# Lecture 5: UDOP, Dependency Grammars

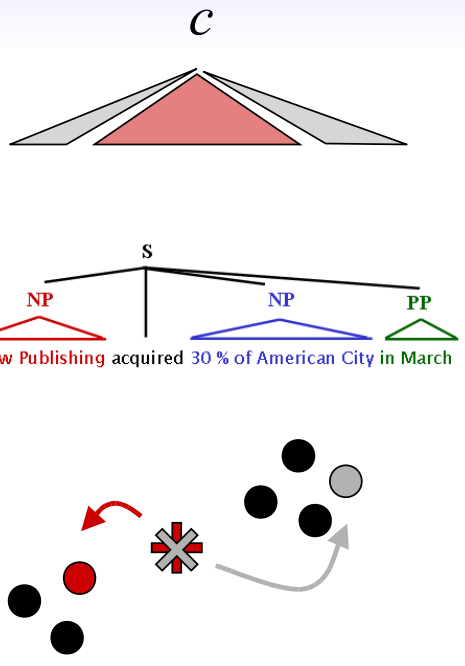
Jelle Zuidema
ILLC, Universiteit van Amsterdam

Unsupervised Language Learning, 2014

| | *Generative Model* | | | |
|---|---|---|---|---|
| *objective* | PCFG | PTSG | CCM | DMV |
| heuristic | Wolff (1984) | UDOP | | |
| ML | IO | | K&M'05 | K&M'04 |
| MAP | Stolcke'94 | | | |

# A Nested Distributional Model

- Klein and Manning (2002) propose a model that:
  - Ties spans to linear contexts (like distributional clustering)
  - Considers only proper tree structures
  - Has no symmetries to break (like a dependency model)



# Generative model

- $S$ is a sentence
- $B$ is a bracketing
- $P_{bin}(B)$ : uniform prob. over binary bracketings
- $\alpha_{ij}$ - parts-of-speech from $i$ to $j$
- $x_{ij}$ – context of $\alpha_{ij}$
- $P(S,B) = P_{bin}(B)\, P(S|B)$
- $P(S|B) = \prod P(\alpha_{ij}|B_{ij})P(x_{ij}|B_{ij})$

# Generative model

- $S$ is a sentence
- $B$ is a bracketing
- $P_{bin}(B)$ : uniform prob. over binary bracketings
- $\alpha_{ij}$ - parts-of-speech from $i$ to $j$
- $x_{ij}$ – context of $\alpha_{ij}$
- $P(S,B) = P_{bin}(B)\, P(S|B)$
- $P(S|B) = \prod P(\alpha_{ij}|B_{ij})P(x_{ij}|B_{ij})$

# Generative model

- $S$ is a sentence
- $B$ is a bracketing
- $P_{bin}(B)$ : uniform prob. over binary bracketings
- $\alpha_{ij}$ - parts-of-speech from $i$ to $j$
- $x_{ij}$ – context of $\alpha_{ij}$
- $P(S,B) = P_{bin}(B)\ P(S|B)$
- $P(S|B) = \prod_{i<j} P(\alpha_{ij}|B_{ij})P(x_{ij}|B_{ij})$

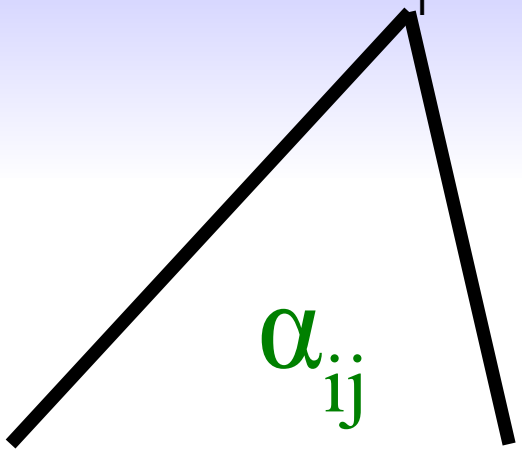
| O | Factory | Paysrolls | Fell | In | September | |
|---|---|---|---|---|---|---|
| | T | T | | | T | O |
| | | T | | | | Factory |
| | | | T | | T | Payrolls |
| | | | | T | T | Fell |
| | | | | | T | In |
| | | | | | | September |

(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

# Constituent...

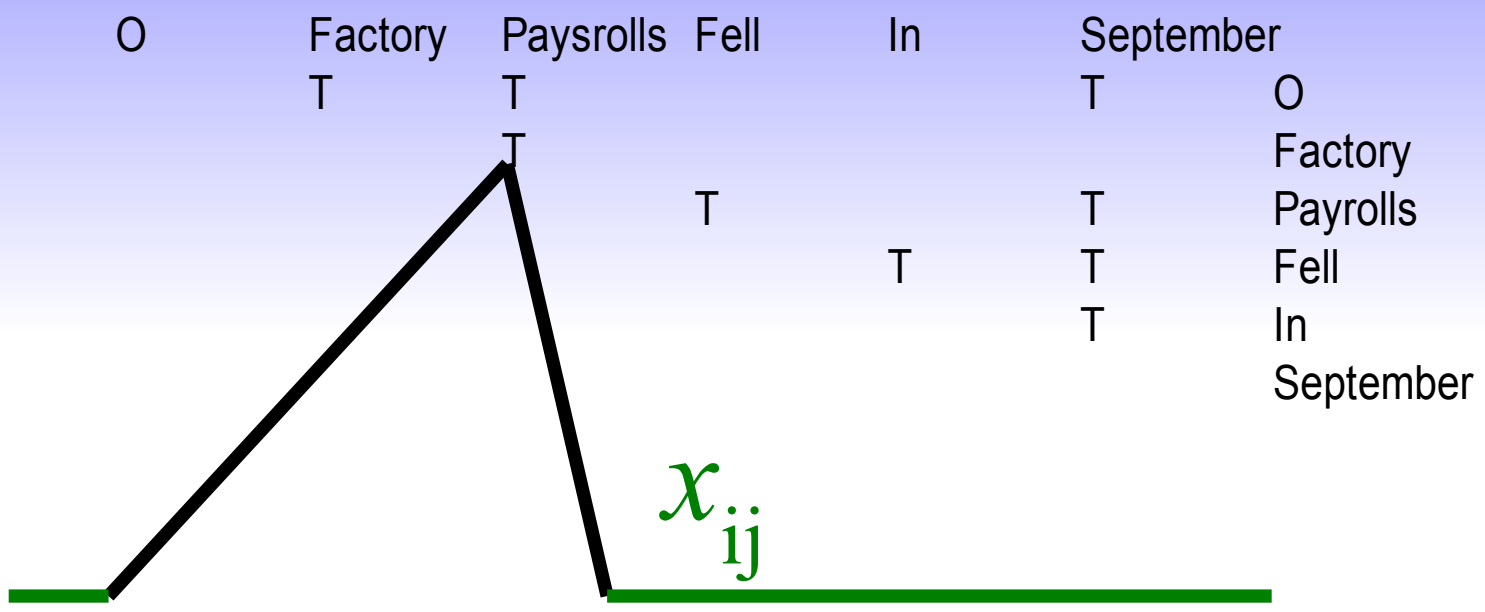
| | | | | | | |
|---|---|---|---|---|---|---|
| O | Factory | Paysrolls | Fell | In | September | |
| | T | T | | | T | O |
| | | T | | | | Factory |
| | | | T | | T | Payrolls |
| | | | | T | T | Fell |
| | | | | | T | In |
| | | | | | | September |




(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

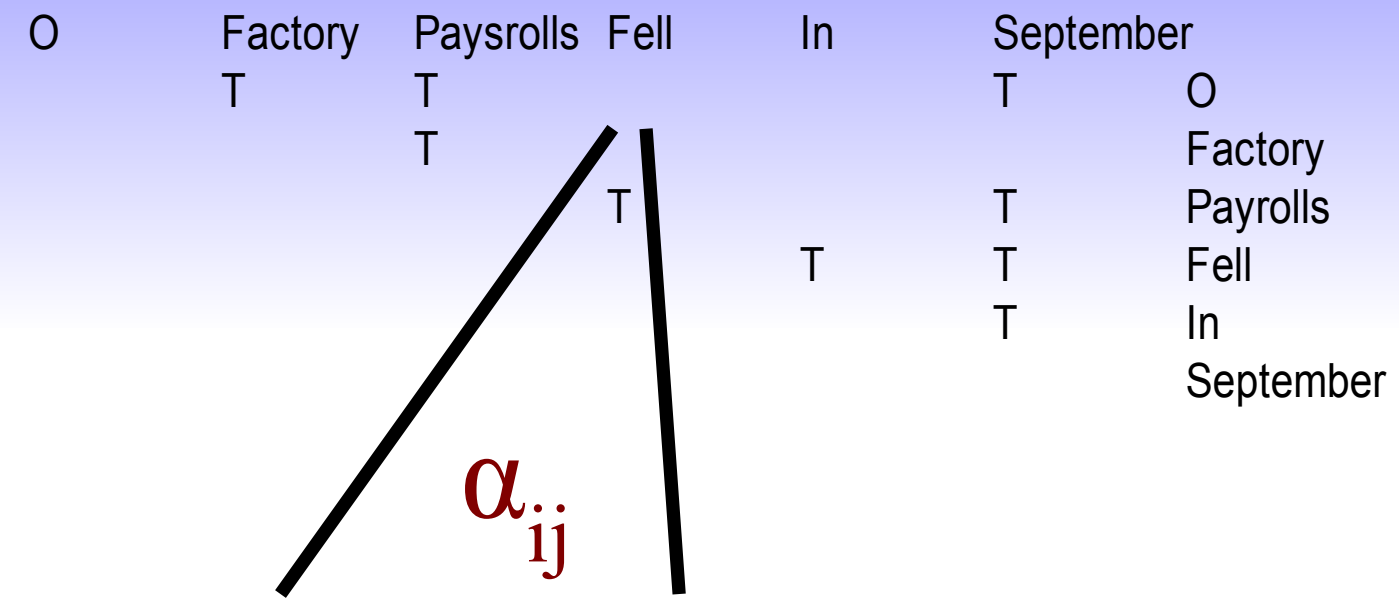
# ... Context




(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

# Distituent...

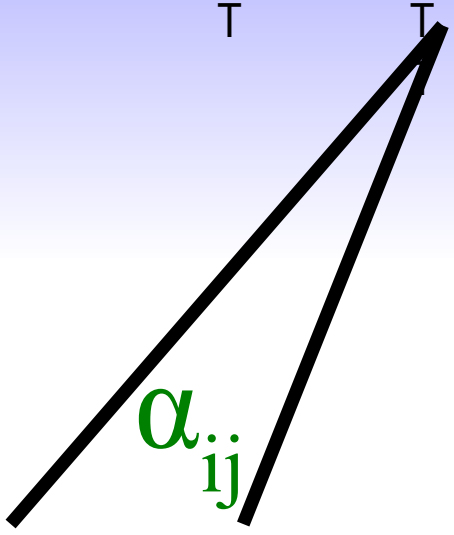



(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

# Constituent...

| | | | | | | |
|---|---|---|---|---|---|---|
| O | Factory | Paysrolls | Fell | In | September | |
| | T | T | | | T | O |
| | | | | | | Factory |
| | | | T | | T | Payrolls |
| | | | | T | T | Fell |
| | | | | | T | In |
| | | | | | | September |

