Stochastic Parsing

Roberto Basili, Fabio Zanzotto

Department of Computer Science, System and Production
University of Roma, *Tor Vergata*
Via Della Ricerca Scientifica s.n.c., 00133, Roma, ITALY

e-mail: {basili, zanzotto}@info.uniroma2.it

Outline:

- Stochastic Processes and Methods
- POS tagging as a stochastic process
- Probabilistic Approaches to Syntactic Analysis
  - CF grammar-based approaches (e.g. PCFG)
  - Other Approaches (lexicalized or dependency based)
- Further Relevant Issues

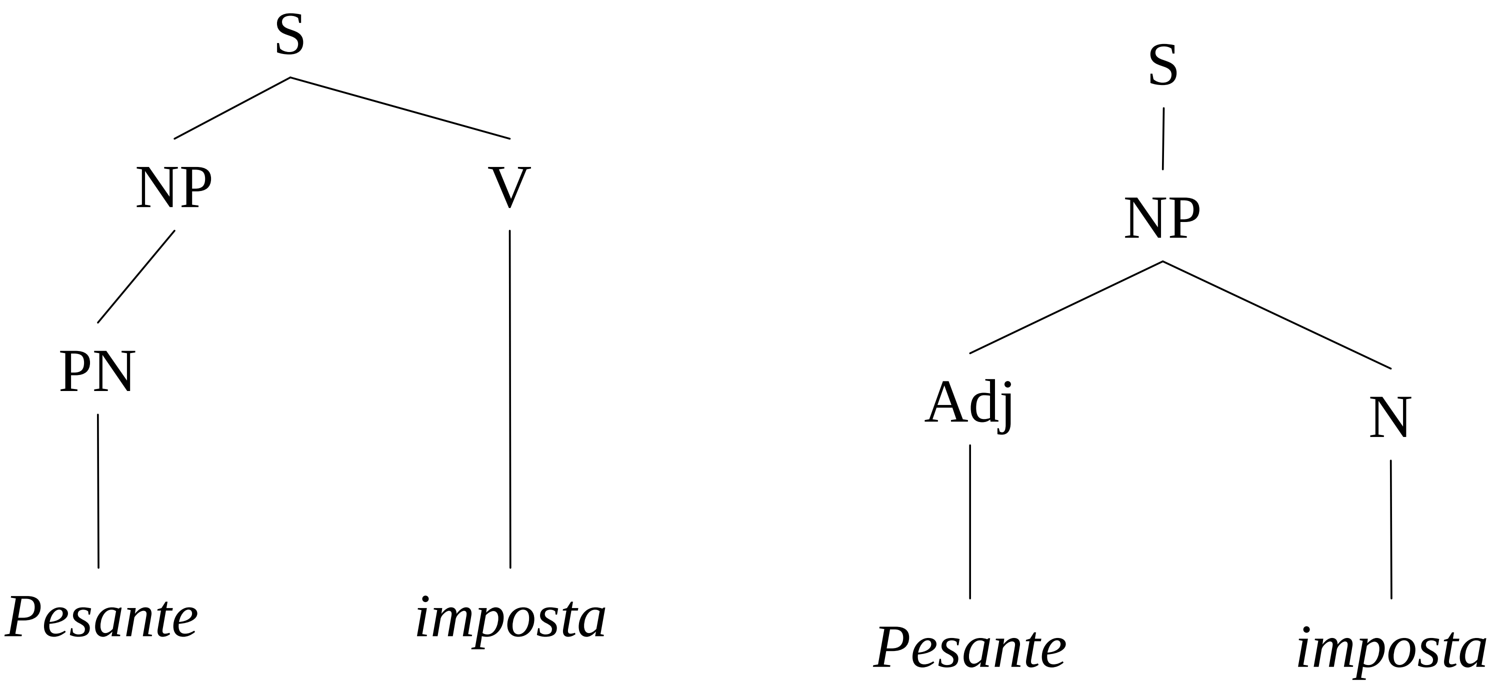
The role of Quantitative Approaches

Weighted grammars are models of the degree of grammaticality able to deal with disambiguation:

```
1.   S  -> NP  V
2.   S  -> NP
3.   NP -> PN
4.   NP -> N
5.   NP -> Adj N
6.   N  -> imposta
7.   V  -> imposta
8.  Adj -> Pesante
9.   PN -> Pesante
...
```

The role of Quantitative Approaches

"Pesante imposta"



Derivation Trees for the sentence "Pesante imposta"

