Tutorial on Corpus-based Natural Language Processing

Anna University, Chennai, India. December, 2001

Anoop Sarkar

anoop@cis.upenn.edu

Chunking and Statistical Parsing

Anoop Sarkar
http://www.cis.upenn.edu/~anoop/
anoop@linc.cis.upenn.edu

Chunking and Statistical Parsing: Lecture 1

- General Introduction
- Plan for the remaining lectures
- Chunking

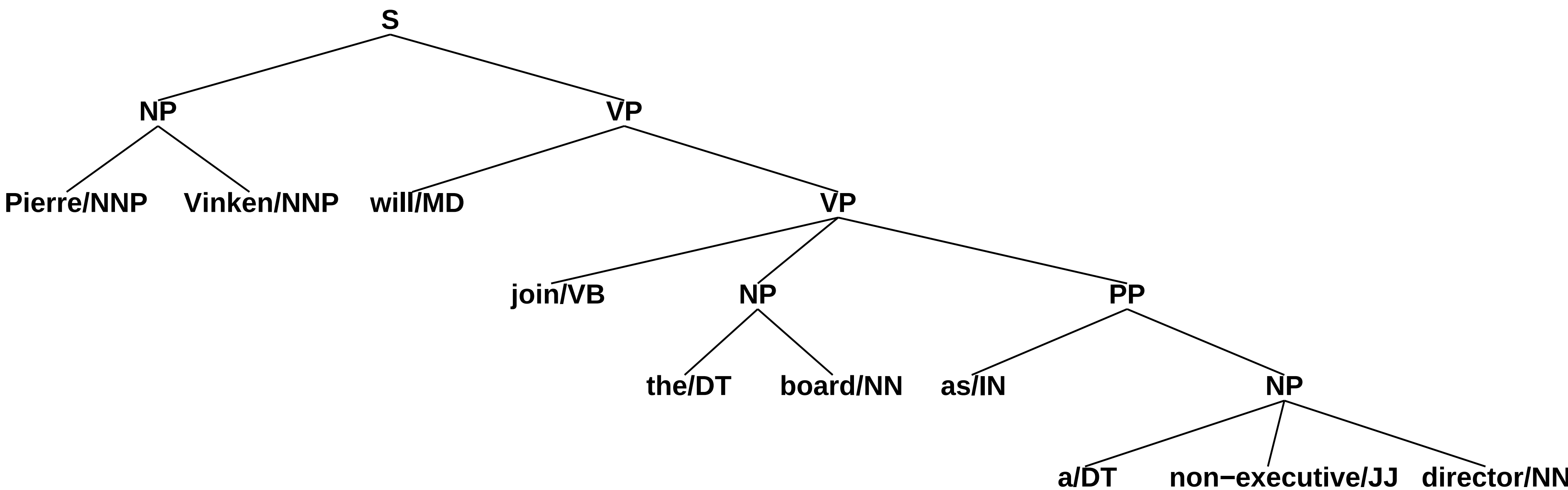
General Introduction

- Why is syntactic analysis scientifically interesting?
  - AI and Cognitive Science:
    model the human ability to map sounds to meaning structures
  - Human-computer interaction (both speech and text)
  - Numerous engineering problems should benefit from progress in parsing
    e.g machine translation, information extraction, summarization, etc.

General Introduction: Types of syntactic analysis

- Constituent analysis
  - Examples: English Treebank, Korean Treebank
  - Complex structural labels and descriptions including empty elements: consequently hard to induce using self-organizing methods
  - Complexity analysis: triangulating polygons

General Introduction: Constituent Analysis



General Introduction: Types of syntactic analysis

- Word-word dependency structure
  - Examples: Various Indian Language corpora, Czech Prague Treebank, Japanese EDR corpus, German Negra Treebank
  - Awkward analysis of crossing dependencies, and phenomena that target constituents (e.g: coordination and ellipsis)
  - Complexity analysis: bipartite graph matching

General Introduction: Dependency Structure

0	Pierre/NNP	1
1	Vinken/NNP	3
2	will/MD	3
3	join/VB	TOP
4	the/DT	5
5	board/NN	3
6	as/IN	3
7	a/DT	9
8	non-executive/JJ	9
9	director/NN	6

General Introduction: Types of syntactic analysis

- Chunking and Shallow Parsing
  - Only non-recursive constituent analysis
  - Complexity analysis: linear time

```
>> [ John/NNP ] saw/VBD [the/DT cat/NN]
        [the/DT dog/NN] liked/VBD ./.
>> [ John/NNP Smith/NNP ] ,/, [ president/NN ]
        of/IN [ IBM/NNP ] ./.
>> [ Pundits/NNS ] condemned/VBD [ terrorism/NN ]
        and [ assassination/NN ] ./.
```

General Introduction: Practical Issues in Parsing

- Polynomial time parsing algorithms

- Ambiguity resolution

General Introduction: Applications of Parsing

- Information extraction: BBN system for MUC-7
- Semantic analysis for dialog systems: ATIS/Communicator systems
- Language modeling: (Chelba and Jelinek 1998), (Srinivas 1999), (Roark 2001)

Plan for the remaining lectures

- Lecture 1 (in time left): Chunking
  - Basal NP chunking and regular grammars
  - NP chunking and machine learning
- Lecture 2: Parsing Algorithms
- Lecture 3: Statistical Parsing
- Lecture 4: Current Directions, Applications and Projects

Classifiers I: Naive Bayes, Mutual Information

Anoop Sarkar

http://www.cis.upenn.edu/~anoop/

anoop@linc.cis.upenn.edu

Naive Bayes

- Building a classifier: $P(\text{class} \mid \text{evidence})$

- For example: document classification

- Problem: classify web pages into course home pages or not

- Given: Training data, web pages classified by hand into 2 classes

Naive Bayes

- Construct: classifier that learns from training data and produces a model $P(c \mid d)$
  where c is the class and d is the input document

- The model picks likely class ($P > 0.5$) and assigns a document to be in a particular class

- For each input document we split up the document into a set of features $f_1, \ldots, f_k$
  For document classification, typically $f_1, \ldots, f_k = w_1, \ldots, w_k$

Naive Bayes

$$\arg\max_{c} P(c \mid f_1 \ldots f_k) =$$

$$\arg\max_{c} \frac{P(f_1 \ldots f_k \mid c) \cdot P(c)}{P(f_1 \ldots f_k)}$$

$$\arg\max_{c} P(c) \cdot \prod_{i=1}^{k} \frac{P(f_i \mid c)}{P(f_i)}$$

$$\arg\max_{c} P(c) \cdot \prod_{i=1}^{k} P(f_i \mid c)$$

Naive Bayes

$$P(f_i \mid c) = \frac{\text{count}(f_i, c)}{\text{count}(c)}$$

$$P(c) = \frac{\text{count}(c)}{|\mathcal{C}|}$$

Experiments with Naive Bayes: (Blum and Mitchell 1998)

- Task: classify web pages into course home pages or not

- Training data: 1051 web pages from 4 universities
  22% were course home pages

- Train two classifiers with different sets of features
  1: words in the document
  2: words in the hyperlink to the document

Experiments with Naive Bayes: (Blum and Mitchell 1998)

	Page based	Hyperlink based	Combined
Error rate	12.9	12.4	11.1

Other Methods: Removing Independence Assumptions

- Unjustified independence assumptions hurt performance

- Example:
  Feature 1 = If `suffix == ing` then part of speech is `verb`
  Feature 2 = If `first letter is capitalized` then part of speech is `noun`

- What about `Boeing`?

- Machine learning methods like Maximum Entropy Models (MRFs), Boosting and Support Vecture Machines (SVMs) try to solve this problem

Sticky Pairs and Semantic Classes

- Finding pairs of words using co-occurence statistics
- Mutual Information between adjacent words:

$$\log \frac{P(w_1, w_2)}{P(w_1) \cdot P(w_2)}$$

- MI can be negative (not correlated) or positive (correlated)
- Not symmetric: $MI(w_1, w_2) \neq MI(w_2, w_1)$

Sticky Pairs and Semantic Classes

- In 59 million words from the Hansards:

Word Pair	MI
Humpty Dumpty	22.5
Klux Klan	22.2
Ku Klux	22.2
Chah Nulth	22.2
...	...
avant garde	22.1
...	...
Taj Mahal	21.8
...	...

Sticky Pairs and Semantic Classes

- Generalize $P(w_1, w_2)$ to $P_{\text{near}}(w_1, w_2)$
- Choose w_1 at random
- Next choose w_2 at random from a surrounding 1000 word window
- If $P_{\text{near}}(w_1, w_2)$ is greater than $P(w_1) \cdot P(w_2)$ then w_1 and w_2 can be said to be *semantically sticky*
- Is symmetric: $MI_{\text{near}}(w_1, w_2) = MI_{\text{near}}(w_2, w_1)$

Sticky Pairs and Semantic Classes

we our us ourselves ours
question questions asking answer answers answering
performance performed perform performs performing
tie jacket suit
write writes writing written wrote pen
morning noon evening night nights midnight bed
attorney counsel trial court judge
problems problem solution solve analyzed solved solving

Chunking and Statistical Parsing

Anoop Sarkar

http://www.cis.upenn.edu/~anoop/

anoop@linc.cis.upenn.edu

Introduction to Chunking

`nltk chunking demo`: Steven Bird and Edward Loper

Chunking and Machine Learning

- Approach so far: writing rules by hand
- Machine learning approach
  - Create or get annotated training data
  - Learn rules from annotated data

Chunking and Machine Learning

```
[nx
          nns          techniques
    [pp
          in           for
        [nx
          vbg          bootstrapping
          jj           broad
          nn           coverage
          nns          parsers
        ]
    ]
]
```

Chunking as Part-of-Speech Tagging

```
>> [ John/NNP ] saw/VBD [the/DT cat/NN]
      I          O        I      I
        [the/DT dog/NN] liked/VBD ./.
          B       I       O         O

>> [ John/NNP Smith/NNP ] ,/, [ president/NN ]
      I         I           O        I
        of/IN [ IBM/NNP ] ./.
         O        I         O

>> [ Pundits/NNS ] condemned/VBD [ terrorism/NN ]
         I             O              I
        and [ assassination/NN ] ./.
         O          I             O
```

Chunking as Tagging: (Ramshaw and Marcus 1995)

- Improving on rote learning: Learning generalizations from annotated data.
- Many methods can be applied, including Hidden Markov Models.
- Modeling involves selection of the space of features.
- Actual features are learned from the labeled data.
- Accuracy: 93% of basal NPs correctly identified in unseen test data.

