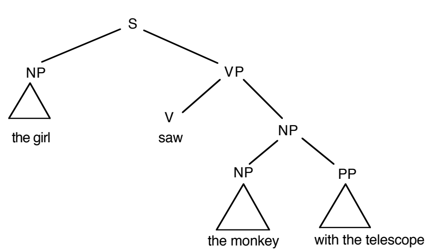
Head-Driven Statistical Models for Natural Language Parsing

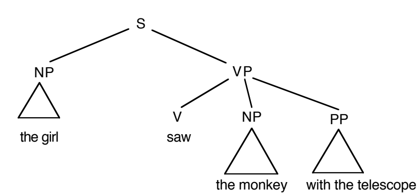
Sebastian Roth
Sebastian.Roth@uni-konstanz.de

The girl saw the monkey with the telescope.

The girl saw the monkey with the telescope.



The girl saw the monkey with the telescope.



The girl saw the monkey with the telescope.

Problem: **Ambiguity**

Combinatorial effects mean that even short sentences can receive a considerable number of parses under a wide-coverage grammar.

The girl saw the monkey with the telescope.

Problem: **Ambiguity**

Combinatorial effects mean that even short sentences can receive a considerable number of parses under a wide-coverage grammar.

Solution: **Statistical Parser**

Statistical parsing approaches tackle the ambiguity problem by assigning a probability to each parse tree, thereby ranking competing trees in order of plausibility.

Statistical Parsing:
How to do it ?

Context-Free Grammar (CFG)

Hopcroft and Ullman (1979)

A context-free grammar is defined by a 4-Tuple (N, Σ, A, R):

N: *Set of nonterminal symbols*
Σ: *Alphabet*
A: *Distinguished start symbol (element of N)*

R: *Finite set of rules of the form X -> ß where*

$$X \in N, \beta \in (N \cup \Sigma)$$

Exemplary CFG:

Phrase Structure Grammar

N *(non-terminals)*:	NP, VP, PP, AP, S, ...
Σ *(alphabet)*:	N,V,P,A,Det, Adv, ...
A *(start symbol)*:	S
R *(rules)*:	S --> NP VP
	NP --> Det N
	VP --> V NP (PP) ...

D: *Function assigning probabilities to each rule in* R

This function expresses the probability *P* that the given non-terminal X will be expanded to the sequence β.

$$P(X \rightarrow \beta)$$

or

$$P(X \rightarrow \beta \mid X)$$

This is the conditional probability of a given expansion given the left-hand-side non-terminal X.

If we consider all the possible expansions of a non-terminal, the sum of their probabilities must be 1.

Probabilistic Context-Free Grammar (PCFG)

A Probabilistic CFG is defined by a 5-Tuple (N, Σ, A, R, D):

N: *Set of nonterminal symbols*
Σ: *Alphabet*
A: *Distinguished start symbol (element of N)*

R: *Finite set of rules of the form X -> ß where*

$$X \in N, \beta \in (N \bigcup \Sigma)$$

D: *Function assigning probabilities to each rule in* R

A PCFG assigns a probability to each parse-tree *T* (i.e. each derivation) of a sentence *S*.

The probability of a given tree-sentence-pair *(T,S)* derived by *n* applications of context-free rules LHS_i --> RHS_i under the PCFG is

$$P(T,S) = \prod_{i=1}^{n} P(RHS_i \mid LHS_i)$$

The resulting probability is both the joint probability of the parse and the sentence, and also the probability of the parse P(T), since the joint probability is defined as:

P(T,S) = P(T) P(S|T)

and P(S|T) = 1 since a parse tree includes all words of the sentence.

Assigning Probabilities

How is it done ?

Now that we have our PCFG...

Assigning Probabilities

How is it done ?

We need to assign a probability to each possible expansion β of each non-terminal X.

Simplest way:

Take a treebank (such as Penn TB), count the number of times every expansion occurs, and normalize !

$$P(X \rightarrow \beta \mid X) = \frac{Count(X \rightarrow \beta)}{\sum_{\gamma} Count(X \rightarrow \gamma)} = \frac{Count(X \rightarrow \beta)}{Count(X)}$$

Now that we have our PCFG...

