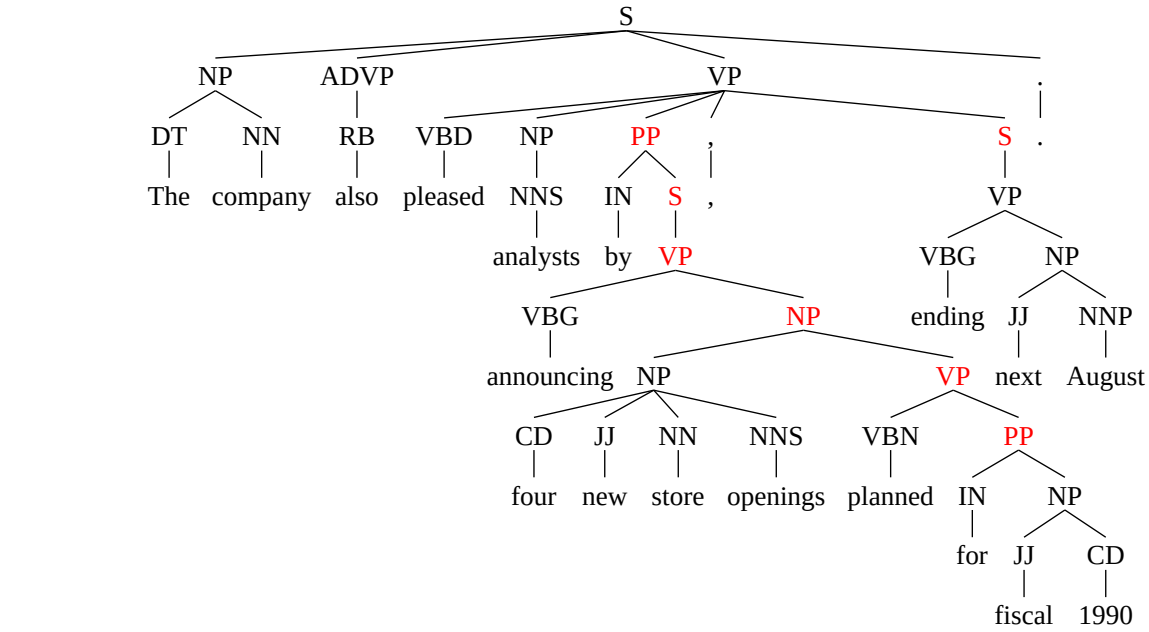

S
NP
VP
.
JJ
NNS
VP
:
CC
VP
.
Soviet
leaders
VBD
SBAR
--
and
AUX
said
S
did
NP
VP
PRP
MD
VP
they
would
VB
NP
PP
support
PRP$
NNP
NNS
IN
NP
their
Kabul
clients
by
NP
ADJP
DT
NNS
JJ
all
means
necessary