Chunking as Tagging: (Ramshaw and Marcus 1995)

- Transformation-Based Learning
  - Start with a default assignment.
  - For training data: find all rules that transform an incorrect tag to a correct tag conditioned on some local context.
  - Pick best rule by ranking each rule based on the number of errors it repairs.
  - Apply rule to training data: return to second step.
  - If no more errors can be eliminated, output ranked list of rules.

Shallow Parsing or Partial Parsing

- `lda demo`: Lightweight Dependency Analyzer (Srinivas and Joshi 1999)
- `cass demo`: Abney's shallow parser (Abney 1996)

Chunking and Statistical Parsing

Anoop Sarkar
http://www.cis.upenn.edu/~anoop/
anoop@linc.cis.upenn.edu

Recognition and Parsing for CFG and TAG

- Why are parsing algorithms important?
- A simple parsing algorithm for CFGs
- Complexity analysis
- Extracting parse derivations: derivation forests
- Extension of CKY for TAGs

Why are parsing algorithms important?

- A linguistic theory is implemented in a formal system to generate the set of grammatical strings and rule out ungrammatical strings.
- Such a formal system has computational properties.
- One such property is a simple decision problem: given a string, can it be generated by the formal system *(recognition)*.
- If it is generated, what were the steps taken to recognize the string *(parsing)*.

Why are parsing algorithms important?

- Consider the recognition problem: find algorithms for this problem for a particular formal system.
- The algorithm must be decidable.
- Preferably the algorithm should be polynomial: enables computational implementations of linguistic theories.
- Elegant, polynomial-time algorithms exist for formalisms like CFG, TAG.

A recognition algorithm for CFGs

- Consider the CFG G:

  1. $S \rightarrow S\ S$
  2. $S \rightarrow a$

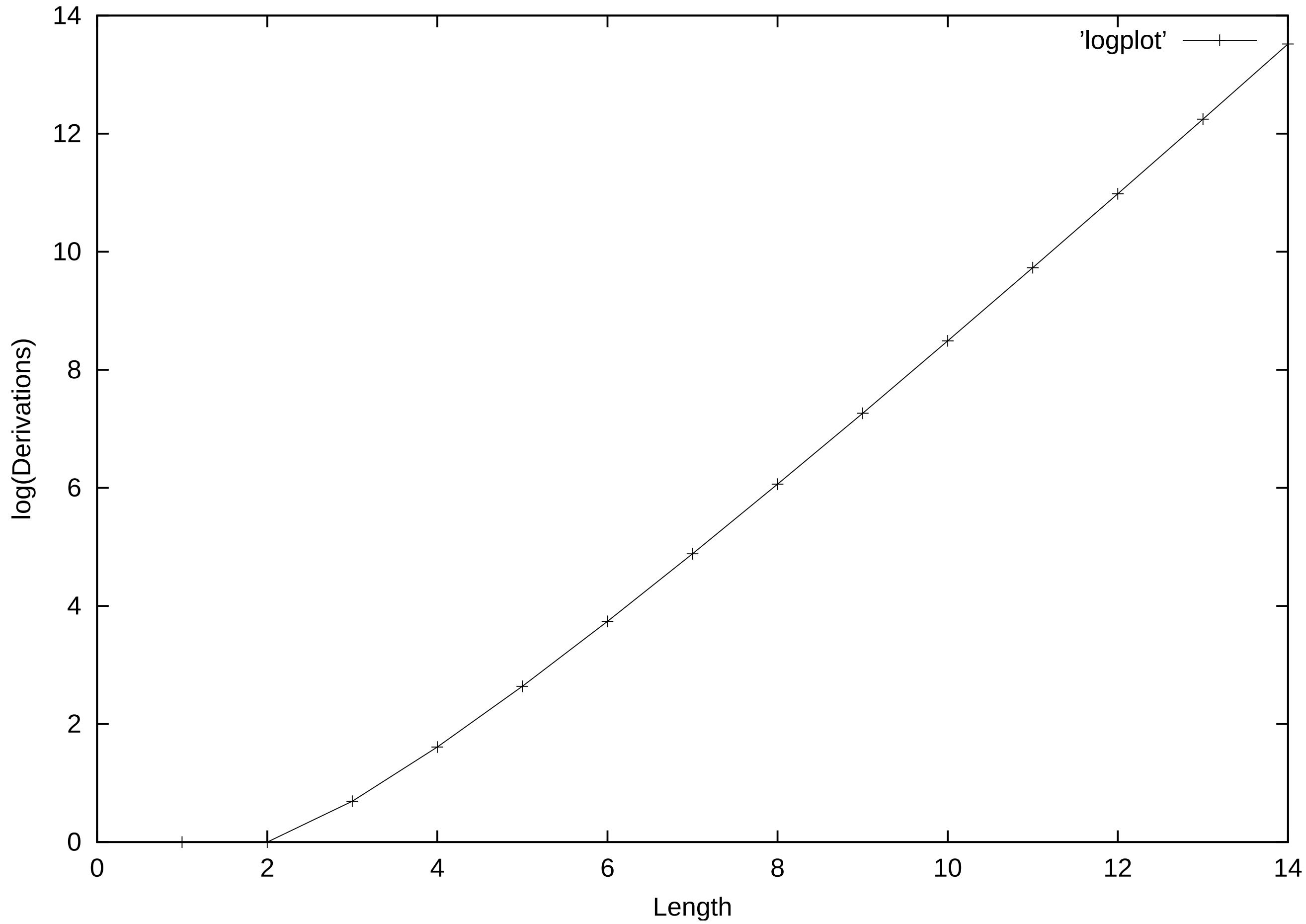
  $L(G) = a^i$ for $i >= 1$

- The recognition question: does the string aaa belong to $L(G)$?
  - Input: aaa
  - Output: $\{$*yes*, *no*$\}$

Parsing algorithms for CFG and TAG

- `nltk parse demo`: Steven Bird and Edward Loper
- `ckycfg parse demo`: Anoop Sarkar
- `ckytig demo`: Anoop Sarkar

Number of derivations grows exponentially

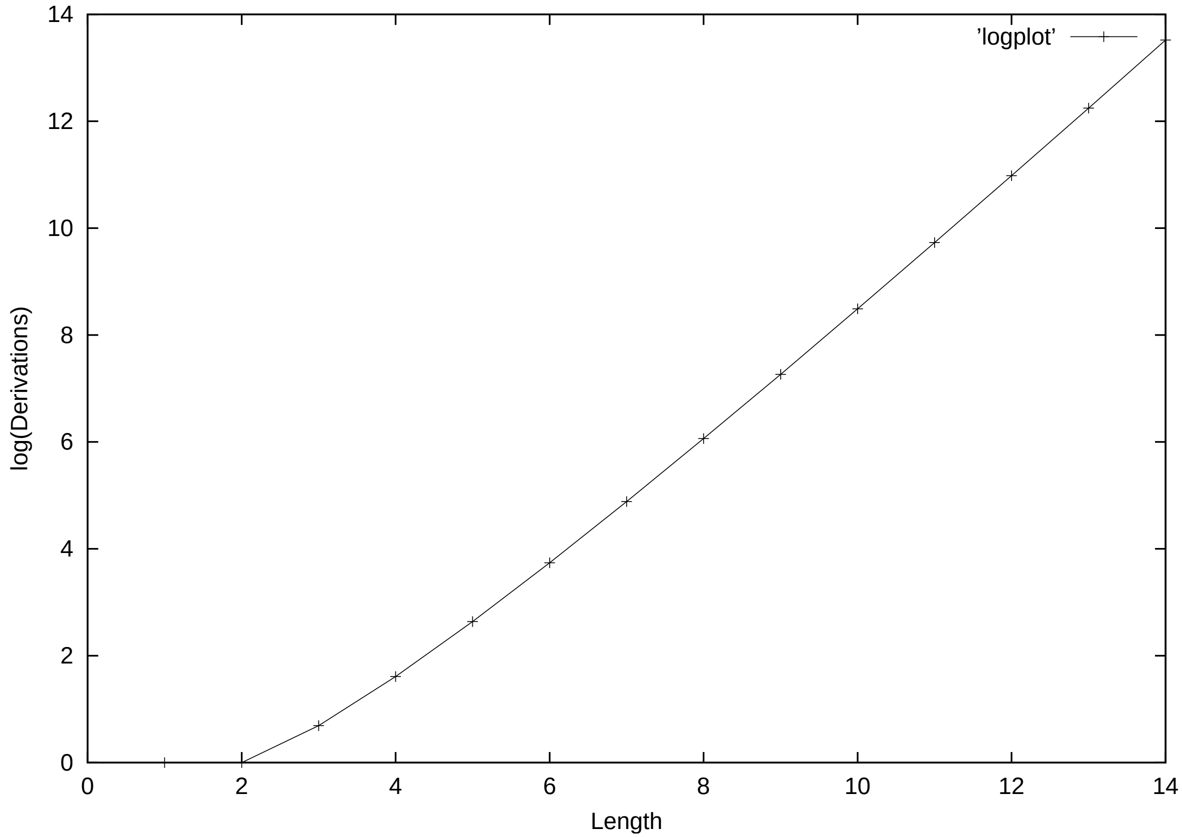


Chunking and Statistical Parsing

Anoop Sarkar
http://www.cis.upenn.edu/~anoop/
anoop@linc.cis.upenn.edu

Number of derivations grows exponentially

e.g. `L(G) = a a ...` for `G = S` $\longrightarrow$ `S S`



Syntactic Ambiguity: (Church and Patil 1982)

- Algebraic character of parse derivations

- Power Series for grammar:

  `NP → cabbages | kings | NP and NP`

```
NP = cabbages + cabbages and kings
       + 2 (cabbages and cabbages and kings)
       + 5 (cabbages and kings and cabbages and kings)
       + 14 ...
```

Syntactic Ambiguity: (Church and Patil 1982)

- Coefficients equal the number of parses for each NP string
- These ambiguity coefficients are Catalan numbers:

$$Cat(n) = \left(\begin{array}{c} 2n \\ n \end{array} \right) - \left(\begin{array}{c} 2n \\ n-1 \end{array} \right)$$

- $\left(\begin{array}{c} a \\ b \end{array} \right)$ is the *binomial coefficient*

$$\left(\begin{array}{c} a \\ b \end{array} \right) = \frac{a!}{(b!(a-b)!)}$$

Syntactic Ambiguity: (Church and Patil 1982)

- $Cat(n)$ also provides exactly the number of parses for the sentence:

```
John saw the man
         on the hill
         with the telescope
```