...let‘s have a look at the problems we face !

Now that we have our PCFG...

...let's have a look at the problems we face !

PCFGs make a fundamental independence assumption:

The expansion of any one non-terminal is independent of the expansion of any other non-terminal.

Each PCFG rule is assumed to be independent of any other rule, thus rule probabilities are just multiplied together.

Now that we have our PCFG...

...let's have a look at the problems we face !

PCFGs make a fundamental independence assumption:

The expansion of any one non-terminal is independent of the expansion of any other non-terminal.

Each PCFG rule is assumed to be independent of any other rule, thus rule probabilities are just multiplied together.

This makes it impossible to model:

- structural dependencies
- lexical dependencies

Structural dependencies:

(1) (a) **She**'s able to take her baby to work with her

(b) Uh, **my wife** worked until we had a family.

Structural dependencies:

(1) (a) **She**'s able to take her baby to work with her

(b) Uh, **my wife** worked until we had a family.

```
Switchboard Corpus:

Of 31,021 subjects of declarative sentences,
91% are pronouns
9% are lexical
```

Structural dependencies:

(1) (a) **She**'s able to take her baby to work with her

(b) Uh, **my wife** worked until we had a family.

```
Switchboard Corpus:

Of 31,021 subjects of declarative sentences,
91% are pronouns
9% are lexical
```

(2) (a) Some laws absolutely prohibit **it**.

(b) All the people signed **confessions**.

Structural dependencies:

(1) (a) **She**'s able to take her baby to work with her

(b) Uh, **my wife** worked until we had a family.

Switchboard Corpus:

Of 31,021 subjects of declarative sentences,
91% are pronouns
9% are lexical

(2) (a) Some laws absolutely prohibit **it**.

(b) All the people signed **confessions**.

```
Of 7,489 direct objects,
34% are pronouns
66% are lexical
```

Switchboard Corpus:

Of 31,021 subjects of declarative sentences,
91% are pronouns
9% are lexical

Of 7,489 direct objects,
34% are pronouns
66% are lexical

These dependencies would be captured if

P(NP -> Pronoun)   vs   P(NP -> Det Noun)

were dependent on whether the NP is subject or an object.

Switchboard Corpus:

Of 31,021 subjects of declarative sentences,
91% are pronouns
9% are lexical

Of 7,489 direct objects,
34% are pronouns
66% are lexical

These dependencies would be captured if

P(NP -> Pronoun)   vs   P(NP -> Det Noun)

were dependent on whether the NP is subject or an object.

This is the kind of probabilistic dependency that a PCFG does not allow !

Lexical dependencies:

(3) (a) Moscow sent more than 100,000 soldiers into Afghanistan.

Lexical dependencies:

(3) (a) Moscow sent more than 100,000 soldiers into Afghanistan.

PP [into Afghanistan] can be attached...

- either to the NP [more than 100,000 soldiers]
- or to the VP headed by [sent]

Lexical dependencies:

(3) (a) Moscow sent more than 100,000 soldiers into Afghanistan.

PP [into Afghanistan] can be attached...

- either to the NP [more than 100,000 soldiers]
- or to the VP headed by [sent]

NP --> NP PP (NP-attachment) (3a)

Lexical dependencies:

(3) (a) Moscow sent more than 100,000 soldiers into Afghanistan.
(b) Moscow sent more than 100,000 soldiers into Afghanistan.

PP [into Afghanistan] can be attached...

- either to the NP [more than 100,000 soldiers]
- or to the VP headed by [sent]

NP --> NP PP (NP-attachment) (3a)

VP --> NP PP (VP-attachment) (3b)

Lexical dependencies:

(3) (a) Moscow sent more than 100,000 soldiers into Afghanistan.
(b) Moscow sent more than 100,000 soldiers into Afghanistan.

PP [into Afghanistan] can be attached...