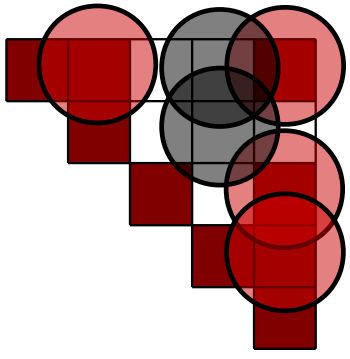


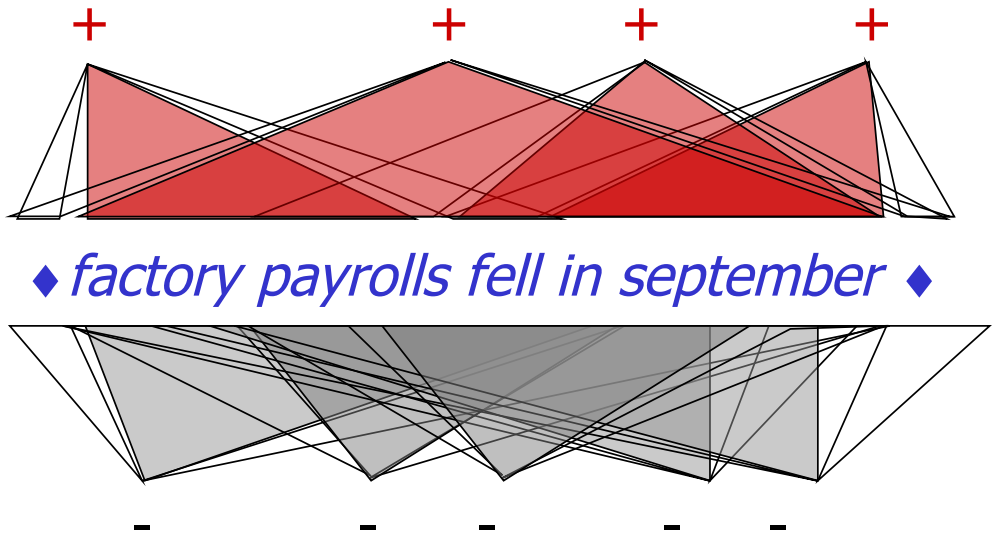

(((Factory) (Payrolls)) (((Fell) ((In) (September))))))

NN NN VBP IN NN

# Constituent-Context Model (CCM)

$$\mathrm{P}(S,B) = \mathrm{P}(B)\ \mathrm{P}(S|B)$$






$\mathrm{P}(B)$ $\mathrm{P}(S|B)$

# CCM

- Defective probability model;
- Good empirical performance;
- “heuristic” rather than “maximum likelihood”;
- Dead end?

# UDOP

# Shortcomings of CCM

- CCM neglects dependencies that are *non*-contiguous:
  - *British Airways carried **more** people **than** cargo* (WSJ)
  - *Most **companies** in Vietnam **are** small-sized* (Business in Asia)

=>We need an *all-subtrees* model, rather than *all-substrings* model: grammar induction with **PTSG** rather than **PCFG**

# Probabilistic Tree Substitution Grammars

- Another generative model
- String sets same as those of context-free grammars
- Larger set of tree languages
- Probabilistically richer than PCFGs (PCFG class is properly contained in PTSG class).

# The ‘all-subtrees approach’ to unsupervised induction

- **Use counts of *all* subtrees from *all* possible trees of sentences to predict the most probable parse trees for (new) sentences** (maximalist approach)
- All-subtrees models have already been widely used in *supervised* parsing:

  DOP, Tree-Kernels, PTSG: Bod (1993, 2003), Collins and Duffy (2002, 2004), Sima'an (2003), Goodman (2003), Kudo et al. (2005), Moschitti et al (2006) ...
- Alternative to Bod's maximalism: Parsimonious Data-oriented Parsing (Zuidema, 2007)
- Recently, application of DOP to **unsupervised** induction

# How can we learn PTSGs?

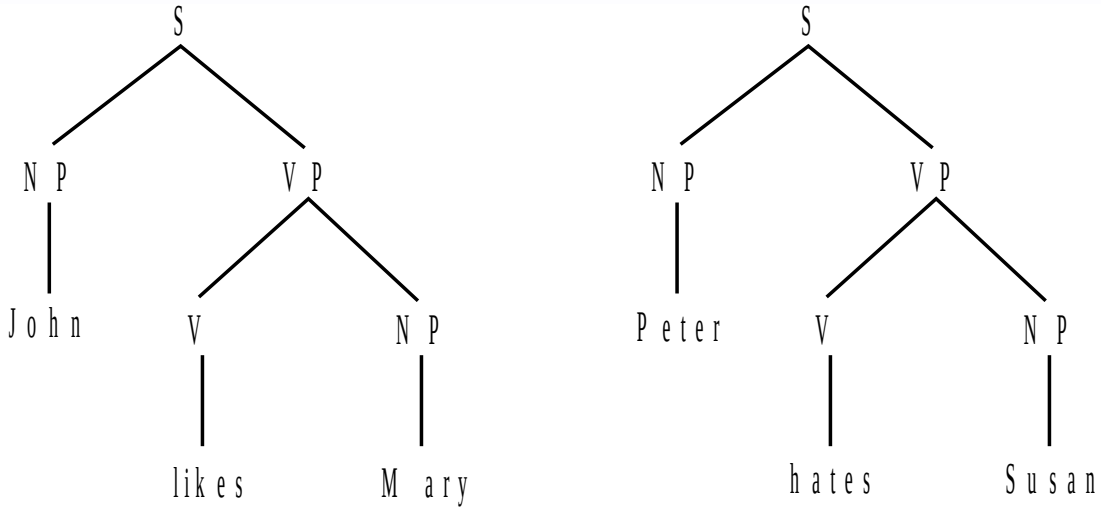
- PTSGs: probabilistic tree-substitution grammars:
  - productive units can span several levels of constituent structure
  - DOP: PTSG where subtrees are learned from a treebank

- We will first deal with **supervised** DOP , next with **unsupervised** DOP (“U-DOP”)

# Basic idea of (supervised) DOP: use all subtrees from a treebank as probabilistic tree grammar

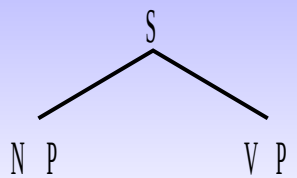
(Scha, 1990; Bod 1993,1998)

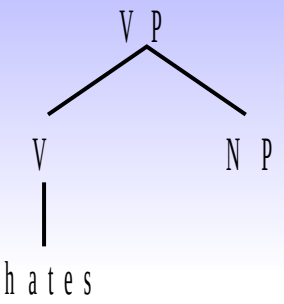
Given an extremely simple treebank consisting of two tree structures:

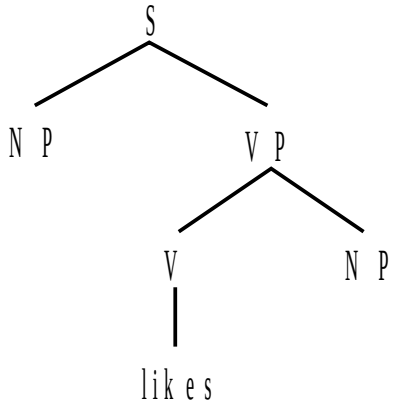


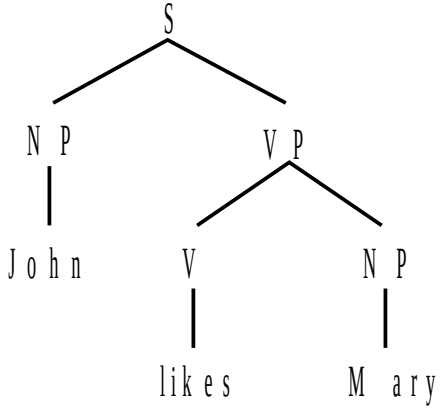
# Divide the trees into fragments (subtrees):





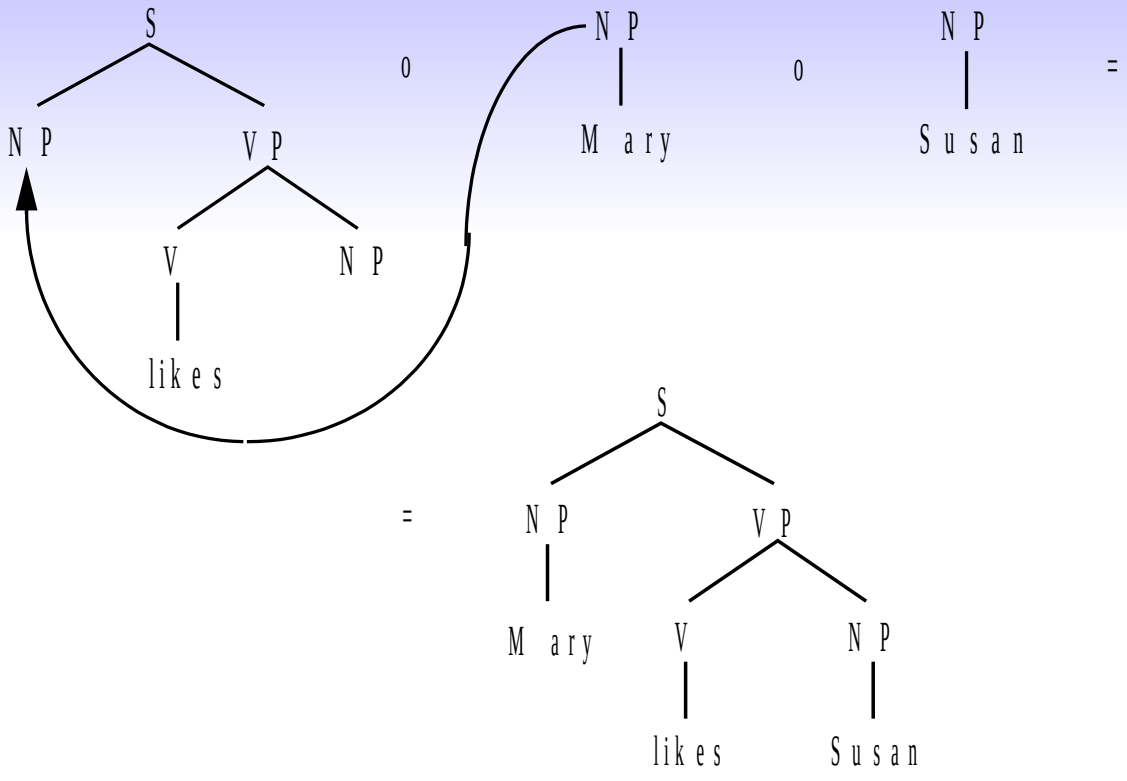



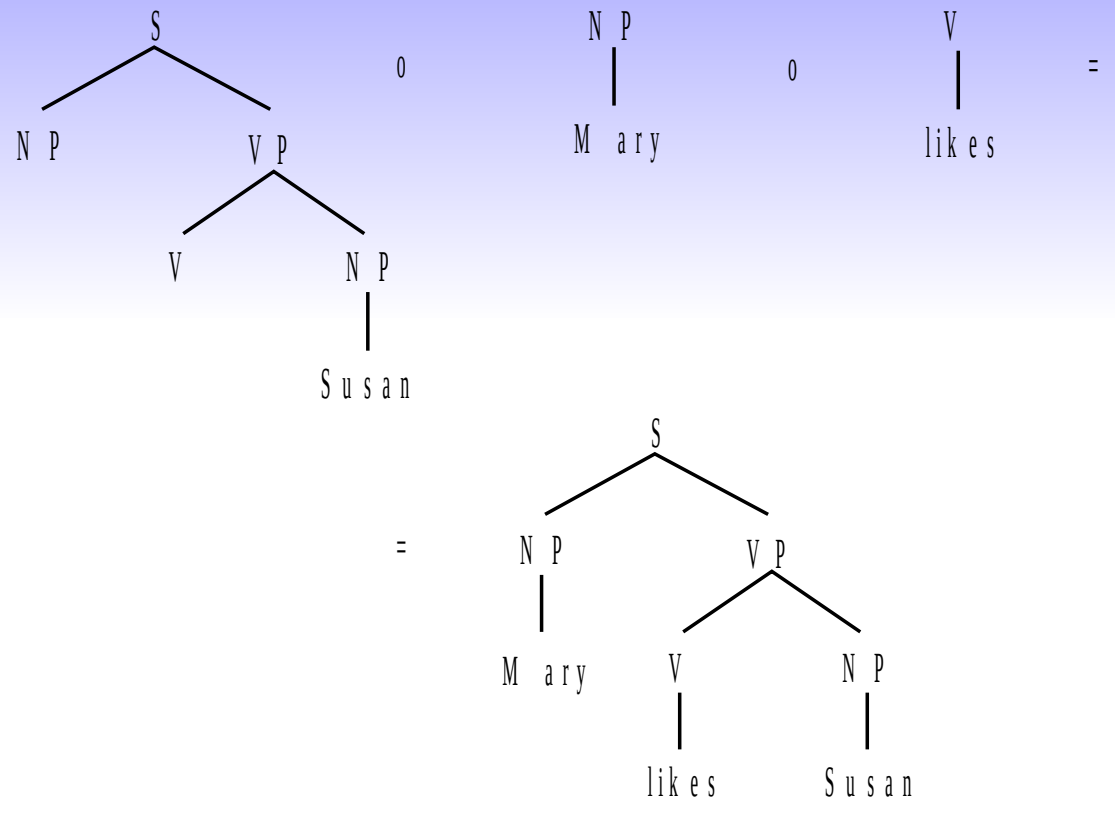

etc.