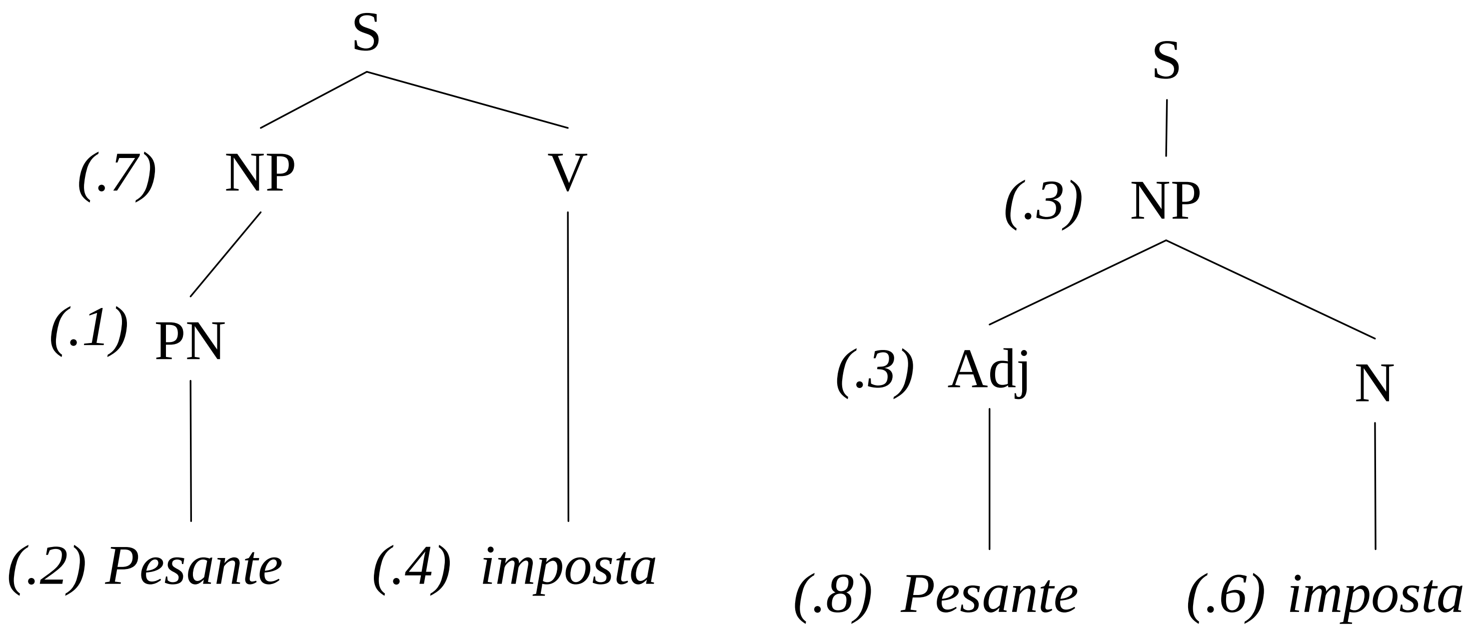
The role of Quantitative Approaches

Weighted grammars are models of the degree of grammaticality able to deal with disambiguation:

```
1.   S  -> NP  V       .7
2.   S  -> NP          .3
3.   NP -> PN          .1
4.   NP -> N           .6
5.   NP -> Adj N       .4
6.   N  -> imposta     .6
7.   V  -> imposta     .4
8.  Adj -> Pesante     .8
9.   PN -> Pesante     .2
...
```

The role of Quantitative Approaches

"Pesante imposta"



Derivation Trees for the sentence "Pesante imposta"

The role of Quantitative Approaches

Weighted grammars are models of the degree of grammaticality able to deal with disambiguation:

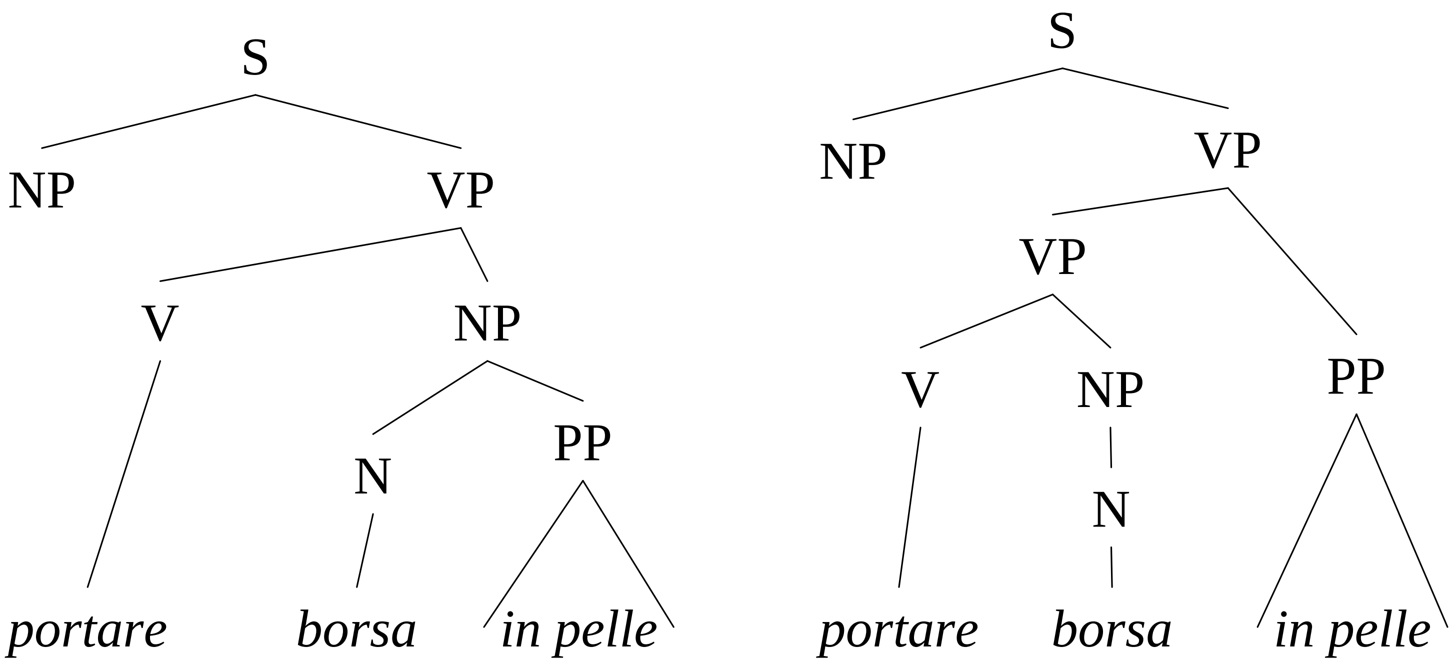
```
1.    S  -> NP  V        .7
2.    S  -> NP           .3
3.    NP -> PN           .1
4.    NP -> N            .6
5.    NP -> Adj N        .3
6.    N  -> imposta      .6
7.    V  -> imposta      .4
8.   Adj -> Pesante      .8
9.    PN -> Pesante      .2
...
```