- Other sub-grammars are simpler:

$$
\begin{aligned}
ADJP &\rightarrow adj\ ADJP \mid \epsilon \\
ADJP &= 1 + adj + adj^2 + adj^3 + \ldots \\
ADJP &= \tfrac{1}{1-adj}
\end{aligned}
$$

Syntactic Ambiguity: (Church and Patil 1982)

- Now consider power series of combinations of sub-grammars:

  $\mathtt{S} = \mathtt{NP} \cdot \mathtt{VP}$

  ```
  ( The number of products over sales ... )
  ( is near the number of sales ... )
  ```

- Both the `NP` subgrammar and the `VP` subgrammar power series have Catalan coefficients

Syntactic Ambiguity: (Church and Patil 1982)

- The power series for the $\texttt{S} \rightarrow \texttt{NP VP}$ grammar is the multiplication:

$$(N \sum_i Cat_i (P\ N)^i) \cdot (is \sum_j Cat_j (P\ N)^j)$$

- In a parser for this grammar, this leads to a cross-product:

$$L \times R = \{(l, r) \mid l \in L \ \& \ r \in R \}$$

Ambiguity Resolution: Prepositional Phrases in English

- Statistical Methods for Prepositional Phrase Attachment: Annotated Data

V	N1	P	N2	Attachment
join	board	as	director	V
is	chairman	of	N.V.	N
using	crocidolite	in	filters	V
bring	attention	to	problem	V
is	asbestos	in	products	N
making	paper	for	filters	N
including	three	with	cancer	N

Prepositional Phrase Attachment

Method	Accuracy
Always noun attachment	59.0
Most likely for each preposition	72.2
Average Human (4 head words only)	88.2
Average Human (whole sentence)	93.2

If $p(1 \mid v, n1, p, n2) >= 0.5$ choose noun attachment

$$
\begin{aligned}
p(1 \mid v, n1, p, n2) \;=\;& \lambda(c_1) && \cdot\; p(1 \mid c_1 = v, n1, p, n2) \\
+\;& \lambda(c_2 + c_3 + c_4) && \cdot\; p(1 \mid c_2 = v, n1, p) \\
&&& \cdot\; p(1 \mid c_3 = v, p, n2) \\
&&& \cdot\; p(1 \mid c_4 = n1, p, n2) \\
+\;& \lambda(c_5 + c_6 + c_7) && \cdot\; p(1 \mid c_5 = v, p) \\
&&& \cdot\; p(1 \mid c_6 = n1, p) \\
&&& \cdot\; p(1 \mid c_7 = p, n2) \\
+\;& \lambda(c_8) && \cdot\; p(1 \mid c_8 = p) \\
+\;& (1 - \textstyle\sum_i \lambda(c_i)) && \cdot\; 1.0 \;\; \text{(default is noun attachment)}
\end{aligned}
$$

Prepositional Phrase Attachment: (Collins and Brooks 1995)

- Lexicalization helps disambiguation by capturing selectional preferences
  (Black et al. 1994; Magerman 1995)

- Smoothing deals with sparse data and improves prediction
  we ignore word senses here; cf. (Stetina and Nagao 1998)

- Uses the head of the phrase (e.g. prep) as privileged

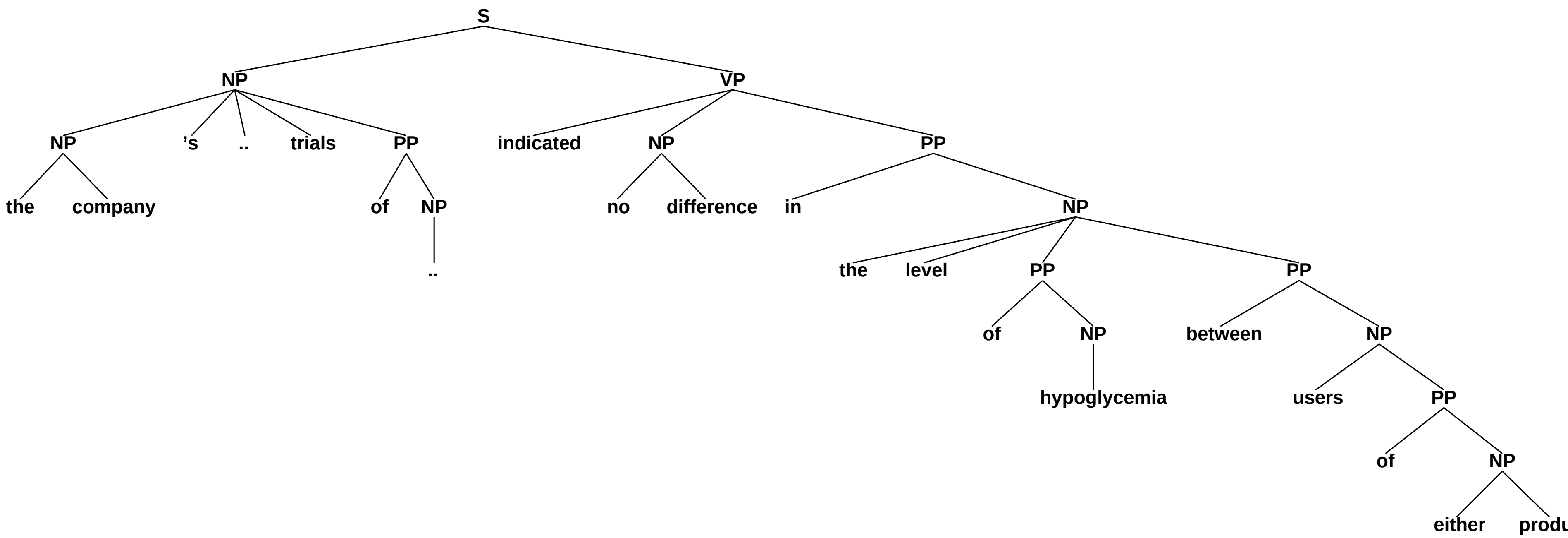
- Similar insights led to lexicalization of grammars in mathematical linguistics and all-paths parsing; cf. TAG, CCG

Prepositional Phrase Attachment: (Collins and Brooks 1995)

- **Results**: 84.1% accuracy
- Adding word sense disambiguation increases accuracy to 88%
  (Stetina and Nagao 1998)
- Can we improve on parsing performance using Probabilistic CFGs by using the insights detailed above

Statistical Parsing: Annotated Data == Treebank:

the company 's clinical trials of both its animal and human-based insulins indicated no difference in the level of hypoglycemia between users of either product



Supervised Models for Parsing: History-based models

- Parsing can be framed as a supervised learning task
- Induce function $f : \mathcal{S} \to \mathcal{T}$ given $S_i \in \mathcal{S}$, pick best T_i from $\mathcal{T}(S)$
- Statistical parser builds model $P(T, S)$ for each (T, S)
- The best parse is then $\underset{T \in \mathcal{T}(S)}{\text{arg max}} \; P(T, S)$

History-based models and PCFGs

- History-based approaches maps (T, S) into a decision sequence $d_1, \ldots, d_n$
- Probability of tree T for sentence S is:

$$P(T, S) = \prod_{i=1 \ldots n} P(d_i \mid \phi(d_1, \ldots, d_{i-1}))$$

- ϕ is a function that groups histories into equivalence classes

History-based models and PCFGs

- PCFGs can be viewed as a history-based model using leftmost derivations

- A tree with rules $\langle \gamma_i \rightarrow \beta_i \rangle$ is assigned a probability $\prod_{i=1}^{n} P(\beta_i \mid \gamma_i)$ for a derivation with n rule applications

Generative models and PCFGs

$$
\begin{aligned}
T_{\textit{best}} &= \underset{T}{\text{arg max}}\; P(T \mid S) \\
&= \underset{T}{\text{arg max}}\; \frac{P(T, S)}{P(S)} \\
&= \underset{T}{\text{arg max}}\; P(T, S) \\
&= \prod_{i=1 \ldots n} P(RHS_i \mid LHS_i)
\end{aligned}
$$

Evaluation of Statistical Parsers: EVALB

$$\text{Bracketing recall} = \frac{\text{num of correct constituents}}{\text{num of constituents in the goldfile}}$$

$$\text{Bracketing precision} = \frac{\text{num of correct constituents}}{\text{num of constituents in the parsed file}}$$

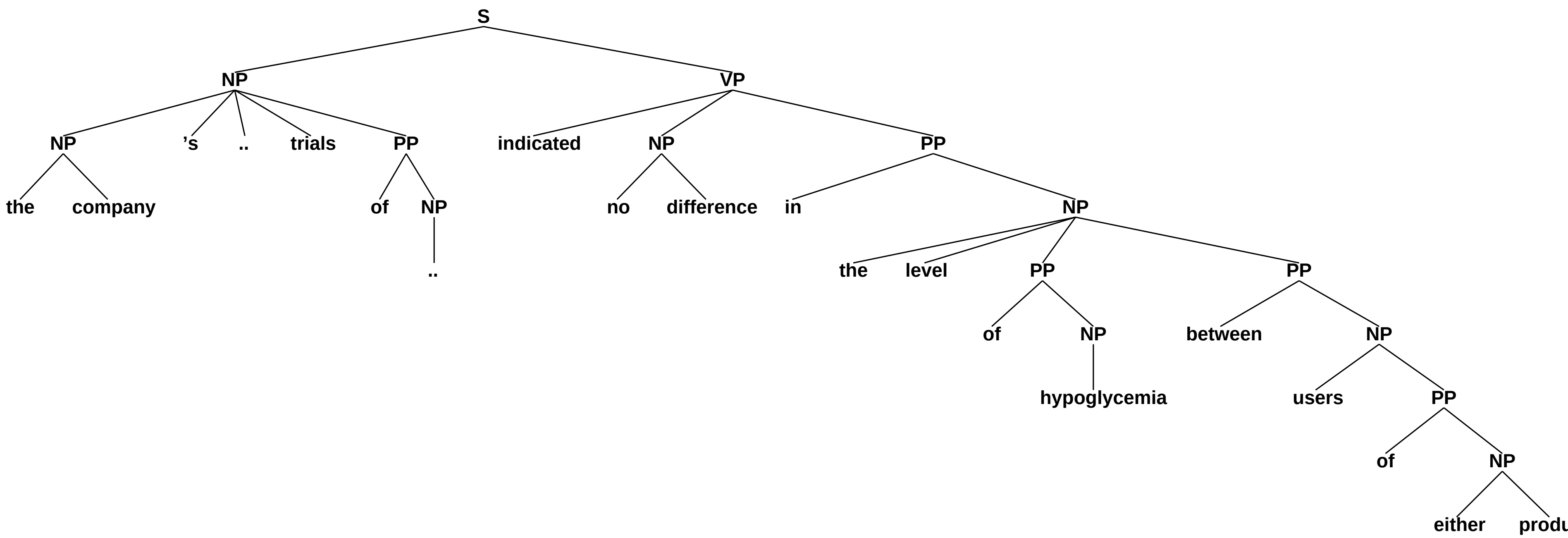
$$\text{Complete match} = \text{\% of sents where recall \& precision are both 100\%}$$

$$\text{Average crossing} = \frac{\text{num of constituents crossing a goldfile constituent}}{\text{num of sents}}$$