Combine fragments to derive trees for new sentences

In DOP, "o" is left-most substitution

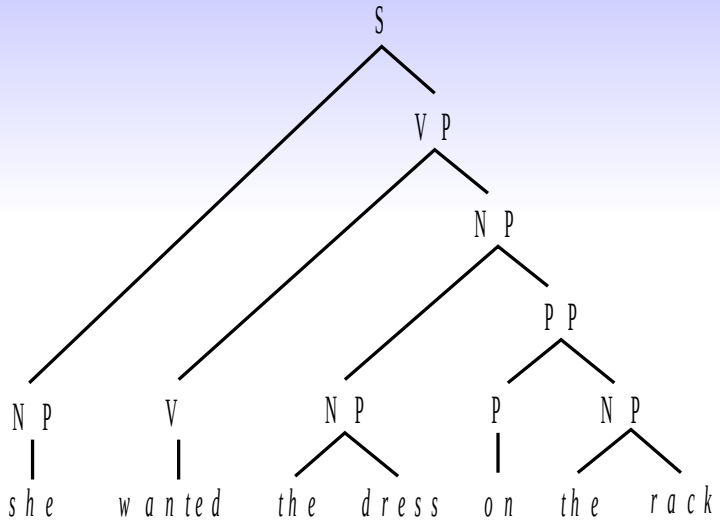


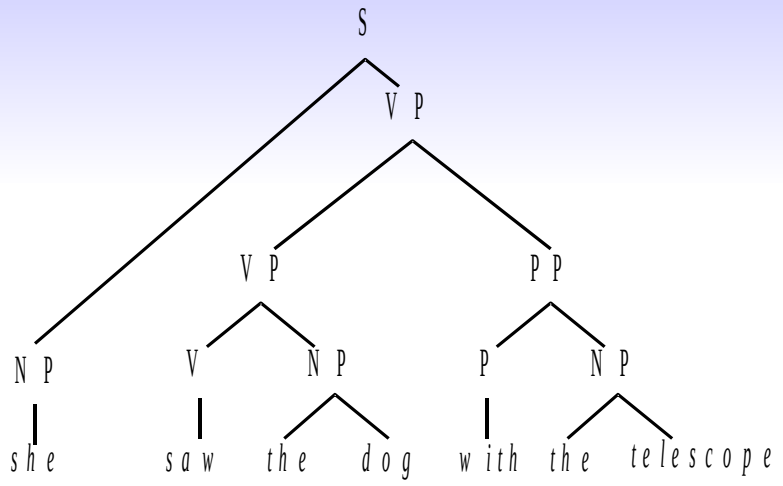
# Another derivation resulting in the same tree:



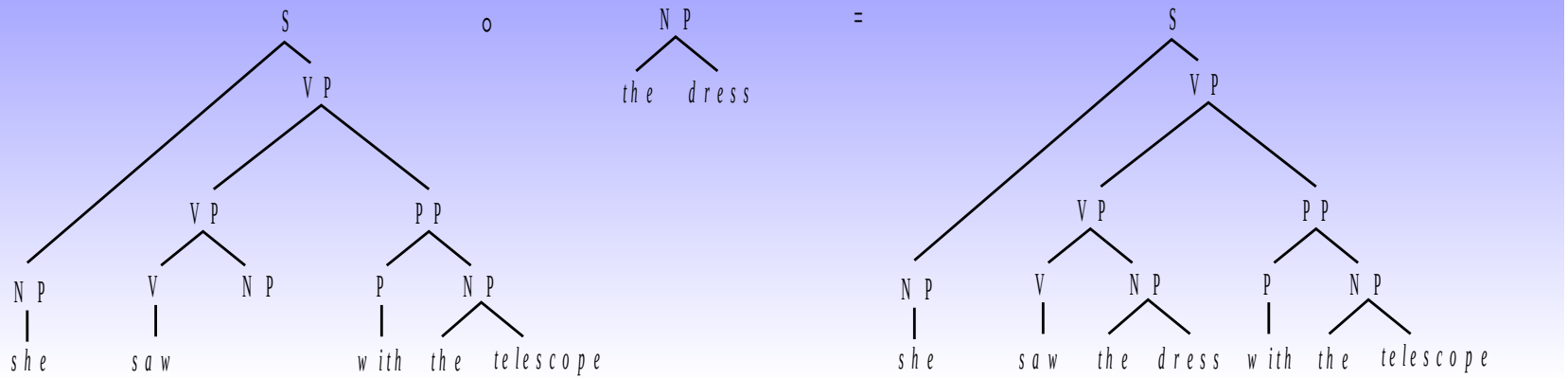
# An example involving structural ambiguity

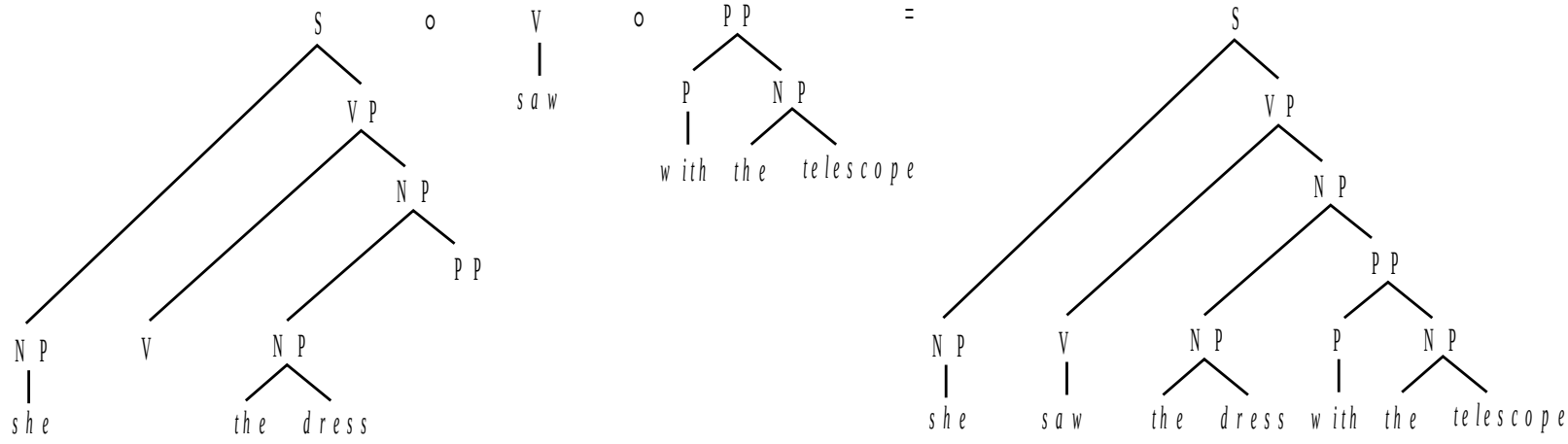
Given following corpus of just two sentences:



*She saw the dress with the telescope* can be derived in two ways by subtrees from the corpus:




=> The first derivation is preferred if we want to maximize 'analogy' with corpus.

# How does DOP determine the 'best' tree?

- DOP1: weights of subtrees are determined by their relative frequencies in a corpus;
- Shortest derivation (of fewest subtrees) maximizes overlaps with previous sentences (Bod 2000): *"structural analogy"*
- If shortest derivation is not unique (i.e. the typical case) compute most probable tree from among trees proposed by shortest derivations:
- **MPSD**: *Most probable shortest derivation*
  (Bod 2003, Zollmann and Sima'an 2005)
- **ML-DOP** uses *EM*, **PDOP** can be given Bayesian interpretation (*work in progress)*

# How does DOP determine the 'best' tree?

- DOP1: weights of subtrees are determined by their relative frequencies in a corpus;
- Shortest derivation (of fewest subtrees) maximizes overlaps with previous sentences (Bod 2000): *"structural analogy"*
- If shortest derivation is not unique (i.e. the typical case) compute most probable tree from among trees proposed by shortest derivations:
- **MPSD**: *Most probable shortest derivation*
  (Bod 2003, Zollmann and Sima'an 2005)
- **ML-DOP** uses *EM*, **PDOP** can be given Bayesian interpretation (*work in progress)*

# How does DOP determine the 'best' tree?

- DOP1: weights of subtrees are determined by their relative frequencies in a corpus;
- Shortest derivation (of fewest subtrees) maximizes overlaps with previous sentences (Bod 2000): *"structural analogy"*
- If shortest derivation is not unique (i.e. the typical case) compute most probable tree from among trees proposed by shortest derivations:
- **MPSD**: *Most probable shortest derivation* (Bod 2003, Zollmann and Sima'an 2005)
- **ML-DOP** uses *EM*, **PDOP** can be given Bayesian interpretation (*work in progress*)

# How does DOP determine the 'best' tree?

- DOP1: weights of subtrees are determined by their relative frequencies in a corpus;
- Shortest derivation (of fewest subtrees) maximizes overlaps with previous sentences (Bod 2000): *"structural analogy"*
- If shortest derivation is not unique (i.e. the typical case) compute most probable tree from among trees proposed by shortest derivations:
- **MPSD**: *Most probable shortest derivation*

  (Bod 2003, Zollmann and Sima'an 2005)
- **ML-DOP** uses *EM*, **PDOP** can be given Bayesian interpretation (*work in progress*)

# How does DOP determine the 'best' tree?

- DOP1: weights of subtrees are determined by their relative frequencies in a corpus;
- Shortest derivation (of fewest subtrees) maximizes overlaps with previous sentences (Bod 2000): *"structural analogy"*
- If shortest derivation is not unique (i.e. the typical case) compute most probable tree from among trees proposed by shortest derivations:
- **MPSD**: *Most probable shortest derivation*

  (Bod 2003, Zollmann and Sima'an 2005)
- **ML-DOP** uses *EM*, **PDOP** can be given Bayesian interpretation (*work in progress)*

# How does DOP determine the 'best' tree?

- Shortest derivation (of fewest subtrees) maximizes overlaps with previous sentences (Bod 2000):

  *"structural analogy"*

- If shortest derivation is not unique (i.e. the typical case) compute most probable tree from among trees proposed by shortest derivations:

- **MPSD**: *Most probable shortest derivation* (Bod 2003, Zollmann and Sima'an 2005)

  1. Maximizes structural analogy
  2. Maximizes probability of the structure given a sentence

- Probability of a parse tree is computed from probabilities of all subtrees in the corpus

# Probability of…

a *subtree* $t$ : $$P(t) = \frac{|t|}{\sum_{t' : root(t')=root(t)} |t'|}$$

a *derivation* $d = t_1 \circ ... \circ t_n$ : $$P(t_1 \circ ... \circ t_n) = \prod_i P(t_i)$$

a *parse tree* $T$ : $$P(T) = \sum_d \prod_i P(t_{id})$$

$t_{id}$ is the $i$-th subtree in derivation $d$ that produces $T$

# Recall: DOP models of this kind are Probabilistic Tree-Substitution Grammars (PTSGs)

- PTSGs subsume:

  probabilistic regular grammars

  probabilistic context-free grammars

  probabilistic lexicalized grammars

  …

- DOP is a PTSG where tree-units are of arbitrary size

- "DOP principle"**:**

  *If you don't know which subtrees are needed, take them all and let statistics (and analogy) decide*

# Computational Issues

- Although there are exponentially many subtrees, there exists an efficient PCFG-reduction for DOP (Goodman 2003)
- This PCFG reduction is ***linear*** in the size of the corpus (8 rules for each node) rather than ***exponential***
- Thus: while there are exponentially many subtrees, the PCFG reduction of U-DOP is linear in size!
- Best tree can be efficiently computed by Viterbi $n$-best (cf. Manning and Schuetze 1999):

  Compute 1,000 most probable derivations by the standard Viterbi algorithm and sum up derivations that lead to the same tree

# Sketch of PCFG reduction of DOP

- Basic idea: Assign every node in every tree a unique number:
- Some terminology:

  $\underline{A@k}$ : the node at address $k$ where $A$ is the nonterminal labeling that node.