- either to the NP [more than 100,000 soldiers]
- or to the VP headed by [sent]

NP --> NP PP (NP-attachment) (3a) ...67%

VP --> NP PP (VP-attachment) (3b) ...33%

Distribution of NP/VP-attachment in the 13 million words AP newswire corpus (Hindle&Rooth 1991)

How do we solve these problems ?

Lexical dependencies:

(3) (a) Moscow sent more than 100,000 soldiers into Afghanistan.
(b) Moscow sent more than 100,000 soldiers into Afghanistan.

PP [into Afghanistan] can be attached...

- either to the NP [more than 100,000 soldiers]
- or to the VP headed by [sent]

NP --> NP PP (NP-attachment) (3a) ...67%

VP --> NP PP (VP-attachment) (3b) ...33%

Correct attachment, as *send* subcategorizes for a destination.

PCFG does not know that !

How do we solve these problems ?

Lexicalized PCFG

How do we solve these problems ?

Lexicalized PCFG

Each PCFG rule is augmented to identify one RHS constituent to be its head daughter.

The headword for a node is then set to the headword of its head daughter.

All non-terminals are now of the format X(x) with

x = lexical information on the head daughter

We can think of a lexicalized grammars as a simple context-free grammar with a lot more rules:

CFG Rule: VP --> V NP PP

Lexicalized Grammar Rules:

VP(throw) --> V(throw) NP(ball) PP(into)

VP(send) --> V(send) NP(soldiers) PP(into)

VP(send) --> V(send) NP(gift) PP(to)

VP(put) --> V(put) NP(ball) PP(below)

... and many many many more

Problem:
How do we assign probabilities to all those rules ?

There is no corpus that is large enough to train such probabilities.
Count(X(x) -->ß) would be ZERO for most of the rules in our set.

Problem:
How do we assign probabilities to all those rules ?

There is no corpus that is large enough to train such probabilities.
Count(X(x) -->ß) would be ZERO for most of the rules in our set.

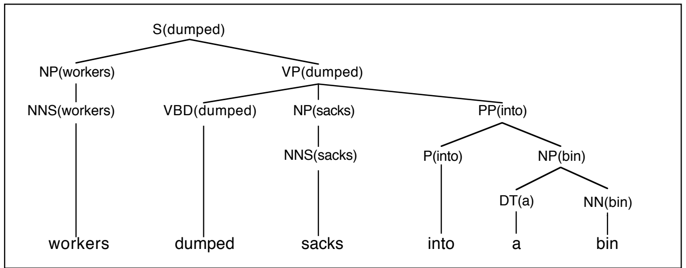
Solution:
Introduce simplifying independence assumptions

It is necessary to find a solution in between completely lexicalized rules and the complete lexical independence of the standard PCFG.

This is where the various models of lexicalized PCFGs differ in the way which independence asssumptions they make.

Following example is a simplified version of the statistical models which Charniak(1997) or Collins(1999/2003) use.

(see Jurafsky & Martin 2000, p. 460)



(4) (a) A lexicalized tree from Collins (1999)

In a standard (non-lexicalized) PCFG the probability of X being expanded by rule (X --> β) was conditioned only by the syntactic category of X:

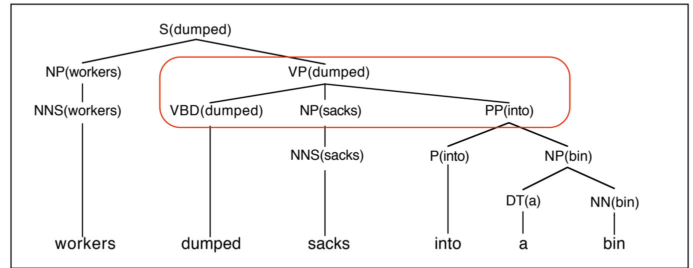
$$P(X \rightarrow \beta \mid X)$$

Now let's introduce another conditioning factor:
The headword of node X (*head(X)*)

$$P(X \rightarrow \beta \mid X, head(X))$$

For the rule VP --> VBD NP PP we compute:

$$P(VP \rightarrow VBD \quad NP \quad PP \mid VP, dumped)$$



(4) (a) A lexicalized tree from Collins (1999)