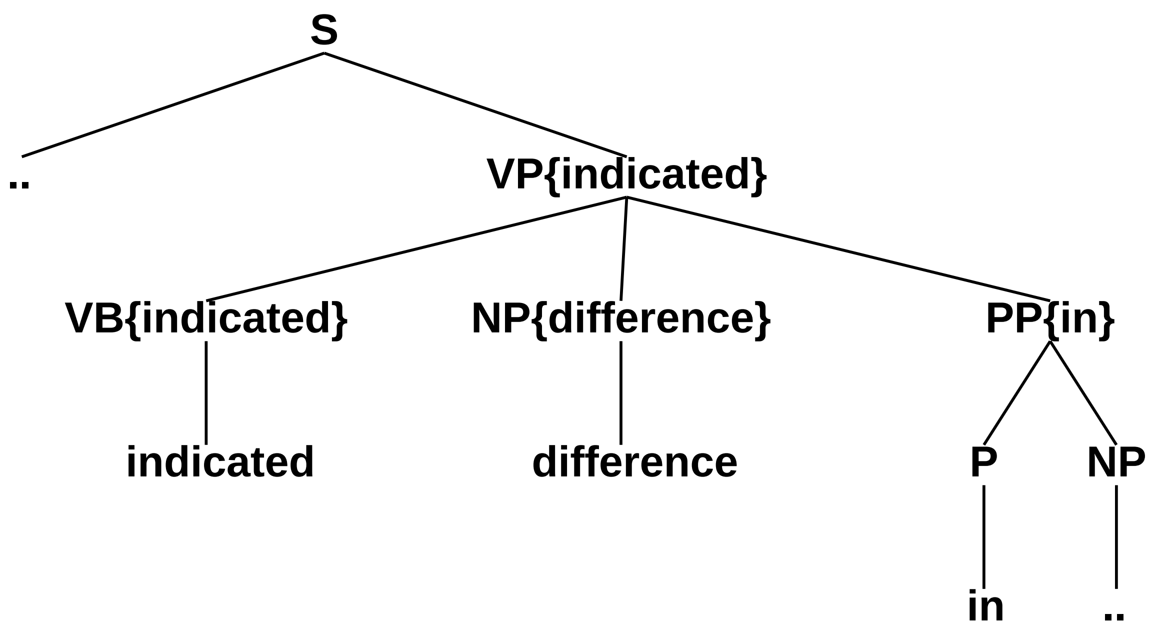
$$\text{No crossing} = \text{\% of sents which have } 0 \text{ crossing brackets}$$

$$2 \text{ or less crossing} = \text{\% of sents which have } \leq 2 \text{ crossing brackets}$$

Statistical Parsing and PCFGs

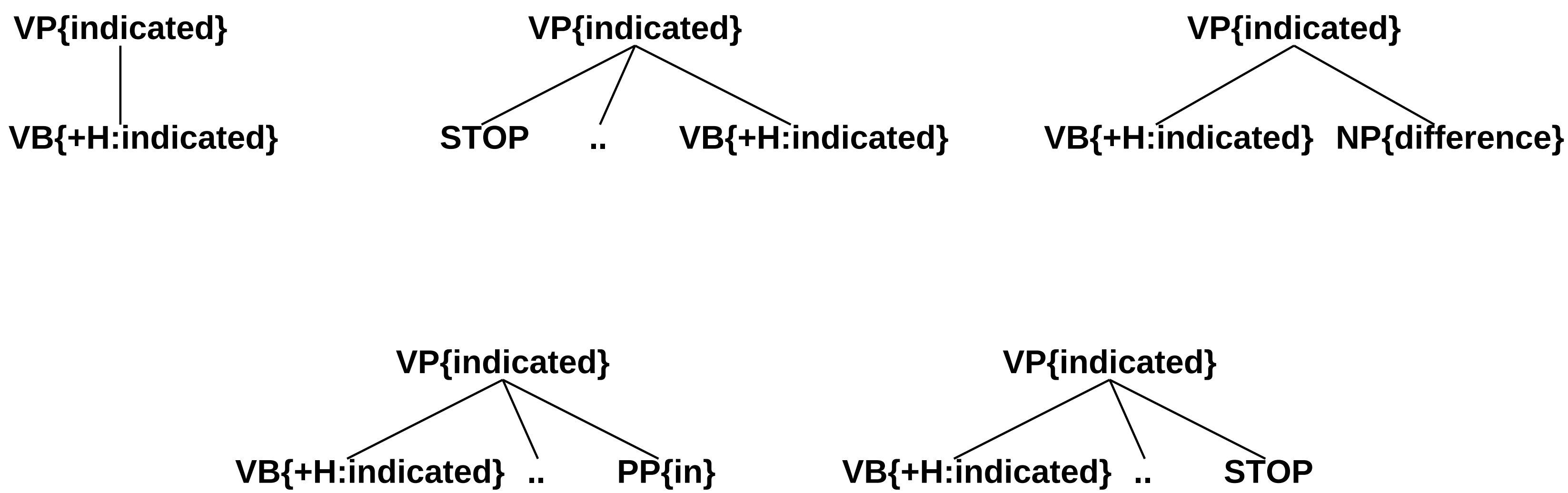


Bilexical CFG: (Collins 1997)



Demo: `mcollins-sec00.txt`

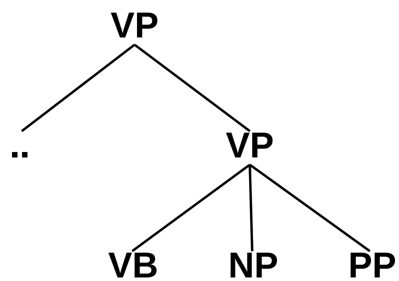
Bilexical CFG: VP{indicate} → VB{+H:indicate} NP{difference} PP{in}



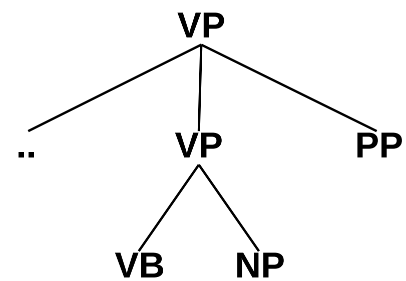
$$
\begin{aligned}
&P_h(\texttt{VB} \mid \texttt{VP}, \texttt{indicated}) \times P_l(\texttt{STOP} \mid \texttt{VP}, \texttt{VB}, \texttt{indicated}) \times \\
&P_r(\texttt{NP(difference)} \mid \texttt{VP}, \texttt{VB}, \texttt{indicated}) \times \\
&P_r(\texttt{PP(in)} \mid \texttt{VP}, \texttt{VB}, \texttt{indicated}) \times \\
&P_r(\texttt{STOP} \mid \texttt{VP}, \texttt{VB}, \texttt{indicated})
\end{aligned}
$$

Independence Assumptions

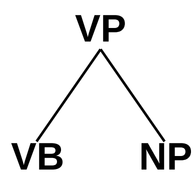
2.23%



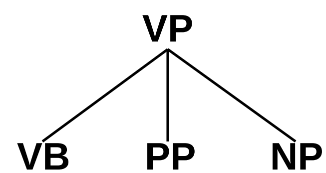
0.06%



60.8%



0.7%



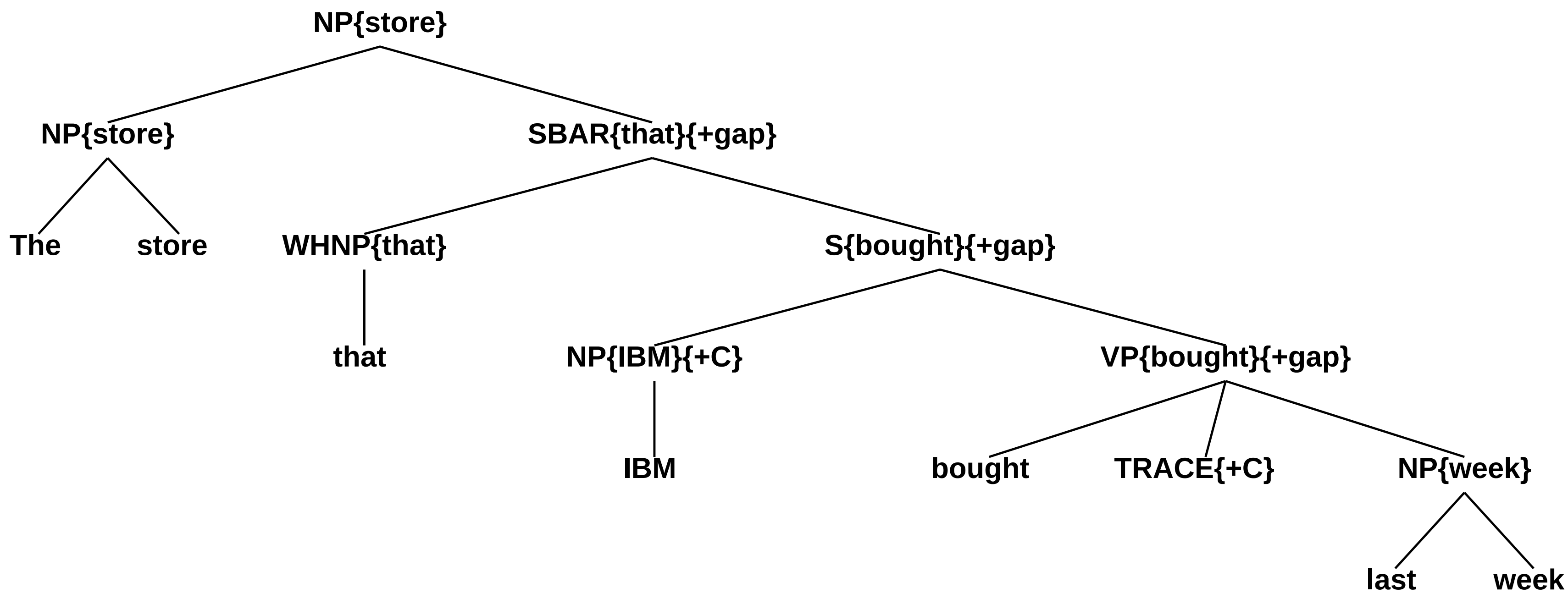
- Also violated in cases of coordination.
  e.g. `NP and NP`; `VP and VP`

- Processing facts like attach low in general.