  A new nonterminal is created for each node in the training data.

  This nonterminal is called $A_k$.

  $a_j$ : the number of subtrees headed by the node $\underline{A@j}$,

  $a$ : the number of subtrees headed by nodes with nonterminal $A$,
  that is $a = \sum_j a_j$.

# Sketch of PCFG reduction of DOP (2)

There is a 'PCFG' with the following property: for every subtree in the training corpus headed by $A$, the grammar will generate an isomorphic subderivation with probability $1/a$.

E.g., for a node $(A@j\ (B@k, C@l))$, the following eight rules and probabilities are generated:

| | | | |
|---|---|---|---|
| $A_j \rightarrow BC$ | $(1/a_j)$ | $A \rightarrow BC$ | $(1/a)$ |
| $A_j \rightarrow B_kC$ | $(b_k/a_j)$ | $A \rightarrow B_kC$ | $(b_k/a)$ |
| $A_j \rightarrow BC_l$ | $(c_l/a_j)$ | $A \rightarrow BC_l$ | $(c_l/a)$ |
| $A_j \rightarrow B_kC_l$ | $(b_kc_l/a_j)$ | $A \rightarrow B_kC_l$ | $(b_kc_l/a)$ |

This construction produces derivations isomorphic to DOP derivations with equal probability (Goodman 2003: 130-133)

Thus, we can use a "compact" set of PCFGs which is ***linear*** in the size of the training corpus (8 rules for each node) rather than ***exponential***

# Computing MPSD with PCFG reduction

- **Recall:** MPSD selects most probable parse from among shortest derivations

  ⇨ Note that the shortest derivation is equivalent to most probable derivation if subtrees are given equal probability

- Thus there there exists also a PCFG reduction for the shortest derivation:

$$
\begin{array}{llll}
X_j \rightarrow BC & 1 & X \rightarrow BC & 0.5 \\
X_j \rightarrow B_k C & 1 & X \rightarrow B_k C & 0.5 \\
X_j \rightarrow B C_l & 1 & X \rightarrow B C_l & 0.5 \\
X_j \rightarrow B_k C_l & 1 & X \rightarrow B_k C_l & 0.5
\end{array}
$$

- Next, most probable parse from among shortest derivations is computed by Viterbi $n$-best as mentioned above

# How can we extend DOP to structure induction?

**Generalize the DOP principle (i.e. *be agnostic!*):**

- If you don't know what kind of structures should be assigned to sentences:

  *allow for all possible trees and let linguistic experience decide*

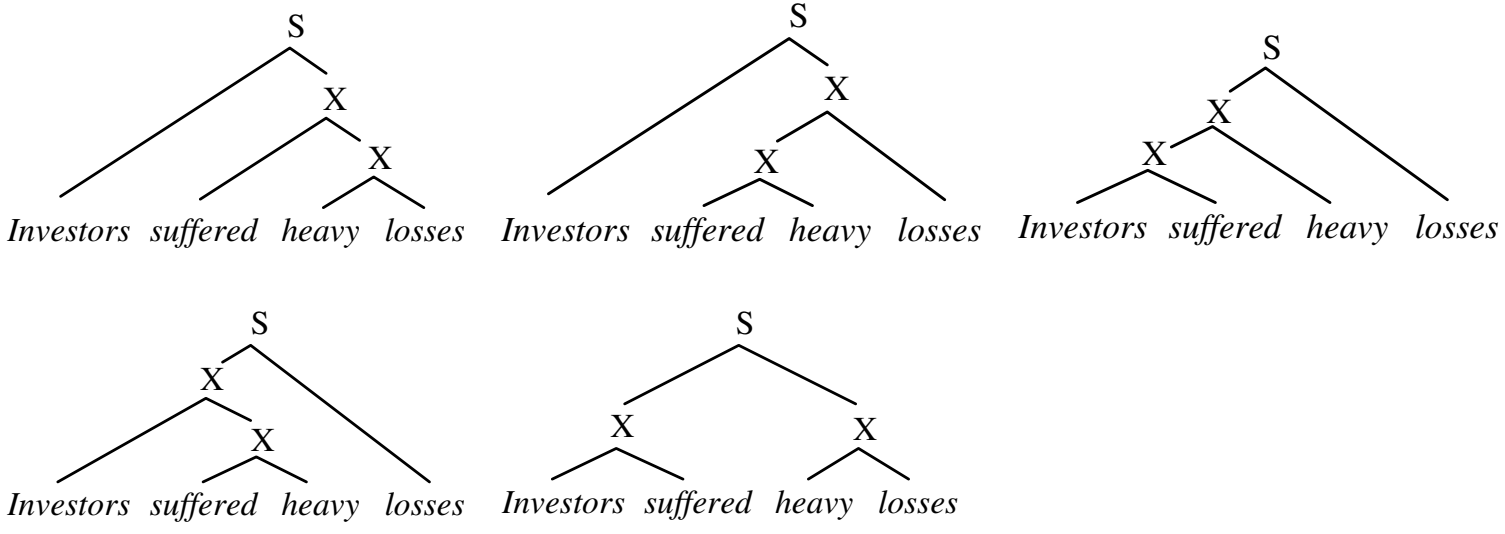
- Unsupervised DOP (*U-DOP*):

  1. Assign *all* possible unlabeled binary trees to a set of sentences
  2. Convert the binary trees into all subtrees
  3. Compute the best tree (MPSD) for each sentence
  4. These best trees form our DOP grammar

# How Does U-DOP Operate?

**1. Assign *all* possible binary trees to strings where each root node is labeled *S* and other nodes labeled *X*, and store them in a parse forest**

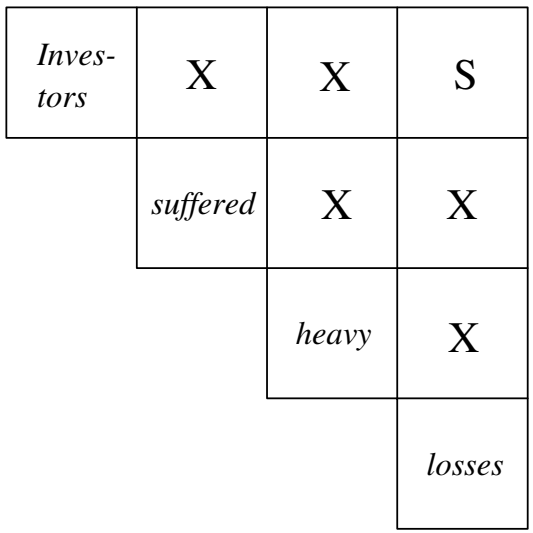
E.g., for WSJ sentence *Investors suffered heavy losses*:



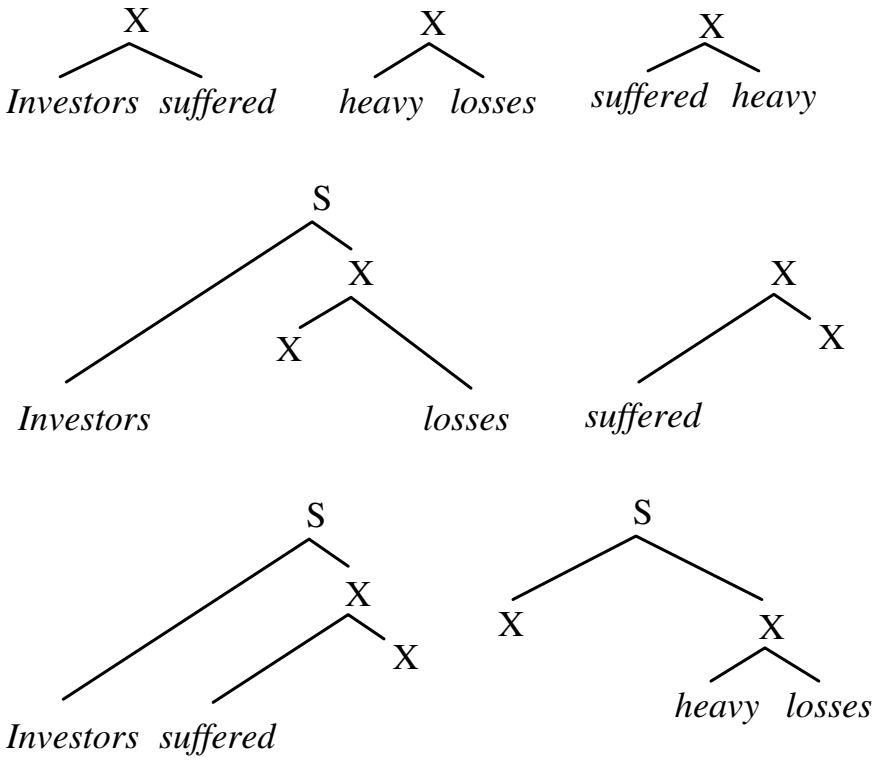
# All Binary Trees are Stored in a Chart

- Number of possible binary trees grows exponentially with string length

  but they can be efficiently stored by a shared *chart* in quadratic space



**2. Convert the set of all trees into all subtrees. For instance:**




=> Note that some subtrees contain discontiguous yields

**3. Compute most probable tree among shortest derivations for new string (as in DOP):**

Probability of…

a *subtree* $t$ : $$P(t) = \frac{|t|}{\sum_{t' : root(t')=root(t)} |t'|}$$

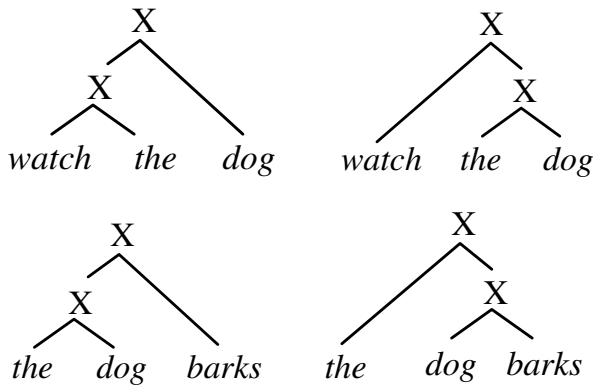
a *derivation* $d = t_1 \circ ... \circ t_n$ : $$P(t_1 \circ ... \circ t_n) = \prod_i P(t_i)$$

a *parse tree* $T$ : $$P(T) = \sum_d \prod_i P(t_{id})$$

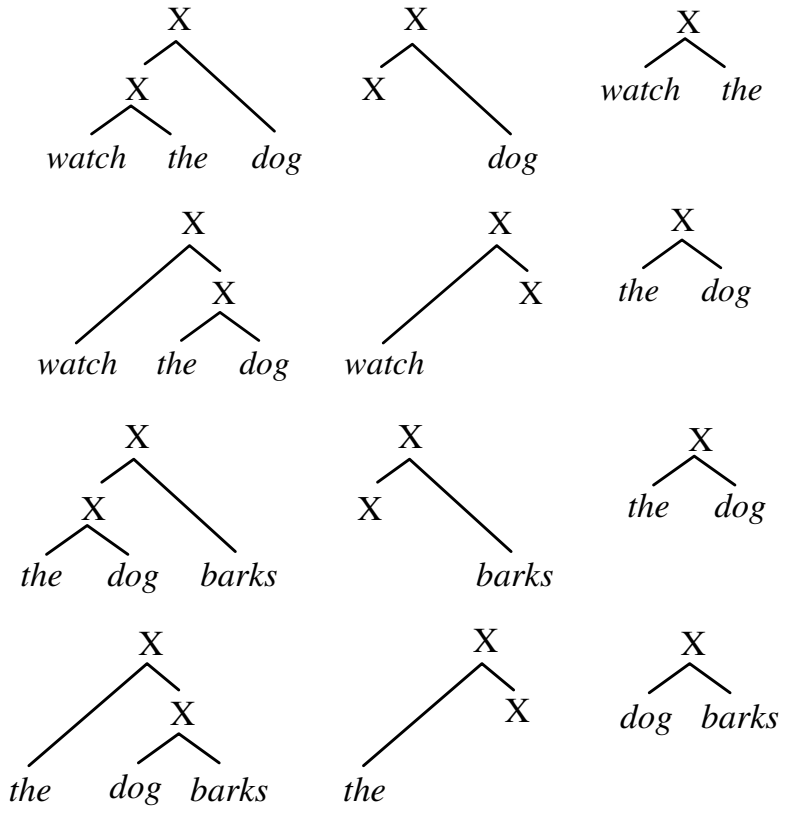
# A simple, conceptual illustration of U-DOP