In a standard (non-lexicalized) PCFG the probability of X being expanded by rule (X --> β) was conditioned only by the syntactic category of X:

$$P(X \rightarrow \beta \mid X)$$

Now let's introduce another conditioning factor:
The headword of node X (*head(X)*)

$$P(X \rightarrow \beta \mid X, head(X))$$

For the rule VP --> VBD NP PP we compute:

$$P(VP \rightarrow VBD \quad NP \quad PP \mid VP, dumped)$$

Now, want to know the actual probability of a VP headed by *dumped* being expanded as *VBD NP PP*.
We'll get that by training our head-driven lexicalized PCFG on the Brown Corpus:

For the rule VP --> VBD NP PP we compute:

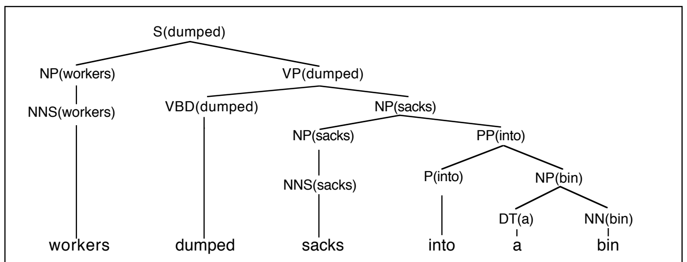
$$P(VP \rightarrow VBD \quad NP \quad PP \mid VP, dumped)$$

Now, want to know the actual probability of a VP headed by *dumped* being expanded as *VBD NP PP*.
We'll get that by training our head-driven lexicalized PCFG on the Brown Corpus:

$$P(VP \rightarrow VBD \quad NP \quad PP \mid VP, dumped) =$$

$$= \frac{Count(VP(dumped) \rightarrow VBD \quad NP \quad PP)}{\sum_{\beta} Count(VP(dumped \rightarrow \beta)} =$$

= ?



(4) (b) An incorrect parse tree of (4a) from Collins (1999)

How about this tree ?
This parse is apparently incorrect, so we want to know what probability it gets assigned:

For the rule VP --> VBD NP PP we compute:

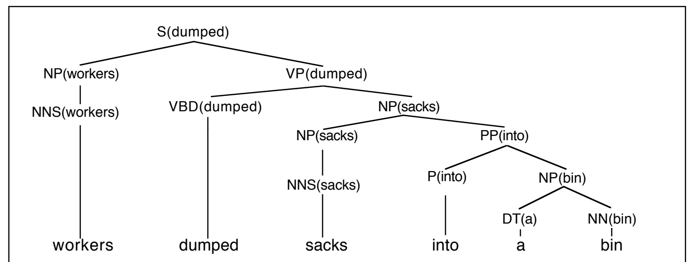
$$P(VP \rightarrow VBD \quad NP \quad PP \mid VP, dumped)$$

Now, want to know the actual probability of a VP headed by *dumped* being expanded as *VBD NP PP*.
We'll get that by training our head-driven lexicalized PCFG on the Brown Corpus:

$$P(VP \rightarrow VBD \quad NP \quad PP \mid VP, dumped) =$$

$$= \frac{Count(VP(dumped) \rightarrow VBD \quad NP \quad PP)}{\sum_{\beta} Count(VP(dumped \rightarrow \beta)} =$$

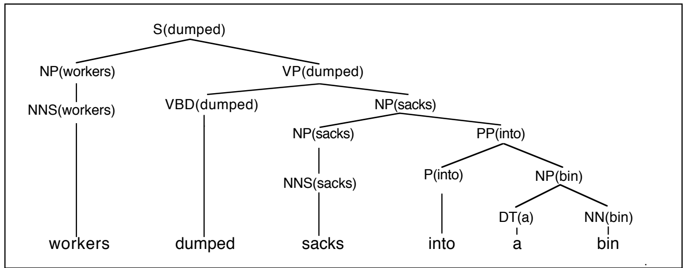
$$= \frac{6}{9} = 0.666666666666666666666666666666666666...$$