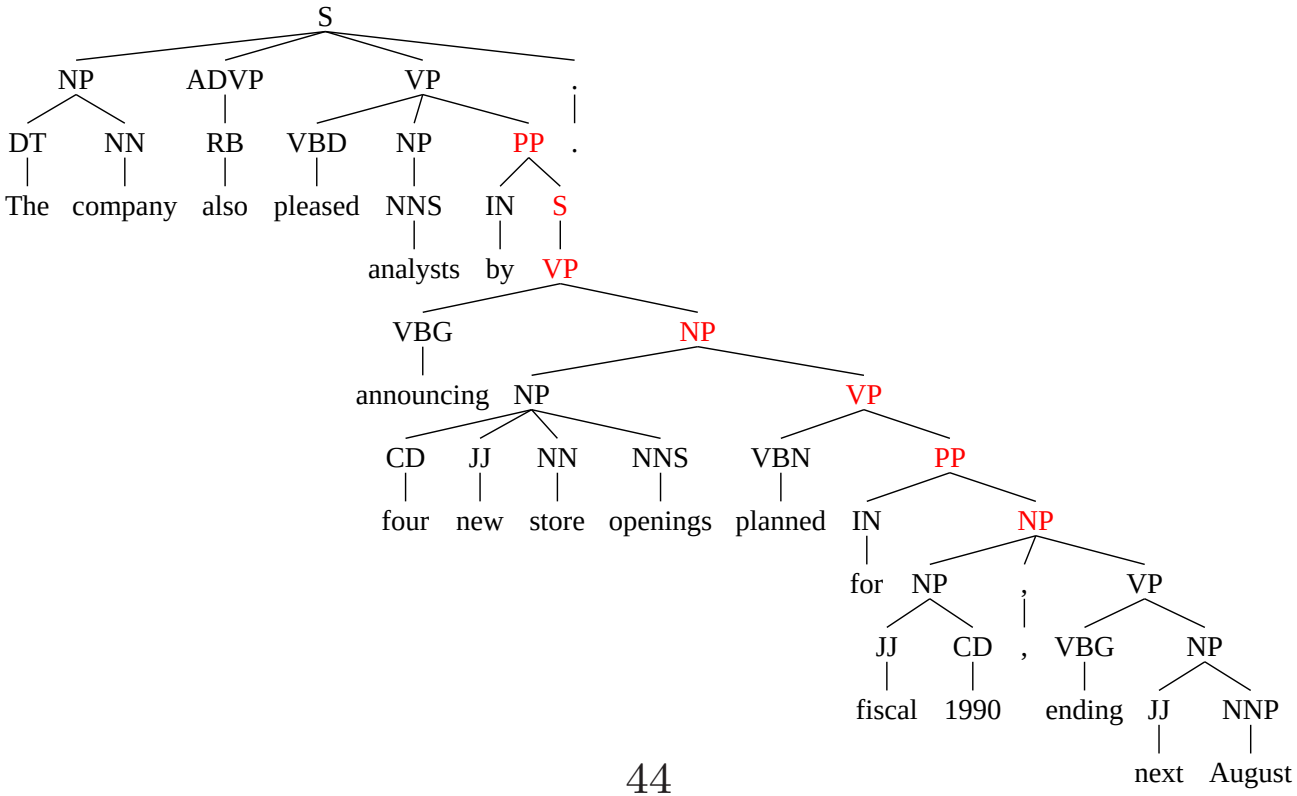

S
NP
VP
.
NNP
AUX
NP
.
Kia
is
NP
PP
DT
ADJP
IN
NP
the
RBS
JJ
of
NP
PP
most
aggressive
DT
NNP
NNP
NNP
IN
NP
the
Korean
Big
Three
in
NN
NN
offering
financing

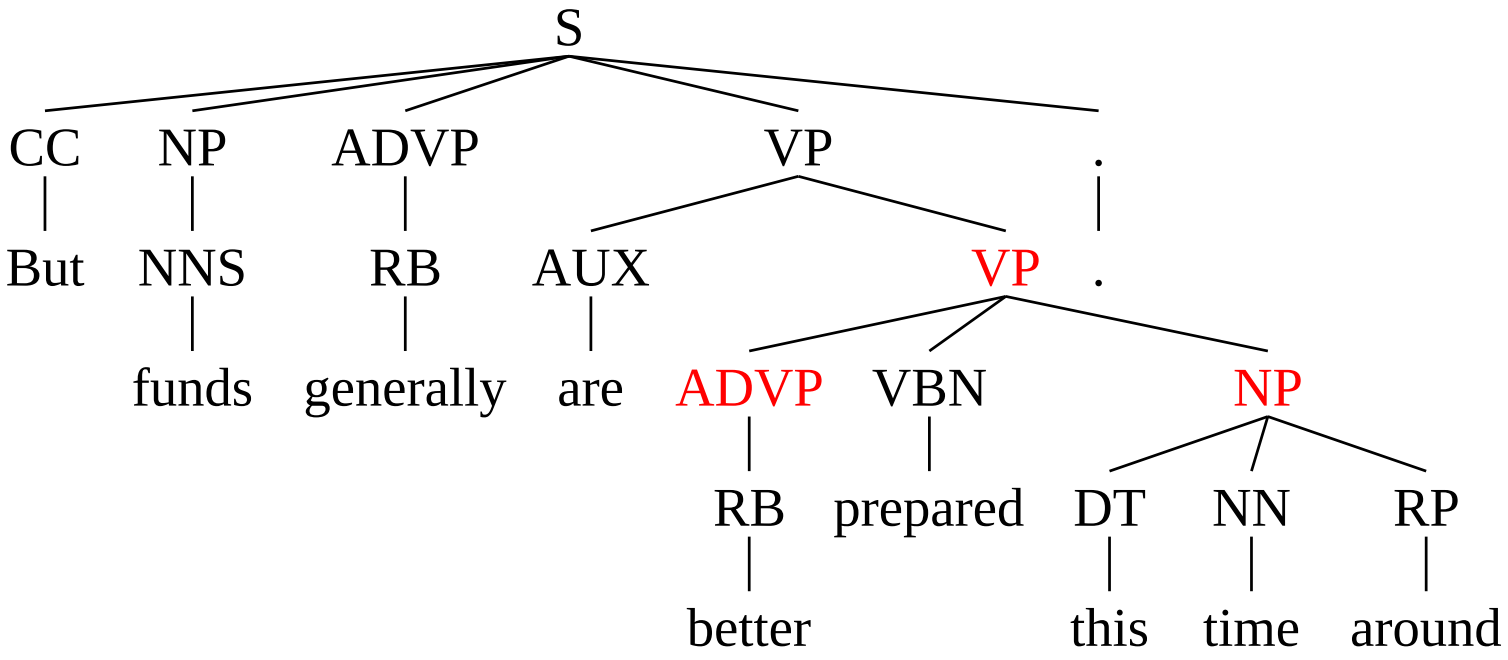

S
NP
VP
.
NNP
AUX
NP
PP
.
Kia
is
NP
PP
IN
S
DT
RBS
JJ
IN
NP
in
VP
the
most
aggressive
of
DT
NNP
NNP
NNP
VBG
NP
the
Korean
Big
Three
offering
NN
financing