Suppose that *Watch the dog* and *The dog barks* are heard by language learner:

**(1) Assign all unlabeled (binary) trees to a set of sentences :**



**(2) Divide the trees into all subtrees (12 subtrees total):**

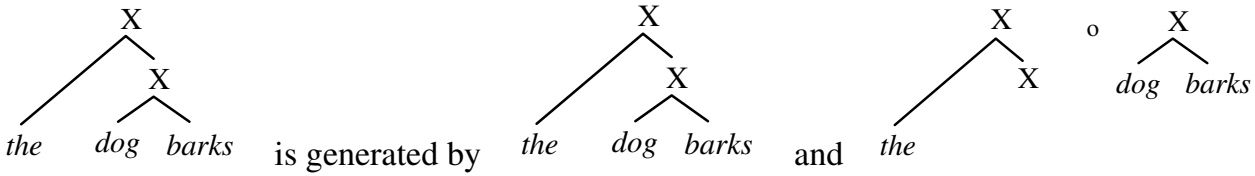



**Probability of each subtree is 1/12 except for [*the dog*] which has probability 2/12**

## (3) Compute the 'best' tree (MPSD) for each sentence

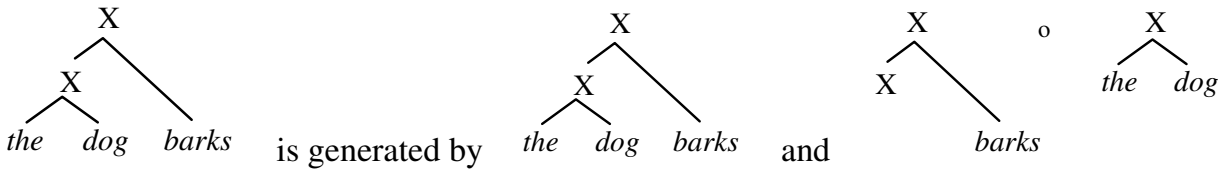
Probability of a tree structure is sum of probabilities of its derivations, e.g. for *the dog bark*:

**Tree-structure I:**




$P = 1/12 + (1/12 \times 1/12) = 13/144$

**Tree-structure II:**




$P = 1/12 + (1/12 \times 2/12) = 14/144$

Shortest derivation is not unique, thus tree structure II wins due to more frequent [*the dog*]

# Computational Issues

- Number of possible binary trees grows exponentially with string length

  but they can be efficiently stored by packed forest in quadratic space

- U-DOP can be converted into a compact PCFG reduction in roughly the same way as proposed for supervised DOP:

  Recall: there is a "compact" PCFG for DOP which is ***linear*** in the size of the training corpus (8 rules for each node) rather than ***exponential***

- How can we apply this PCFG reduction technique to parse forests of all binary trees of sentences?

# Computational Issues (2)

- U-DOP's trees are stored by AND-OR graph (parse forest) in O($n^3$):

  AND nodes are the usual parse tree nodes
  OR nodes are distinct partial parses occurring in the same context

- Instead of assigning a unique address to each node in each tree (as in DOP), *we assign a unique address to each node in each parse forest*

  ⇨ Same node may be part of more than one tree

- This means that there are cubically many nodes to which the PCFG reduction needs to be applied

# How can we evaluate U-DOP?

1. **Quantitative evaluation**:

   Induce constituent structures for Wall St Journal and CHILDES data and next compare results with hand-annotations

2. **Simulations**:

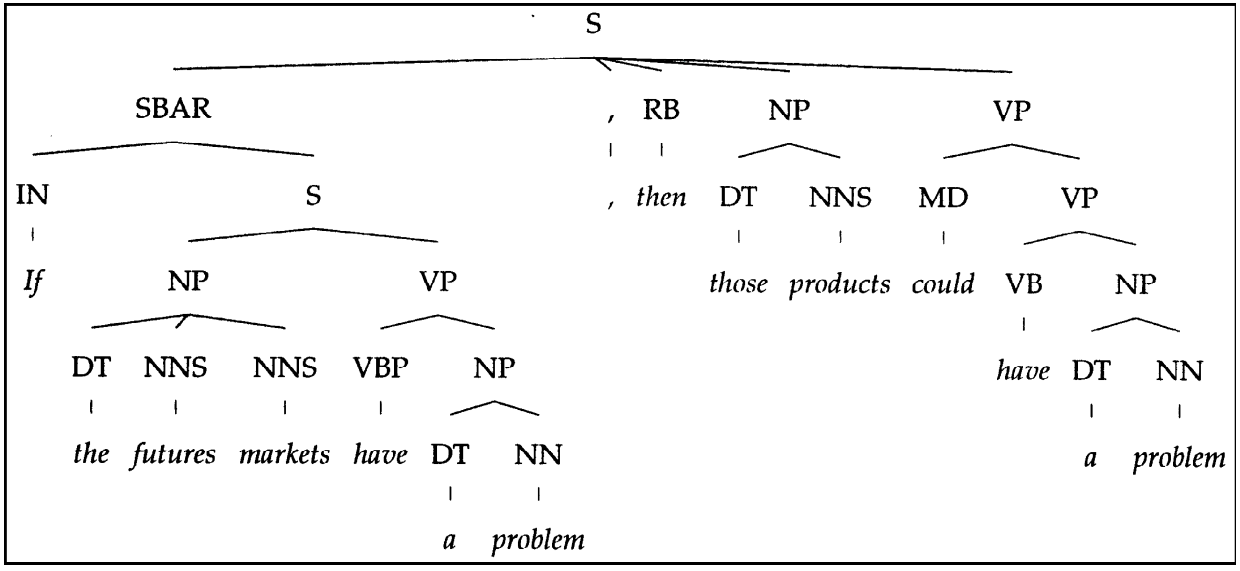
   Simulate learning of specific linguistic phenomena such as agreement, auxiliary inversion or particle verbs

3. **Qualitative evaluation**:

   Analyze the induced structures and simulated sentences

**Wall St Journal data is widely used in structure induction** (Klein and Manning 2004)**:**

"*If the future markets have a problem, then those products could have a problem*":




- All annotations are binarized to make comparison possible. We discard labels
- Accuracy is computed by f-score (F) of precision (P) and recall (R): F = 2*P*R/(P+R)

# Experiments with U-DOP on Penn Treebank data

unlabeled F-score on **word strings** up to 100 words (!) from sects 01-21 using 10-fold testing

| **Max. Subtree Depth** | **F-score** (Wall St Journal) |
|---|---|
| 1 (simple PCFG) | 28.7 |
| 2 | 40.5 |
| 3 | 58.9 |
| 4 | 66.0 |
| 6 | 68.7 |
| 8 | 68.8 |
| 10 | 70.2 |
| 16 | **70.3** |

- F-score increases with larger subtrees
- F-score is very low at depth 1 (simple "PCFG", cf. Lari and Young 1990)

# U-DOP compared to other models on WSJ-10

(using 7422 sentences up to **10** words, as in Klein and Manning 2004)

| **Model** | **F-score on WSJ-10** |
|---|---|
| CCM | 71.9 |
| DMV | 52.1 |
| DMV+CCM | 77.6 |
| U-DOP | **82.7** |
| U-DOP without discontiguous subtrees | **72.1** |

CCM: Klein and Manning (2002) based on all linear (contiguous) contexts *without* holes

DMV: Klein and Manning (2004) using dependency structures

U-DOP: equivalent to CCM *plus* discontiguous contexts *with* holes: *11% improvement in F-score*

# U-DOP for other languages

| **Model** | **English** (WSJ10) | | | **German** (NEGRA10) | | | **Chinese** (CTB10) | | |
|---|---|---|---|---|---|---|---|---|---|
| | UP | UR | **F1** | UP | UR | **F1** | UP | UR | **F1** |
| CCM | 64.2 | 81.6 | **71.9** | 48.1 | 85.5 | **61.6** | 34.6 | 64.3 | **45.0** |
| DMV | 46.6 | 59.2 | **52.1** | 38.4 | 69.5 | **49.5** | 35.9 | 66.7 | **46.7** |
| DMV+CCM | 69.3 | 88.0 | **77.6** | 49.6 | 89.7 | **63.9** | 33.3 | 62.0 | **43.3** |
| U-DOP | 75.9 | 90.9 | **82.7** | 52.4 | 91.0 | **66.5** | 37.6 | 65.7 | **47.8** |

WSJ10: 7422 sentences $\leq$ 10 words in American English
NEGRA10: 2298 sentences $\leq$ 10 words in German
CTB10: 2205 sentences $\leq$ 10 words in Mandarin Chinese

=> F-scores for German and Chinese are lower than for English.

# Shortcomings of U-DOP

- Viewed from the statistical inference perspective, the model relies much on heuristics: initialization, training & stopping
- Results with UML-DOP (Bod,06) suggests it is approximately Maximum Likelihood...
- ... but not over the entire PTSG space, as there are exponentially many subtrees, and exponentially many trees for a sentence!
- Implementation must somehow restrict space; efficiency remains the achilles heel.

# Shortcomings of U-DOP

- Viewed from the statistical inference perspective, the model relies much on heuristics: initialization, training & stopping
- Results with UML-DOP (Bod,06) suggests it is approximately Maximum Likelihood...
- ... but not over the entire PTSG space, as there are exponentially many subtrees, and exponentially many trees for a sentence!
- Implementation must somehow restrict space; efficiency remains the achilles heel.

# Shortcomings of U-DOP

- Viewed from the statistical inference perspective, the model relies much on heuristics: initialization, training & stopping
- Results with UML-DOP (Bod,06) suggests it is approximately Maximum Likelihood...
- ... but not over the entire PTSG space, as there are exponentially many subtrees, and exponentially many trees for a sentence!
- Implementation must somehow restrict space; efficiency remains the achilles heel.

# Shortcomings of U-DOP

- Viewed from the statistical inference perspective, the model relies much on heuristics: initialization, training & stopping
- Results with UML-DOP (Bod,06) suggests it is approximately Maximum Likelihood...
- ... but not over the entire PTSG space, as there are exponentially many subtrees, and exponentially many trees for a sentence!
- Implementation must somehow restrict space; efficiency remains the achilles heel.

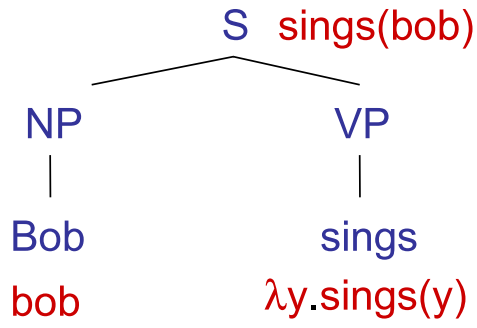
# Constituent labels

- Neither CCM nor U-DOP assign labels to constituents;
- Constituent labels are meant to define *substitutability*;
- Many difficulties with gold standard labels, but ignoring them is not the solution;
- BMM gives the currently best results on unsupervised labeling when brackets given.