(4) (b) An incorrect parse tree of (4a) from Collins (1999)

How about this tree ?
This parse is apparently incorrect, so we want to know what probability it gets assigned:

$$P(VP \rightarrow VBD \quad NP \mid VP, dumped) = \frac{Count(VP(dumped) \rightarrow VBD \quad NP)}{\sum_{\beta} Count(VP(dumped) \rightarrow \beta)} =$$



(4) (b) An incorrect parse tree of (4a) from Collins (1999)

How about this tree ?
This parse is apparently incorrect, so we want to know what probability it gets assigned:

$$P(VP \rightarrow VBD \quad NP \mid VP, dumped) = \frac{Count(VP(dumped) \rightarrow VBD \quad NP)}{\sum_{\beta} Count(VP(dumped) \rightarrow \beta)} =$$

= 0/9 = 0

Conditioning on headwords lets us capture subcategorization information and bears good results.

But we want more....

Conditioning on headwords lets us capture subcategorization information and bears good results.

But we want more....

It would be great if we had a way of computing the probability of a certain head.

In *VP,dumped --> VBD NP PP* it is completely irrelevant what kind of PP we have.

Conditioning on headwords lets us capture subcategorization information and bears good results.

But we want more....

It would be great if we had a way of computing the probability of a certain head.

In *VP,dumped --> VBD NP PP* it is completely irrelevant what kind of PP we have.

„What is the probability that a PP whose mother‘s head is *dumped* has the head *into* ?“

$$P(head(X) \mid X, head(mother(X)))$$

Results from the Brown Corpus:

$$P(\text{into} \mid PP, dumped) = \frac{Count(X(dumped) \rightarrow ...PP(\text{into})...}{\sum_{\beta} Count(X(dumped) \rightarrow ...PP...)} = \frac{2}{9} = .22$$

Results from the Brown Corpus:

$$P(\text{into} \mid PP, dumped) = \frac{Count(X(dumped) \rightarrow ...PP(\text{into})...)}{\sum_{\beta} Count(X(dumped) \rightarrow ...PP...)} = \frac{2}{9} = .22$$

Let's check the results for the incorrect parse (PP(*into*) attached to *sacks*):

$$P(\text{into} \mid PP, sacks) = \frac{Count(X(sacks) \rightarrow ...PP(\text{into})...)}{\sum_{\beta} Count(X(sacks) \rightarrow ...PP...)} = \frac{0}{0} = ?$$

Again, head probabilities correctly predict that *dumped* is more likely to be modified by *into* than is *sacks*.

Taking these dependencies into account, our final equation for calculating the probability of a whole parse tree looks like this:

$$P(T,S) = \prod_{n \in T} P(X \rightarrow \beta \mid X, head(X)) \times P(head(X) \mid X, head(mother(X)))$$

Very simplified variant of Collins' parser

His models also include:

- distinction of arguments(adjuncts)
- distance measures
- punctuation
- methods for handling coordination
- traces and movement

Evaluation

PARSEVAL measures(Black et al. 1991):

$$\text{labeled recall} := \frac{\#\ \text{of correct constituents in candidate parse of S}}{\#\ \text{of correct constituents in treebank parse of S}}$$

$$\text{labeled precision} := \frac{\#\ \text{of correct constituents in candidate parse of S}}{\#\ \text{of total constituents in candidate parse of S}}$$

$$\text{cross-brackets} := \#\ \text{of crossed brackets}$$

(Number of constituents for which the treebank has a bracketing such as ((AB) C) and the candidate parse has bracketing (A (B C)))

QUESTIONS
&
DISCUSSION

Evaluation Results for the parser from Collins(2000):

Labeled Recall:	90.1 %
Labeled Precision:	90.4 %
Av. Crossed Brackets:	0.73
0 Crossed Brackets:	70.7 %
<=2 Crossed Brackets:	89.6 %