- Also, English parse trees are generally right branching due to SVO structure.

- Language specific features are used heavily in the statistical model for parsing: cf. (Haruno et al. 1999)

Bilexical CFG with probabilistic ‘features’ (Collins 1997)



NP	$\longrightarrow$	[	NP{+H}	SBAR{+gap}		]
SBAR{+gap}	$\longrightarrow$	[	WHNP	S{+H}{+C}{+gap}		]
S{+gap}	$\longrightarrow$	[	NP{+C}	SBAR{+H}{+gap}		]
VP{+gap}	$\longrightarrow$	[	VB{+H}	TRACE{+C}	NP	]

Statistical Parsing Results using Lexicalized PCFGs

System	$\leq 40wds$ LP	$\leq 40wds$ LR	$\leq 100wds$ LP	$\leq 100wds$ LR
(Magerman 95)	84.9	84.6	84.3	84.0
(Collins 99)	88.5	88.7	88.1	88.3
(Charniak 97)	87.5	87.4	86.7	86.6
(Ratnaparkhi 97)			86.3	87.5
(Charniak 99)	90.1	90.1	89.6	89.5
(Collins 00)	90.1	90.4	89.6	89.9
Voting (HB99)	92.09	89.18		

Tree Adjoining Grammars

- Locality and independence assumptions are captured elegantly.

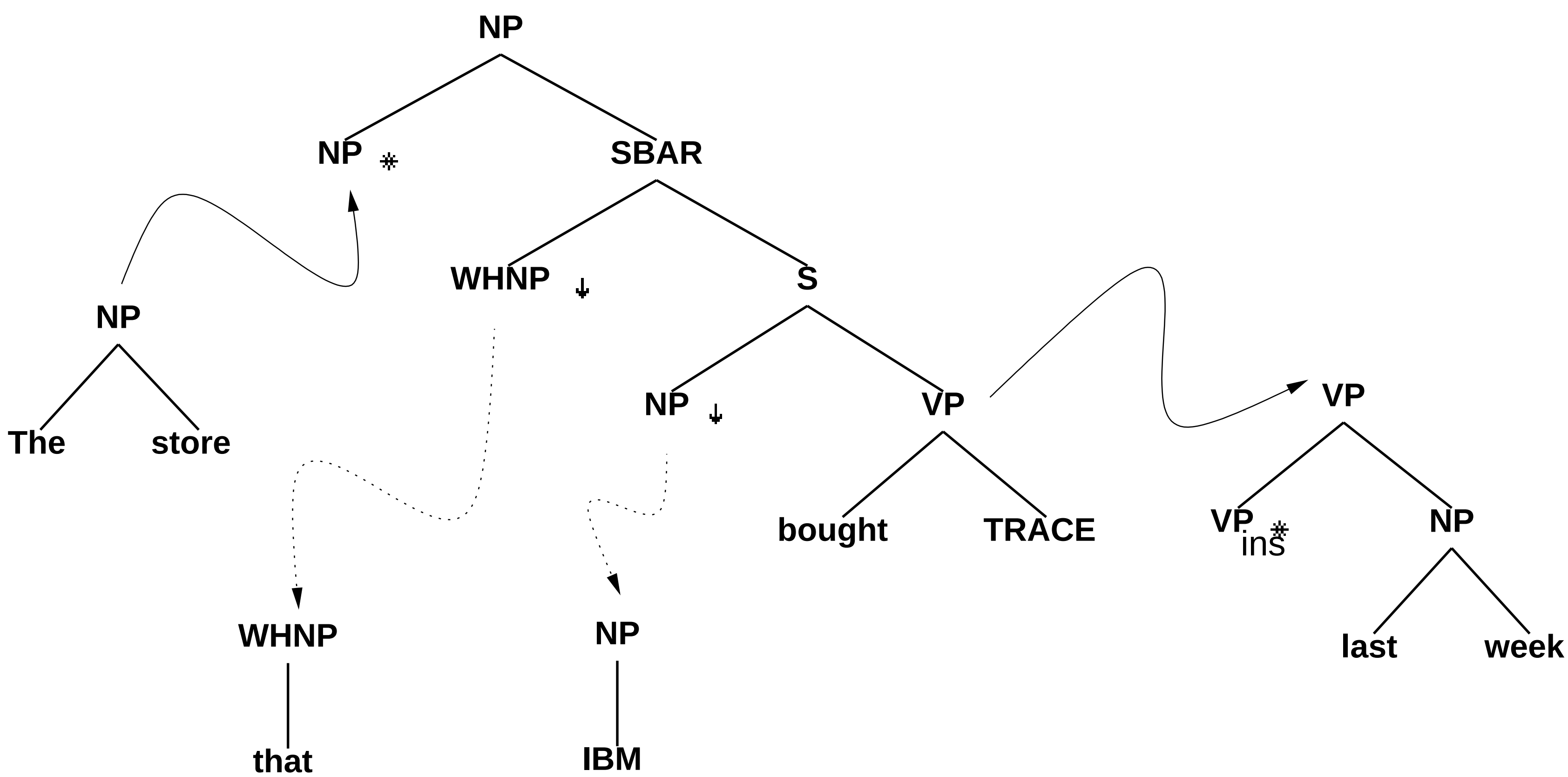
- Simple and well-defined probability model.

- Parsing can be treated in two steps:

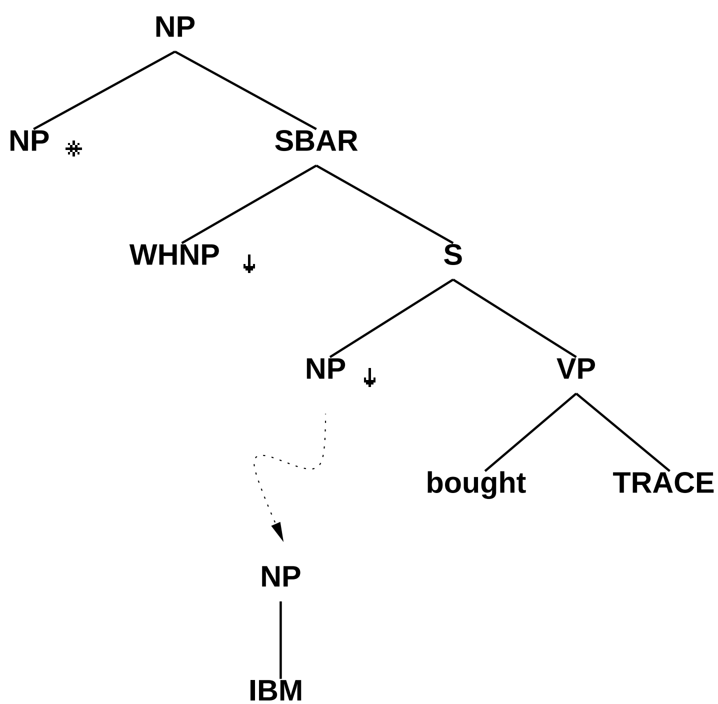
  1. Classification: structured labels (elementary trees) are assigned to each word in the sentence.

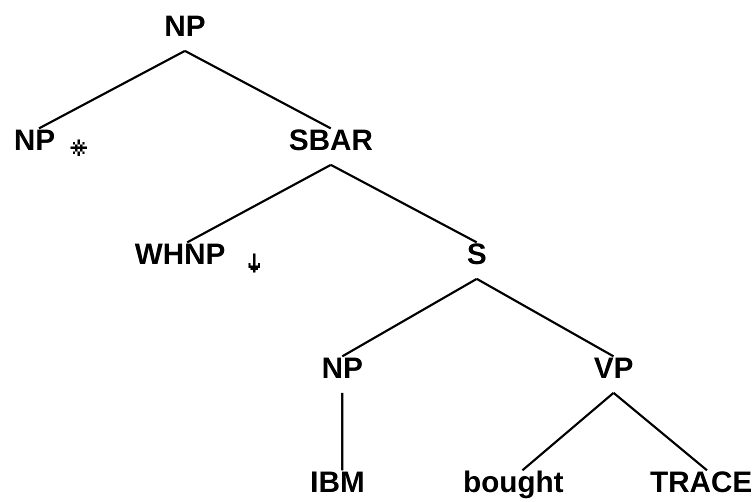
  2. Attachment: the elementary trees are connected to each other to form the parse.

Tree Adjoining Grammars: Different Modeling of Bilexical Dependencies



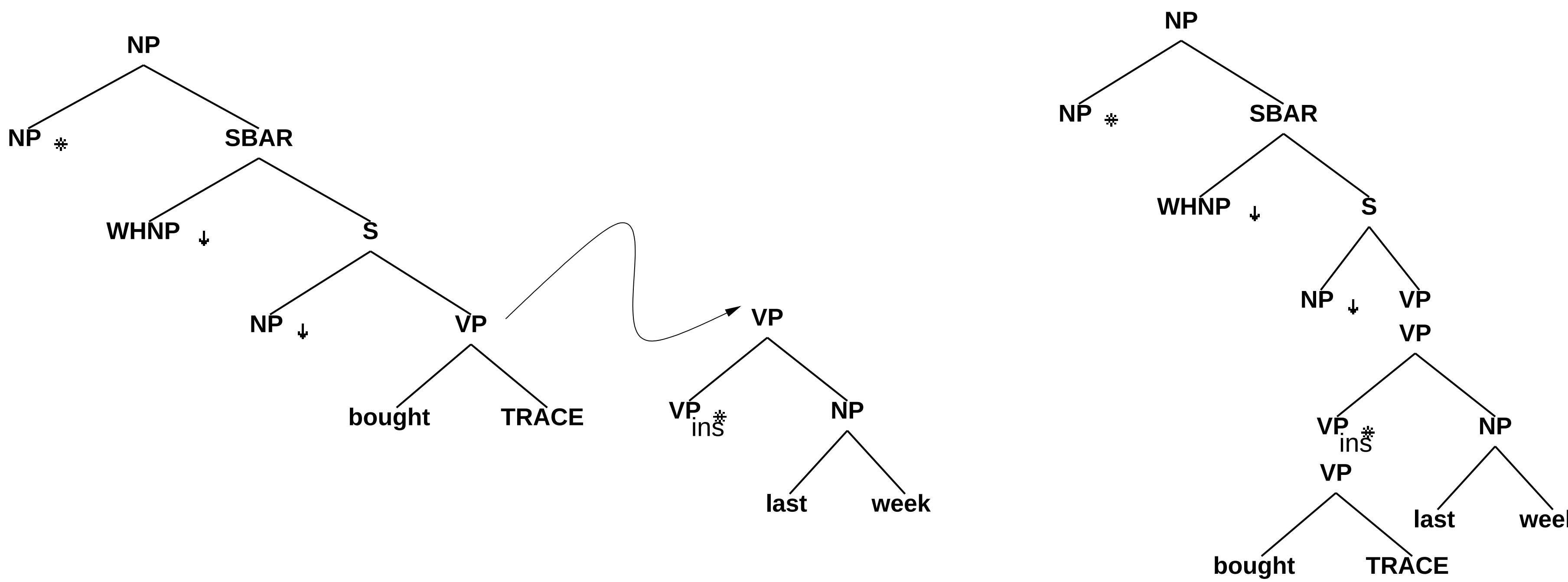
Probabilistic TAGs: Substitution





$$\sum_{t'} \mathcal{P}(t, \eta \rightarrow t') = 1$$

Probabilistic TAGs: Adjunction



$$\mathcal{P}(t, \eta \rightarrow NA) + \sum_{t'} \mathcal{P}(t, \eta \rightarrow t') = 1$$