# Dependency Grammars

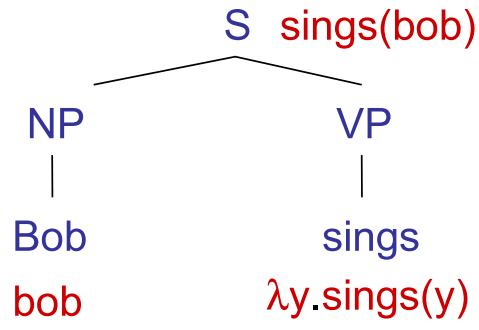
# Truth-Conditional Semantics

- Linguistic expressions:
  - "Bob sings"
- Logical translations:
  - sings(bob)
  - Could be p_1218(e_397)
- Denotation:
  - [[bob]] = some specific person (in some context)
  - [[sings(bob)]] = ???
- Types on translations:
  - bob : e (for entity)
  - sings(bob) : t (for truth-value)



# Truth-Conditional Semantics

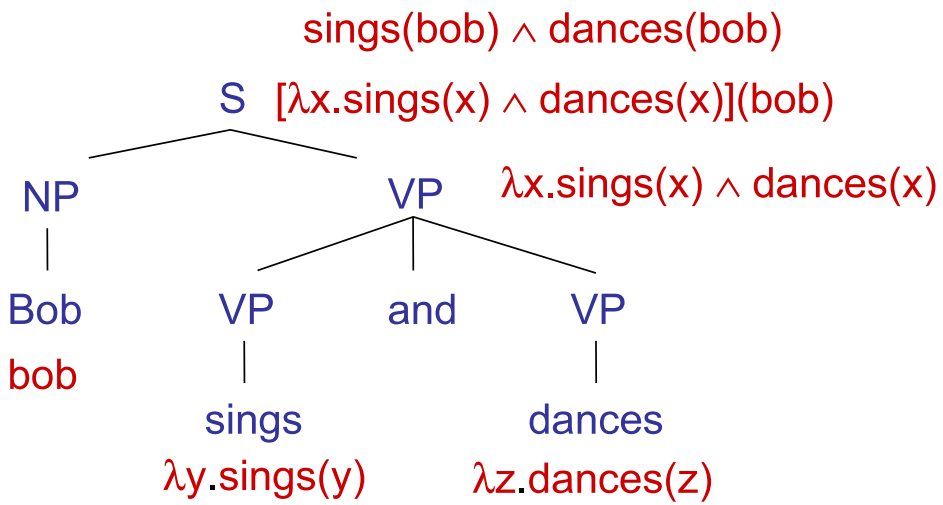
- Proper names:
  - Refer directly to some entity in the world
  - Bob : bob  $[[bob]]^w \rightarrow ???$
- Sentences:
  - Are either true or false (given how the world actually is)
  - Bob sings : sings(bob)



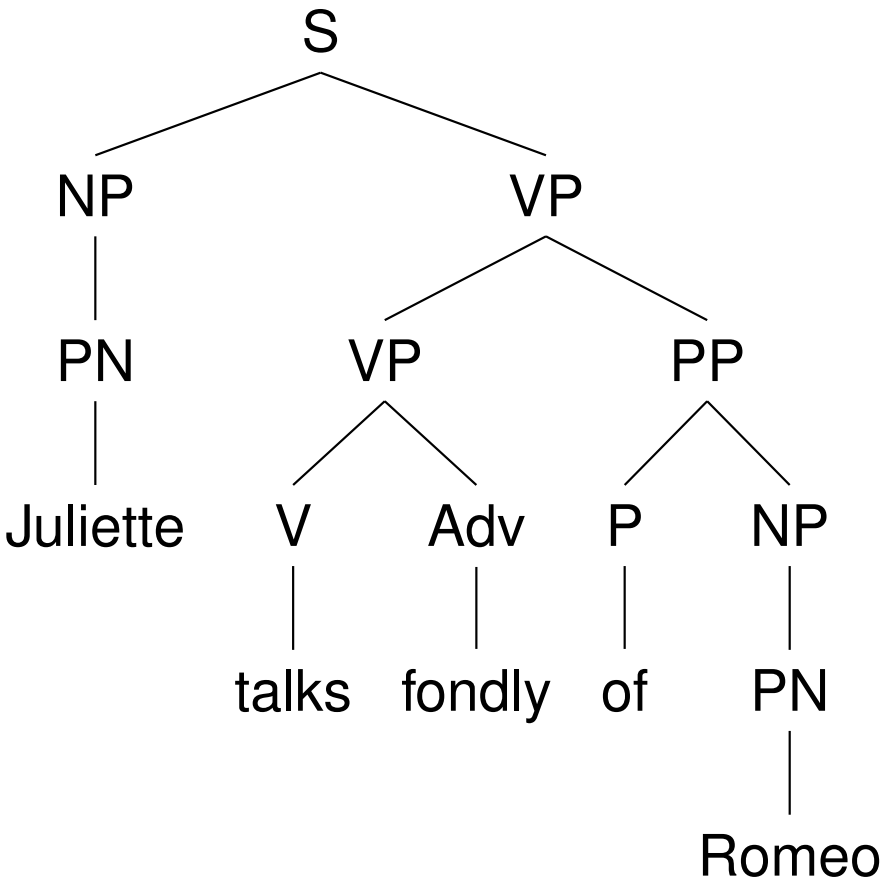

- So what about verbs (and verb phrases)?
  - sings must combine with bob to produce sings(bob)
  - The λ-calculus is a notation for functions whose arguments are not yet filled.
  - sings : λx.sings(x)
  - This is *predicate* – a function which takes an entity (type e) and produces a truth value (type t). We can write its type as e→t.
  - Adjectives?

# Compositional Semantics

- So now we have meanings for the words
- How do we know how to combine words?
- Associate a combination rule with each grammar rule:
  - S : $\beta(\alpha) \rightarrow$ NP : $\alpha$ VP : $\beta$ (function application)
  - VP : $\lambda x\ .\ \alpha(x) \wedge \beta(x) \rightarrow$ VP : $\alpha$ and : $\varnothing$ VP : $\beta$ (intersection)
- Example:



# Do we need phrase-structure?




talks(arg1,arg2,mod)

- Juliette
- Romeo
- fondly

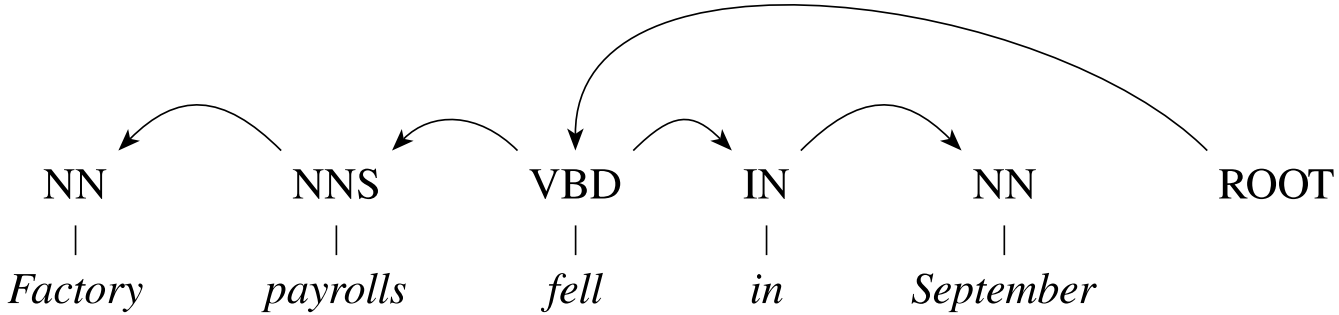
= talks(J,R,fondly)

Functor-argument structure

# Dependency Model

More recent models of unsupervised parsing use dependencies:

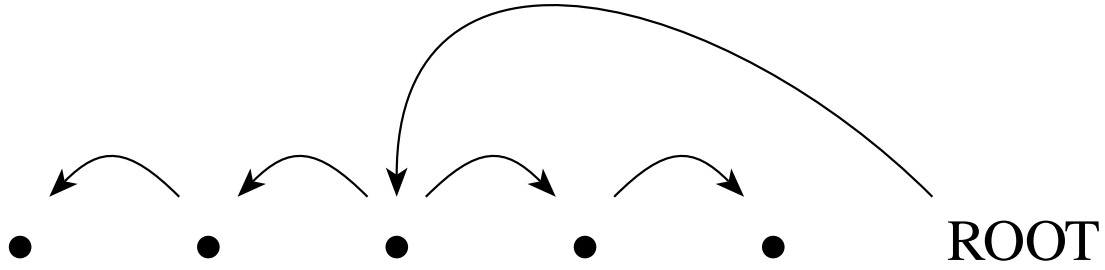
- dependency structures captures the head-argument and head-modifier relationships in a sentence;
- dependencies hold directly between lexical items;
- for language which have a flexible word order (German) and few function words (Chinese), dependency structure is easier to acquire than phrase structure.
- in a dependency graphs, arrows go from head to argument.



# Definitions

Definitions:

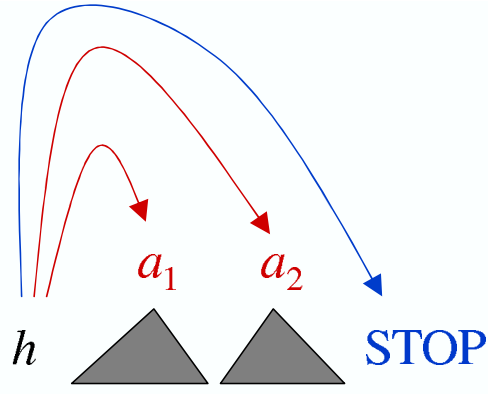
- a dependency $d$ is a pair $\langle h, a \rangle$, where $h$ is the head and $a$ the argument, both are words in sentence $s$;
- a dependency structure $D$ is set of dependencies which form a planar, acyclic graph rooted in ROOT;
- the skeleton $G$ of a dependency structure specifies the arrows, but not the words; $G$ and $s$ together fully determine the dependency structure.



# Dependency Model with Valence

Klein and Manning (2004) propose a model that generates dependencies outwards from the head:

- generate a set of arguments on one side of the head, then a STOP argument to terminate;
- the do the same thing on the other side;
- terminate with probability $P_{\text{STOP}}$, if not STOP then choose another argument with probability $P_{\text{CHOOSE}}$.



# Dependency Model with Valence

For a dependency structure $D$, let each word $h$ have left dependents $deps_D(h, l)$ right dependents $deps_D(h, r)$. Then the probability of the fragment $D(h)$ is:

$$P(D(h)) = \prod_{dir \in \{r,l\}} \prod_{a \in deps_D(h,dir)} P_{\text{STOP}}(\neg\text{STOP}|h, dir, adj)$$
$$P_{\text{CHOOSE}}(a|h, dir)P(D(a))P_{\text{STOP}}(\text{STOP}|h, dir, adj)$$

The binary variable *adj* indicates whether or not a dependent has already been generated in direction *dir*.