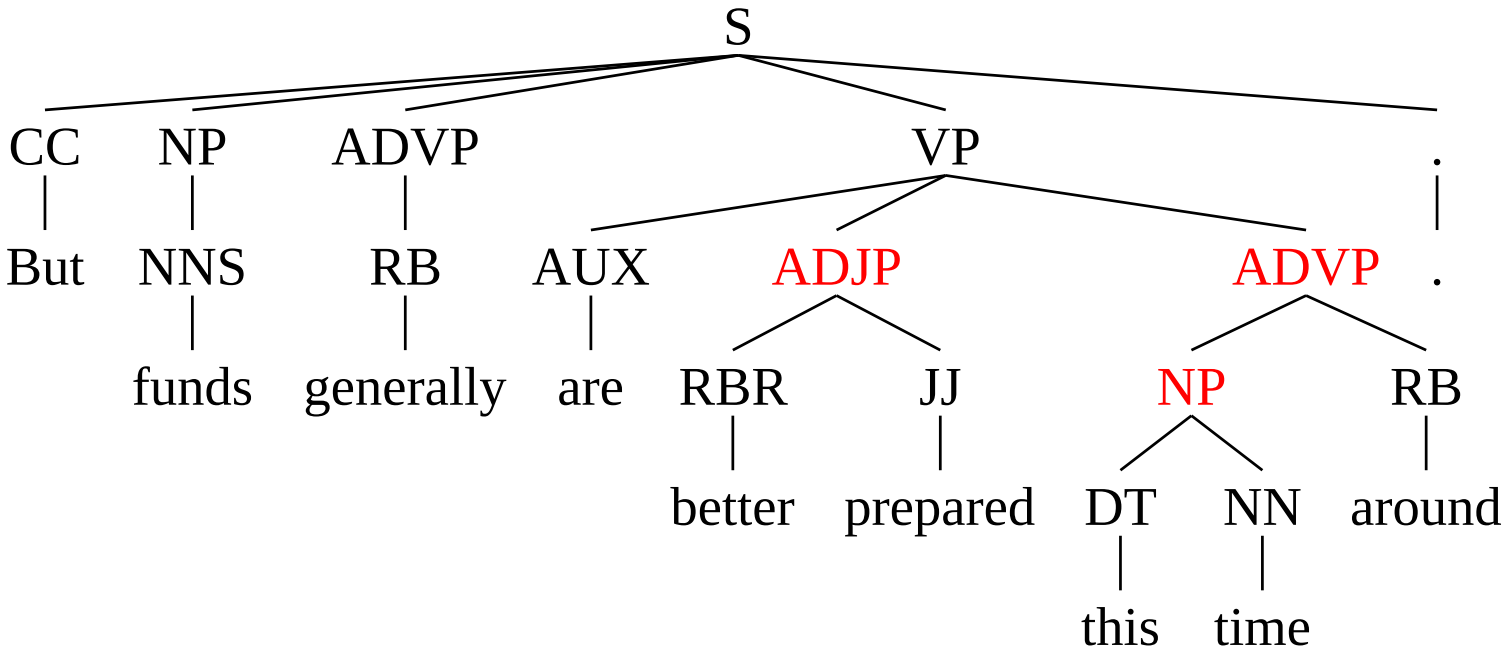

S
ADVP , NP VP .
NP RB , DT NN VBD S S .
CD NNS ago the district decided VP VP
Two years TO VP TO VP
to VB NP to VB NP PP
limit DT NNS fire NNS IN NP
the bikes roads in PRP$ CD JJ NNS
its 65,000 hilly acres


S
ADVP , NP VP .
NP IN , DT NN VBD S .
CD NNS ago the district decided VP
Two years TO VP
to VB NP PP
limit DT NNS TO NP
the bikes to NP PP
NN NNS IN NP
fire roads in PRP$ CD JJ NNS
its 65,000 hilly acres


S
NP ADVP VP .
DT NN RB VBD NP PP , S .
The company also pleased NNS IN S , VP
analysts by VP VBG NP
VBG NP ending JJ NNP
announcing NP VP next August
CD JJ NN NNS VBN PP
four new store openings planned IN NP
for JJ CD
fiscal 1990


S
NP ADVP VP .
DT NN RB VBD NP PP .
The company also pleased NNS IN S
analysts by VP
VBG NP
announcing NP VP
CD JJ NN NNS VBN PP
four new store openings planned IN NP
for NP , VP
JJ CD , VBG NP
fiscal 1990 ending JJ NNP
next August


S
CC
NP
ADVP
VP
.
But
NNS
RB
AUX
VP
.
funds
generally
are
ADVP
VBN
NP
RB
prepared
DT
NN
RP
better
this
time
around


S
CC
NP
ADVP
VP
.
But
NNS
RB
AUX
ADJP
ADVP
.
funds
generally
are
RBR
JJ
NP
RB
better
prepared
DT
NN
around
this
time