Tree Adjoining Grammars

- Simpler model for parsing.
  Performance(Chiang 2000): 86.9% LR 86.6% LP ($\leq$ 40 words)
  Latest results: $\approx$ 88% average P/R

- Parsing can be treated in two steps:

  1. Classification: structured labels (elementary trees) are assigned to each word in the sentence.

  2. Attachment: Apply substitution or adjunction to combine the elementary trees to form the parse.

Tree Adjoining Grammars

- Produces more than the phrase structure of each sentence.

- A more embellished parse in which phenomena such as predicate-argument structure, subcategorization and movement are given a probabilistic treatment.

Practical Issues: Beam Thresholding and Priors

- Probability of nonterminal X spanning $j \ldots k$: $N[X, j, k]$

- Beam Thresholding compares $N[X, j, k]$ with every other Y where $N[Y, j, k]$

- But what should be compared?

- Just the *inside probability*: $P(X \stackrel{*}{\Rightarrow} t_j \ldots t_k)$?
written as $\beta(X, j, k)$

- Perhaps $\beta(\texttt{FRAG}, 0, 3) > \beta(\texttt{NP}, 0, 3)$, but `NP`s are much more likely than `FRAG`s in general

Practical Issues: Beam Thresholding and Priors

- The correct estimate is the *outside probability*:

$$P(S \stackrel{*}{\Rightarrow} t_1 \ldots t_{j-1}\ X\ t_{k+1} \ldots t_n)$$

  written as $\alpha(X, j, k)$

- Unfortunately, you can only compute $\alpha(X, j, k)$ efficiently after you finish parsing and reach $(S, 0, n)$

Practical Issues: Beam Thresholding and Priors

- To make things easier we multiply the prior probability $P(X)$ with the inside probability

- In beam Thresholding we compare every new insertion of X for span j, k as follows:
  Compare $P(X) \cdot \beta(X, j, k)$ with every Y $P(Y) \cdot \beta(Y, j, k)$

- Other more sophisticated methods are given in (Goodman 1997)

Chunking and Statistical Parsing

Anoop Sarkar
http://www.cis.upenn.edu/~anoop/
anoop@linc.cis.upenn.edu

Statistical Parsing Results using Lexicalized PCFGs

System	$\leq 40wds$ LP	$\leq 40wds$ LR	$\leq 100wds$ LP	$\leq 100wds$ LR
(Magerman 95)	84.9	84.6	84.3	84.0
(Collins 99)	88.5	88.7	88.1	88.3
(Charniak 97)	87.5	87.4	86.7	86.6
(Ratnaparkhi 97)			86.3	87.5
(Charniak 99)	90.1	90.1	89.6	89.5
(Collins 00)	90.1	90.4	89.6	89.9
Voting (HB99)	92.09	89.18		

Voting Methods: Parser Combination (Henderson 1999)

- Techniques for Combining Parsers
  - Parse Hybridization: Best constituents selected from each parser
    - Constituent Voting
    - Naive Bayes
  - Parser Switching: Learn best parser for a given sentence
    - Similarity Switching
    - Naive Bayes

Voting Methods: Naive Bayes

- $\pi(c) = 1$ when constituent c should be included in the output parse
- $M_i(c) = 1$ indicates that parser i suggests that constituent c should be in the output parse
- The model picks likely constituents ($P > 0.5$) to be in the output parse

Voting Methods: Naive Bayes

$$\underset{\pi(c)}{\text{arg max}}\ P(\pi(c) \mid M_1(c) \ldots M_k(c)) =$$

$$\underset{\pi(c)}{\text{arg max}}\ \frac{P(M_1(c) \ldots M_k(c) \mid \pi(c)) \cdot P(\pi(c))}{P(M_1(c) \ldots M_k(c))}$$

$$\underset{\pi(c)}{\text{arg max}}\ P(\pi(c)) \cdot \prod_{i=1}^{k} \frac{P(M_i(c) \mid \pi(c))}{P(M_i(c))}$$

$$\underset{\pi(c)}{\text{arg max}}\ P(\pi(c)) \cdot \prod_{i=1}^{k} P(M_i(c) \mid \pi(c))$$

Voting Methods: Parser Combination (Henderson 1999)

Reference/System	LP	LR
Average Individual Parser	87.14	86.91
Best Individual Parser	88.73	88.54
Parser Switching Oracle	93.12	92.84
Maximum Precision Oracle	100.00	95.41
Similarity Switching	89.50	89.88
Constituent Voting	92.09	89.18

Other Parser Combination Methods

- Combining the same statistical parser by training on various subsets of the training data
- Eliminating noise in the annotated data
  also see (Abney et al. 2000)
- Bagging and Boosting statistical parsers
  (Henderson and Brill 2000)

Discriminative Methods: Parse Reranking (Collins 2000)

- Expts in (Ratnaparkhi 1999) showed that if the parser was allowed upto 20 chances to get the best parse, accuracy could be as high as 96% avg LP/LR

- 20 ranked guesses are easy to produce: $T_{\textit{best}} \ldots T_{\textit{best}-19}$

- Automatic reranking can be used to produce a more accurate parser

- (Collins 2000) showed that reranking can improve parsing accuracy

Discriminative Methods: Parse Reranking (Collins 2000)

- $x_{i,j}$ is the jth parse of the ith sentence
- $L(x_{i,j}) = \text{log}Q(x_{i,j})$: the log probability of a parse
- The task is to learn a ranking function $F(x_{i,j})$
- Baseline ranking: $F(x_{i,j}) = L(x_{i,j})$

Discriminative Methods: Parse Reranking (Collins 2000)

- $\alpha = \{\alpha_0, \ldots, \alpha_m\}$
- $F(x_{i,j}, \alpha) = \alpha_0 L(x_{i,j}) + \sum_{s=1}^{m} \alpha_s h_s(x_{i,j})$
- $h_s(x) = \begin{cases} 1 & \text{if } x \text{ contains rule } \mathtt{S} \longrightarrow \mathtt{NP\ VP} \\ 0 & \text{otherwise} \end{cases}$
- Minimize *ranking error rate*: number of times a lower scoring parse is ranked above best parse

Discriminative Methods: Parse Reranking (Collins 2000)

- (Collins 2000) gives various discriminative methods to minimize the ranking error rate

- Various non-local features can now be used — previously unavailable within a top-down generative model

- Parsing performance improved with a 13% decrease in the labeled constituent error rate

- LP 90.4% LR 90.1% ($\leq$ 40 wds) and
  LP 89.6% LR 89.9% ($\leq$ 100 wds)

Discriminative Methods: Parse Reranking (Collins 2000)

Rules context-free rules, e.g. `VP` $\longrightarrow$ `PP VBD NP NP SBAR`

Bigrams Pairs of non-terminals to the left and right of the head, e.g. `(Right, VP, NP, NP)`,`(Right, VP, NP, SBAR)`,`(Right, VP, PP, STOP)` and `(Left, VP, PP, STOP)`

Grandparent Rules Same as **Rules** including the parent of the LHS

Grandparent Bigrams Same as **Bigrams** including parent of the LHS

Lexical Bigrams Same as **Bigrams** but with lexical heads

Discriminative Methods: Parse Reranking (Collins 2000)

Two-level Rules Same as **Rules** but with the LHS expanded to an entire rule

Two-level Bigrams Same as **Bigrams** but with the LHS expanded to an entire rule

Trigrams All trigrams within a rule, e.g. `(VP, STOP, PP, VBD!)`

Head-Modifiers All head-modifier pairs with grandparent non-terminals, e.g. `(Left, VP, VBD, PP)`

Discriminative Methods: Parse Reranking (Collins 2000)

PPs Lexical trigrams for PPs, e.g. (NP (NP the president) (PP of (NP IBM))) produces `(NP, NP, PP, NP, president, of, IBM)` as well as `(NP, NP, PP, NP, of, IBM)`

Distance Head-Modifiers Distance between head words in a CFG attachment rule

Closed-class Lexicalization Add closed-class words to non-lexicalized non-terminals in above rules (except last 3)

Limited labeled data: The EM algorithm

- Unsupervised learning using the Maximum Likelihood principle
- Find parameter values that reduce the training set entropy
- For PCFGs: the Inside-Outside algorithm
- *inside probability*: $P(X \stackrel{*}{\Rightarrow} t_j \ldots t_k)$
  written as $\beta(X, j, k)$
- *outside probability*: $P(S \stackrel{*}{\Rightarrow} t_1 \ldots t_{j-1}\ X\ t_{k+1} \ldots t_n)$
  written as $\alpha(X, j, k)$

Limited labeled data: The Inside-Outside algorithm

- Initialize PCFG parameters to random values
- For training set, compute for each non-terminal X calculate values of α and β
- For each rule use values of α and β to compute new expected parameter values by using the maximum likelihood principle
  e.g. c(X $\rightarrow$ A) = 80 and c(X $\rightarrow$ A B) = 20 then P(X $\rightarrow$ A) = 0.8

Limited labeled data: The Inside-Outside algorithm

- Iterate to convergence: Theorem by (Dempster, Laird and Rubin 1977) states that likelihood is always non-decreasing
- Theoretically well motivated, but computationally expensive $\mathcal{O}(n^3)$ per sentence in each iteration for PCFGs

Case Study in Unsupervised Methods: POS Tagging

- POS Tagging: finding categories for words
- *. . . the stocks* $\boxed{\textit{rose}}$/V *. . .* vs. *. . . a* $\boxed{\textit{rose}}$/N *bouquet . . .*
- Tag dictionary: $\boxed{\textit{rose}}$*: N, V*

  and nothing else

Case Study: Unsupervised POS Tagging

- (Cutting et al. 1992) The Xerox Tagger: used HMMs with hand-built tag dictionaries. High performance: 96% on Brown

- (Merialdo 1994; Elworthy 1994) used varying amounts of labeled data as seed information for training HMMs.