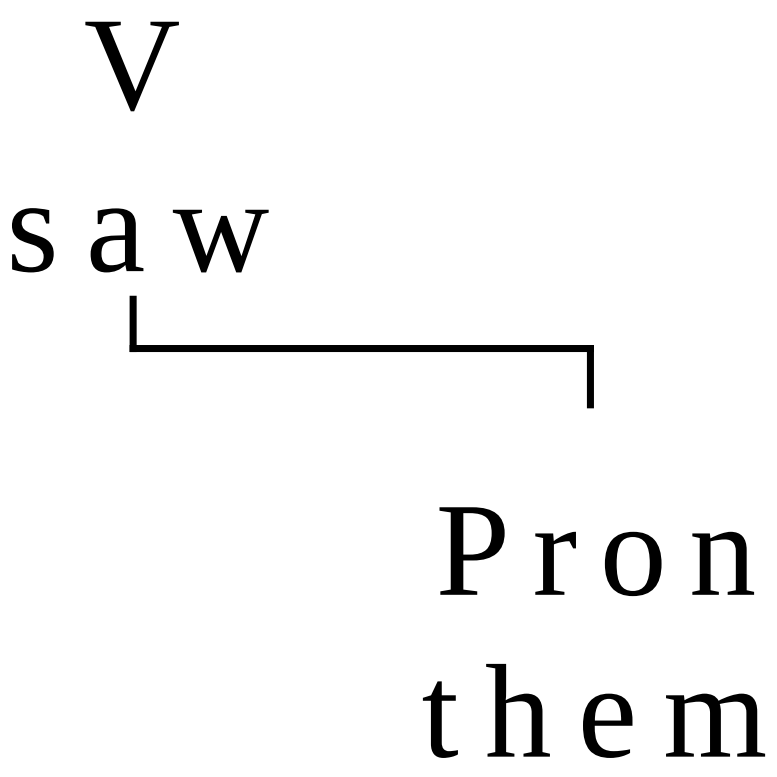
Use the EM algorithm to estimate these probabilities.