  Conclusion: HMMs do not effectively combine labeled and unlabeled data

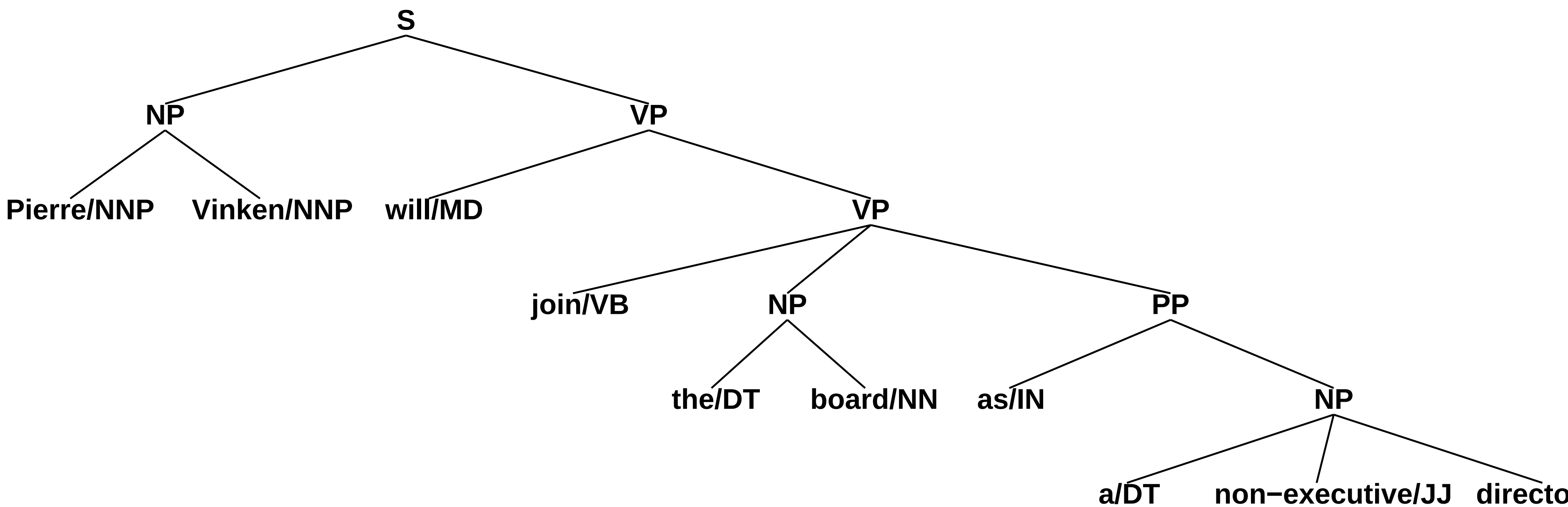
- (Brill 1997) aggressively used tag dictionaries taken from labeled data to train an unsupervised POS tagger. c.f. text classification results

  Performance: 95% on WSJ. Approach does not easily extend to parsing: no notion of tag dictionary.

Co-Training (Blum and Mitchell 1998; Yarowsky 1995)

- Pick two "views" of a classification problem.
- Build separate models for each of these "views" and train each model on a small set of labeled data.
- Sample an unlabeled data set and to find examples that each model independently labels with high confidence. (Nigam and Ghani 2000)
- Pick confidently labeled examples.
  (Collins and Singer 1999; Goldman and Zhou 2000); Active Learning
- Each model labels examples for the other in each iteration.

Pierre Vinken will join the board as a non-executive director



Recursion in Parse Trees

- Usual decomposition of parse trees:

  S(join) $\rightarrow$ NP(Vinken) VP(join)

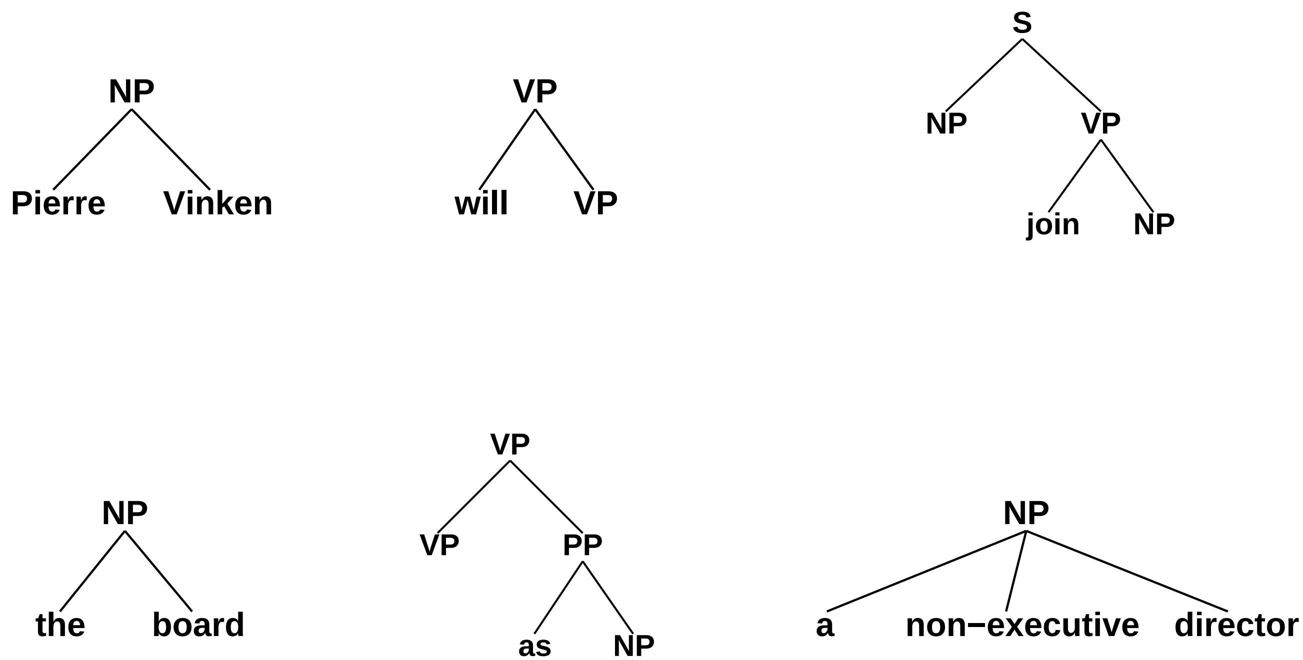
  NP(Vinken) $\rightarrow$ Pierre Vinken

  VP(join) $\rightarrow$ will VP(join)

  VP(join) $\rightarrow$ join NP(board) PP(as)

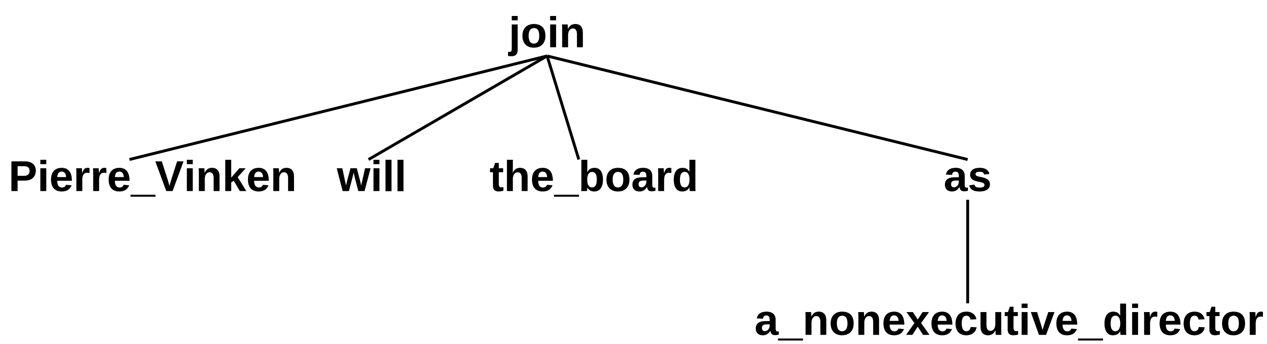
  ...

Parsing as Tree Classification and Attachment: (Srinivas 1997; Xia 2000)



Model H1: $\mathcal{P}(T_i \mid T_{i-2}T_{i-1}) \times \mathcal{P}(w_i \mid T_i)$

Parsing as Tree Classification and Attachment



Model H2: $\mathcal{P}(\text{TOP} = w, T) \times \Pi_i \mathcal{P}(w_i, T_i \mid \eta, w, T)$

The Co-Training Algorithm

1. Input: *labeled* and *unlabeled*
2. Update cache
   - Randomly select sentences from *unlabeled* and refill *cache*
   - If *cache* is empty; exit
3. Train models H1 and H2 using *labeled*
4. Apply H1 and H2 to cache.
5. Pick most probable n from H1 (run through H2) and add to *labeled*.
6. Pick most probable n from H2 and add to *labeled*
7. $n = n + k$; Go to Step 2

Results (Sarkar 2001)

- *labeled* was set to Sections 02-06 of the Penn Treebank WSJ (9625 sentences)

- *unlabeled* was 30137 sentences (Section 07-21 of the Treebank stripped of all annotations).

- A tree dictionary of all lexicalized trees from *labeled* and *unlabeled*.
  Similar to the approach of (Brill 1997)
  Novel trees were treated as unknown tree tokens

- The *cache* size was 3000 sentences.

Results

- Test set: Section 23
- Baseline Model was trained only on the *labeled* set:
  and Labeled Bracketing Precision = 72.23% Recall = 69.12%
- After 12 iterations of Co-Training:
  Labeled Bracketing Precision = 80.02% Recall = 79.64%

Limited labeled data: Active Learning and Sample Selection

- Active learning or sample selection aims to discover which data when annotated would be most informative
- Out of a large pool of text, what subset should be annotated to create a training set?
- A better approach than blindly labeling an arbitrary set of data.
- Drawbacks of sample selection: biased to a particular learning model
- Common answer: Committee of learners, e.g. (Engelson and Dagan 1996) for POS Tagging

Limited labeled data: Sample Selection (Hwa 2001)

Algorithm:

U is a set of unlabeled candidates
L is a set of labeled training examples
M is the current model
Select(n, U, M, f):
returns n candidates picked from U based on an evaluation function f and a model M

Initialize:

$M \leftarrow$ Train(L)

Repeat:

N $\leftarrow$ Select(n, U, M, f)
$U \leftarrow U$ - N
$L \leftarrow L \cup$ Label(N)
$M \leftarrow$ Train(L)

Until:

$U = \emptyset$ or *human stops*

Limited labeled data: Sample Selection (Hwa 2001)

- Evaluation function f should denote the uncertainty of a parser for a particular sentence

- Tree entropy of a parser M for sentence u:

$$TE(u, M) = -\sum_{t \in \mathcal{T}} P(t \mid u, M) \cdot \mathit{log}_2 P(t \mid u, M)$$

- Evaluation function: $f_{te}(u, M) = \frac{TE(u,M)}{\mathit{log}_2 |\mathcal{T}|}$

- Experiment: Trained the Collins parser with same accuracy while reducing number of annotated constituents by 23%

Information Extraction and Parsing

- Parsers trained on annotated parse trees that encode semantic values have been used for dialog projects like `ATIS`, `How May I Help You?` and `Communicator`

- MUC-7 tasks like Template Element (TE) and Template Relations (TR) have been typically performed using shallow parsing methods using finite-state techniques due to a more expressive domain

- Statistical Parsing has recently been applied to this domain

Information Extraction: Template Element (TE)

```
<entity id="5" ent_type="person">
  <ent_name value="Nance"/>
  <ent_name value="Nancy Hedberg"/>
  <ent_descriptor value="paid consultant to ABC news"/>
</entity>

<entity id="6" ent_type="organization">
  <ent_name value="ABC"/>
  <ent_name value="ABC news"/>
</entity>
```

Information Extraction: Template Relations (TR)

```
<relation type="employee_of">
  <entity arg="0" type="person" value="5"/>
  <entity arg="1" type="organization" value="6"/>
</relation>
```

Information Extraction: Combining Syntax and Semantics (Miller et al 2000)

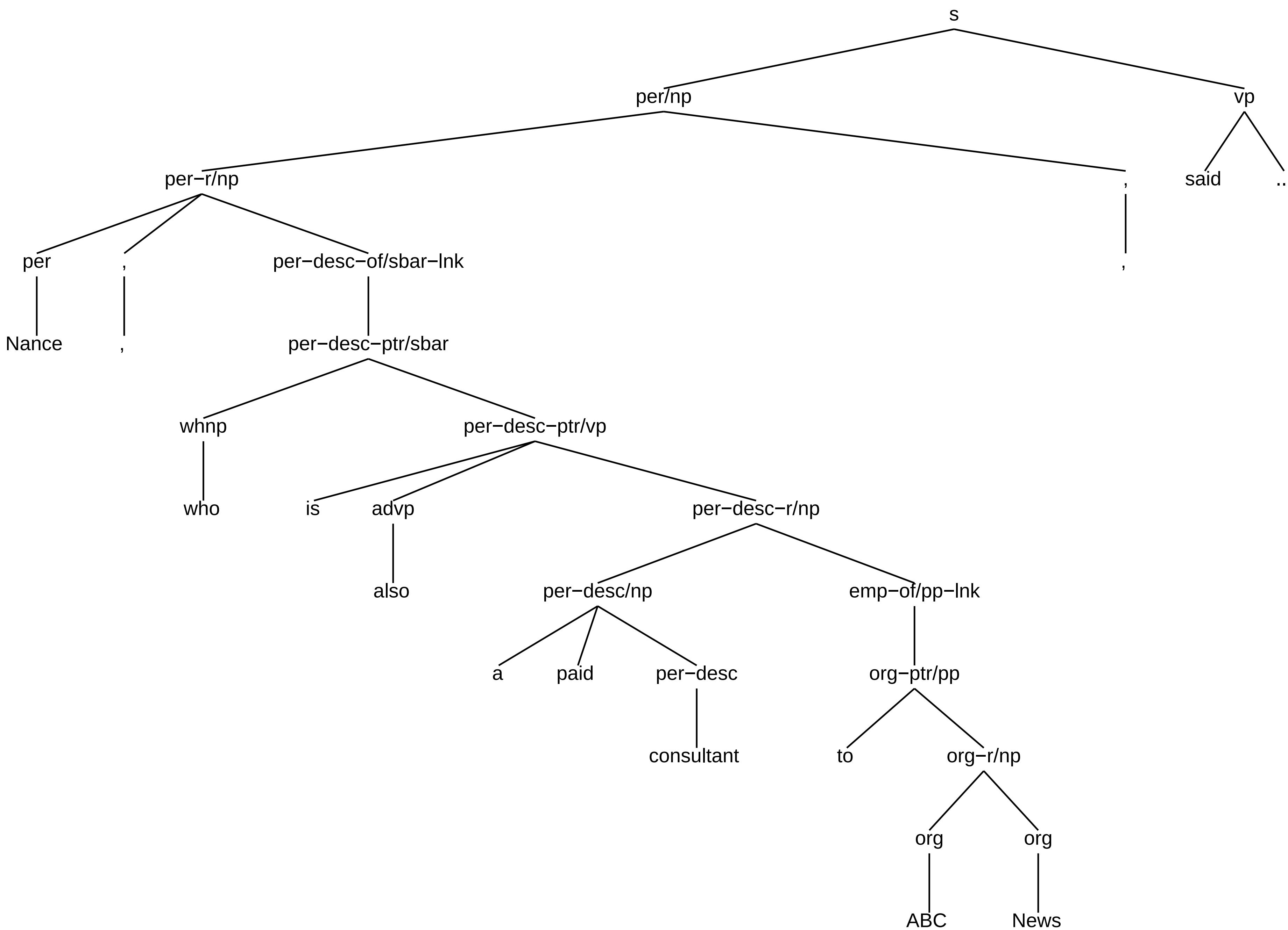
- Train a statistical parser on a general domain

- Annotate a small set L of sentences with the expected output relations using domain-specific semantic values

- Parse L using the statistical parser to find parses consistent with the semantic annotation and combine the syntactic analysis and the semantic annotation

  (based on a crossing bracket measure)

- Retrain a new statistical parser that will produce the combined output


s
per/np
vp
per−r/np
,
said
...
per
,
per−desc−of/sbar−lnk
,
Nance
,
per−desc−ptr/sbar
whnp
per−desc−ptr/vp
who
is
advp
per−desc−r/np
also
per−desc/np
emp−of/pp−lnk
a
paid
per−desc
org−ptr/pp
consultant
to
org−r/np
org
org
ABC
News

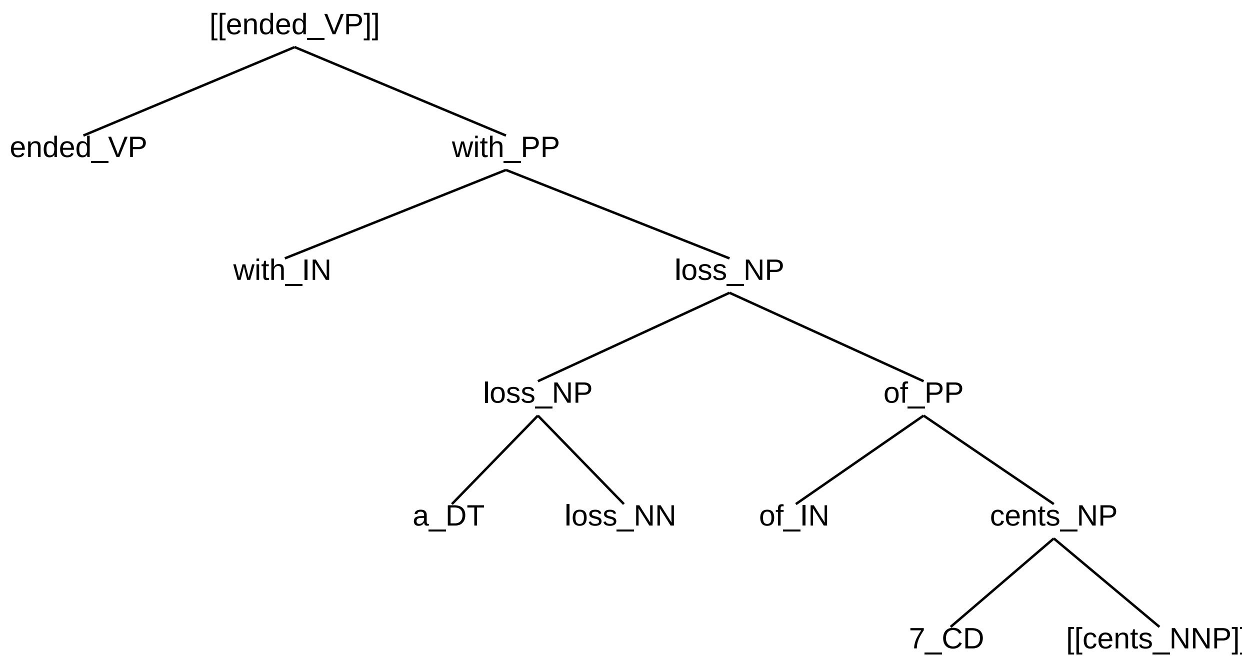
Information Extraction: Combining Syntax and Semantics (Miller et al 2000)

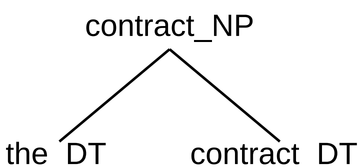
Task	Recall	Precision
Entities (TE)	83.0	84.0
Relations (TR)	64.0	81.0

Applications: Language Modeling (Chelba and Jelinek 1998)

- Speech recognition requires a model for the most likely next token
- Language model: $P(t_k \mid t_1 \ldots t_{k-1})$
- Parsing a word lattice: spans in a string become states in a finite-state automaton

The contract ended with a loss of 7 cents [[after]]





Applications: Language Modeling

- (Chelba and Jelinek 1998):

Iteration/Baseline	Test set Perplexity	Interpolation with 3-gram
Baseline Trigram	167.14	167.14
Iteration 0	167.47	152.25
Iteration 3	158.28	**148.90**

- (Wu and Khudanpur 1999):

Model	Perplexity	WER
Baseline Trigram	79.0	38.5
N-gram + Syntactic	73.5	37.7