Sentence # 6

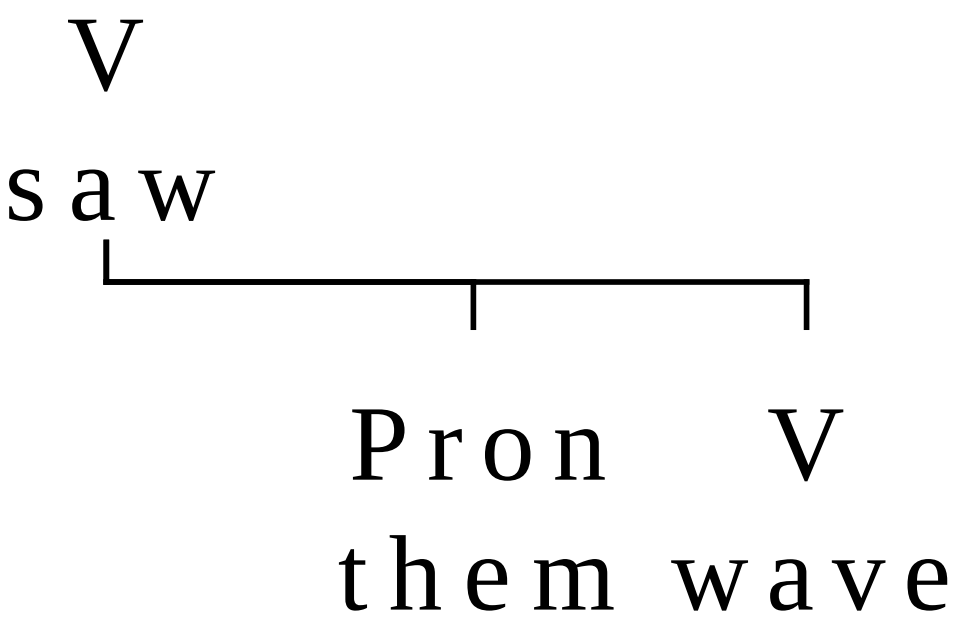
V

s a w

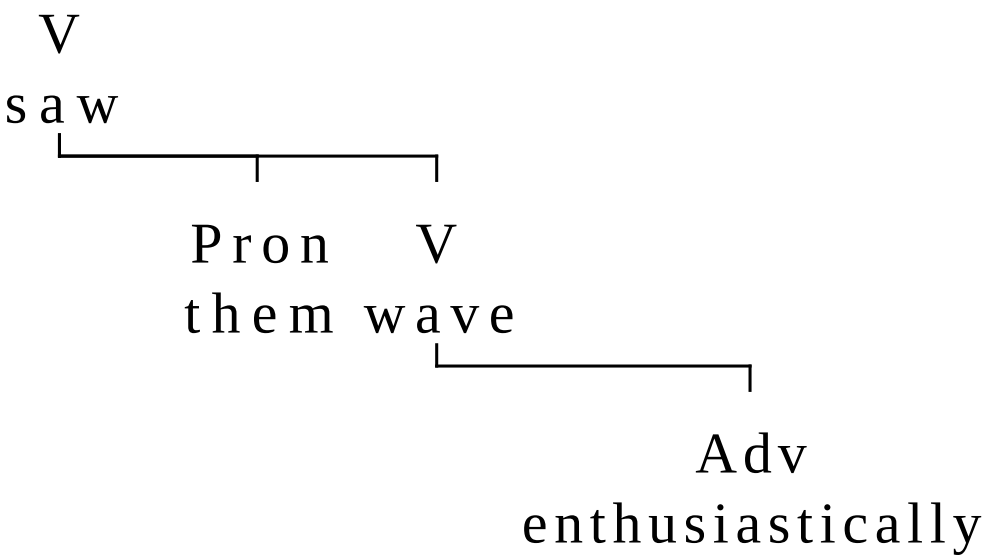
Sentence # 7



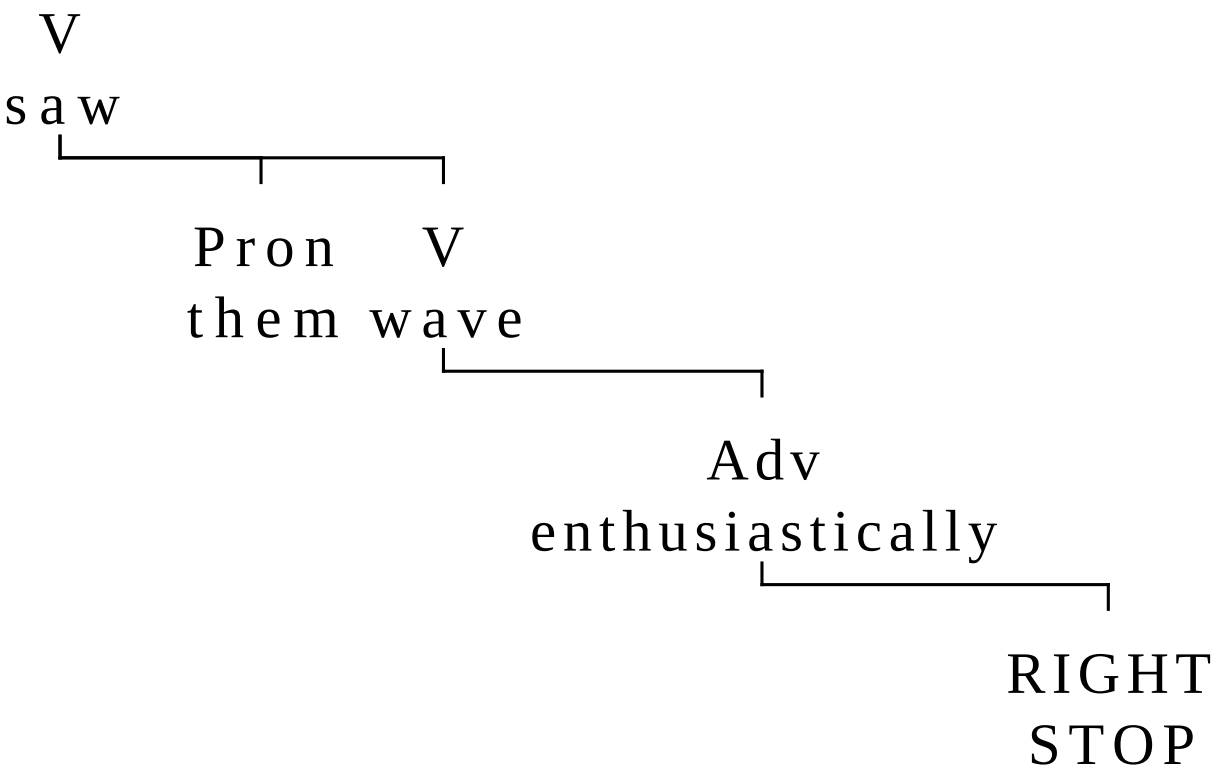
Sentence # 8



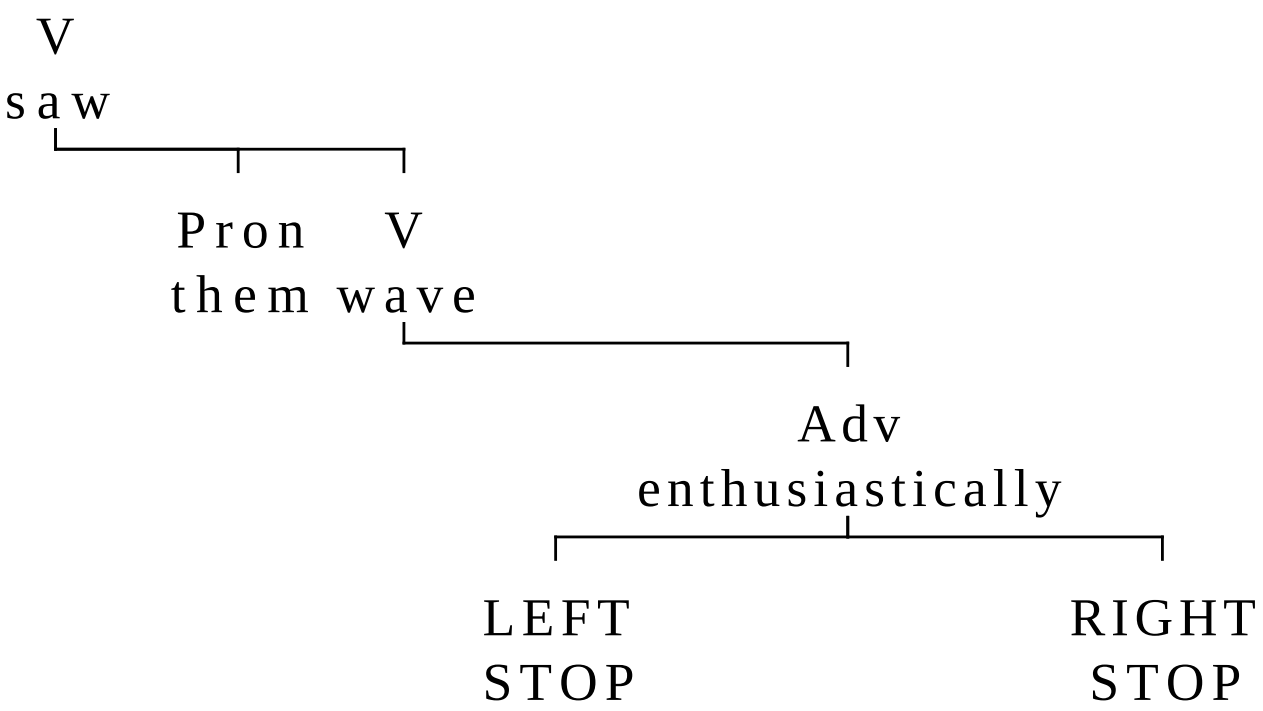
Sentence # 9

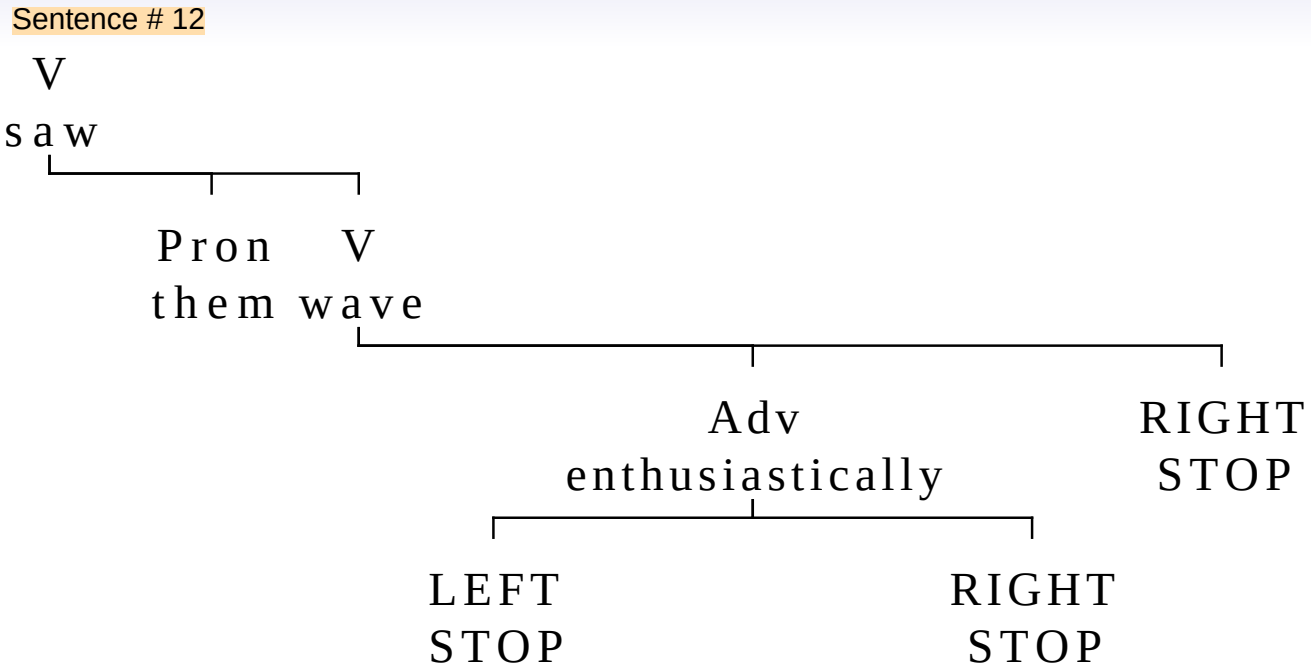


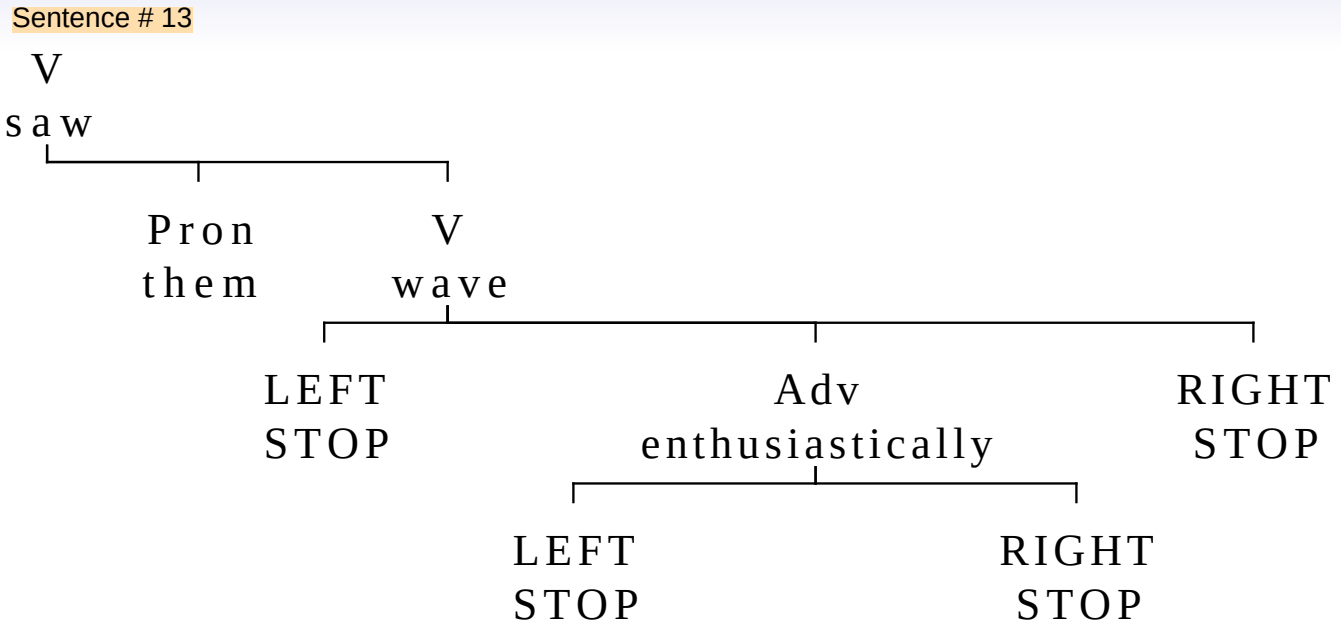
Sentence # 10



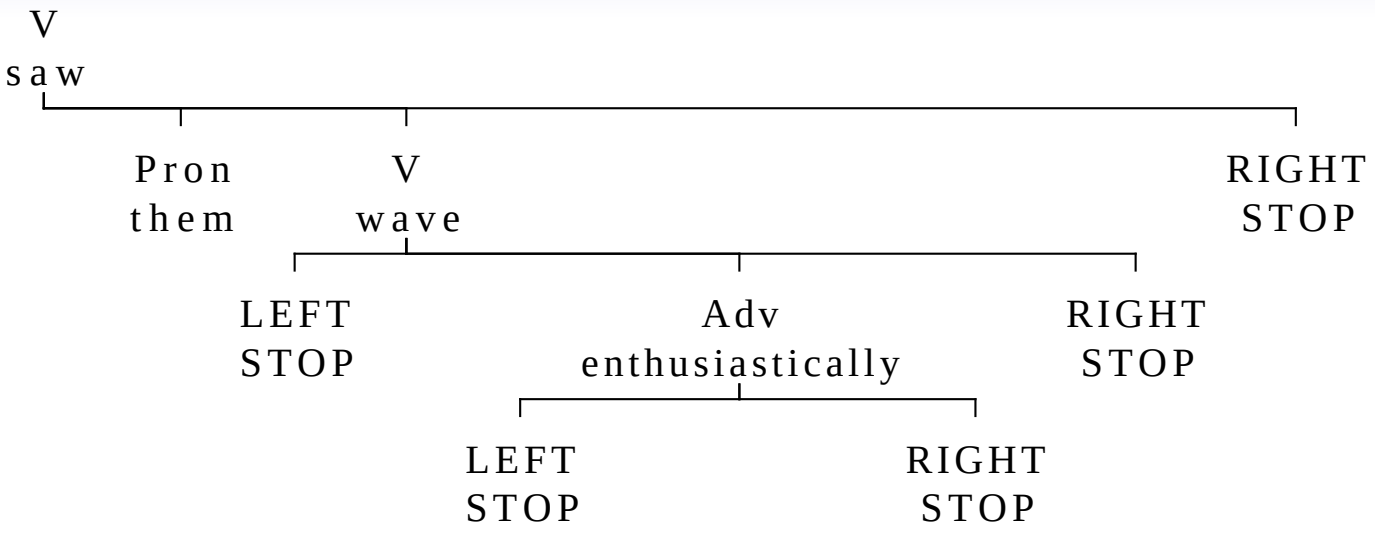
Sentence # 11


V
saw
Pron
them
V
wave
Adv
enthusiastically
LEFT
STOP
RIGHT
STOP

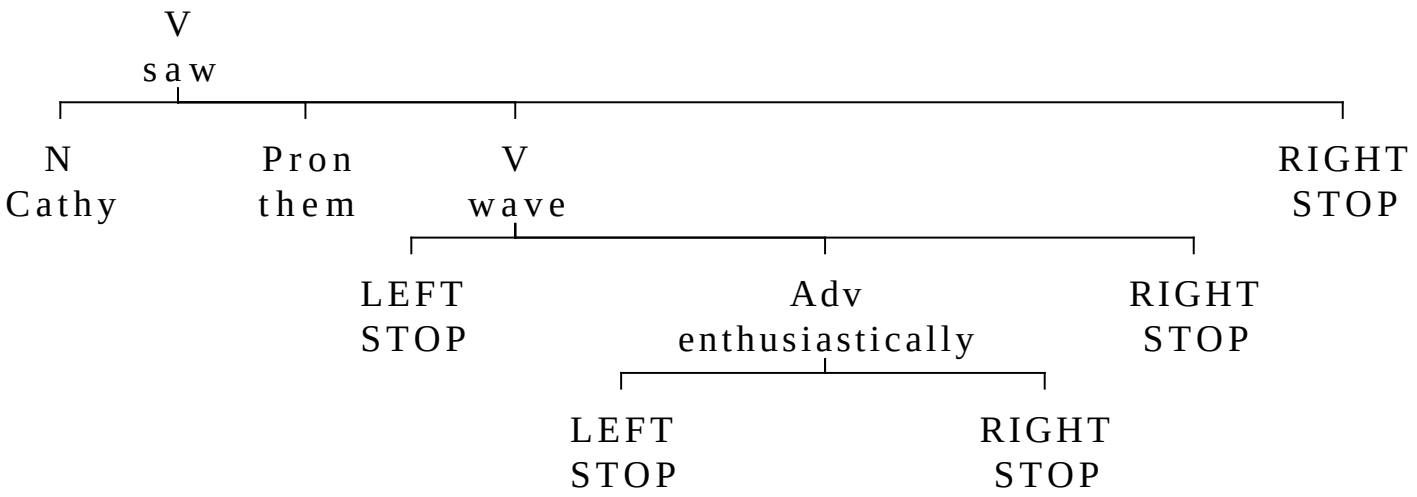

Sentence # 12
V
saw
Pron
them
V
wave
Adv
enthusiastically
RIGHT
STOP
LEFT
STOP
RIGHT
STOP

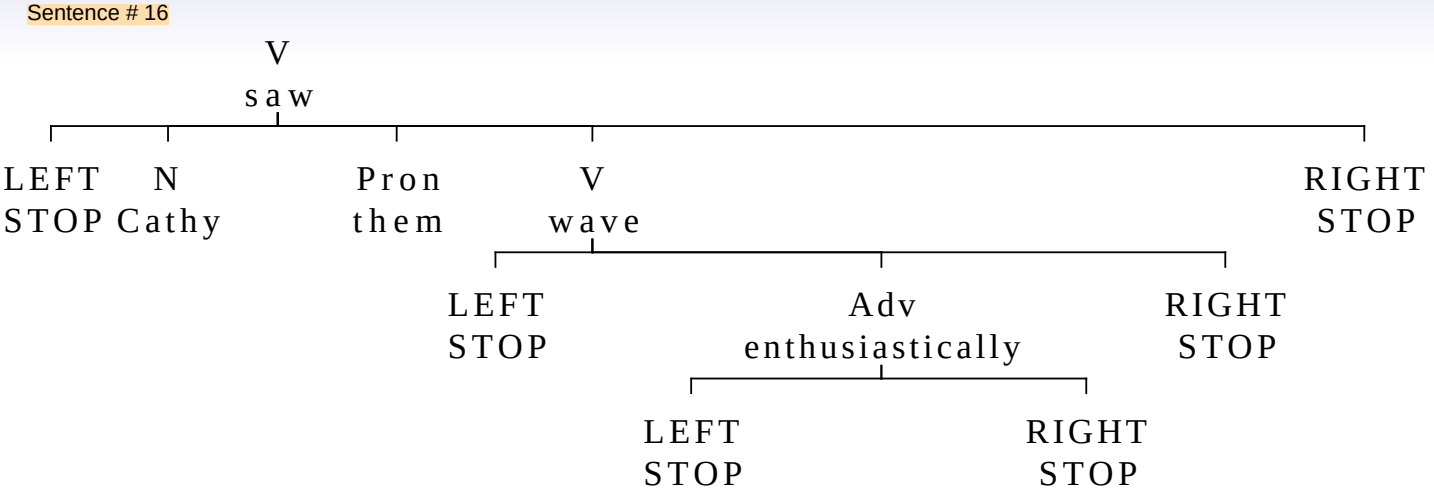

Sentence # 13
V
saw
Pron
them
V
wave
LEFT
STOP
Adv
enthusiastically
RIGHT
STOP
LEFT
STOP
RIGHT
STOP

Sentence # 14




Sentence # 15




Sentence # 16
V
saw
LEFT
STOP
N
Cathy
Pron
them
V
wave
RIGHT
STOP
LEFT
STOP
Adv
enthusiastically
RIGHT
STOP
LEFT
STOP
RIGHT
STOP