prob(((Pesante)$_{PN}$ (imposta)$_V$)$_S$)= (.7 * .1 * .2 * .4) = 0.0084

prob(((Pesante)$_{Adj}$ (imposta)$_N$)$_S$)= (.3 * .3 * .8 * .6) = 0.0432

Structural Disambiguation

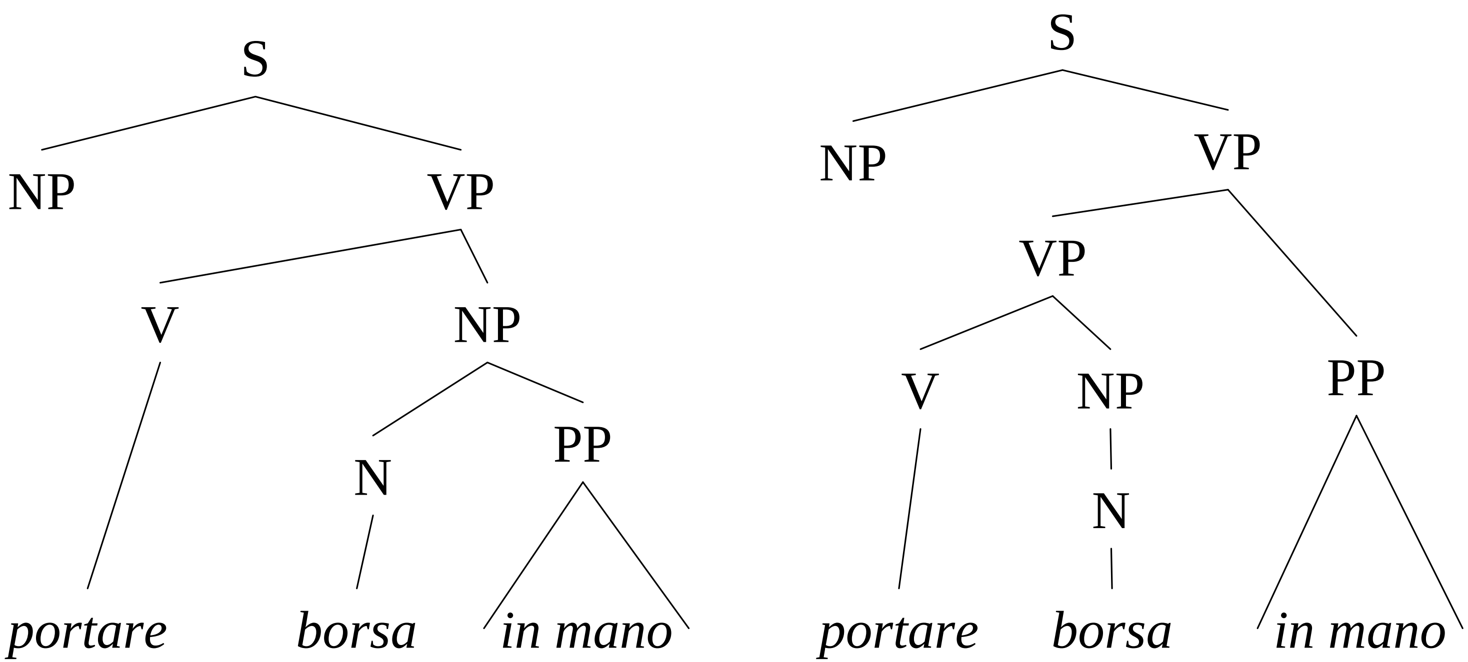
"*portare borsa in pelle*"



Derivation Trees for a structurally ambiguous sentence.

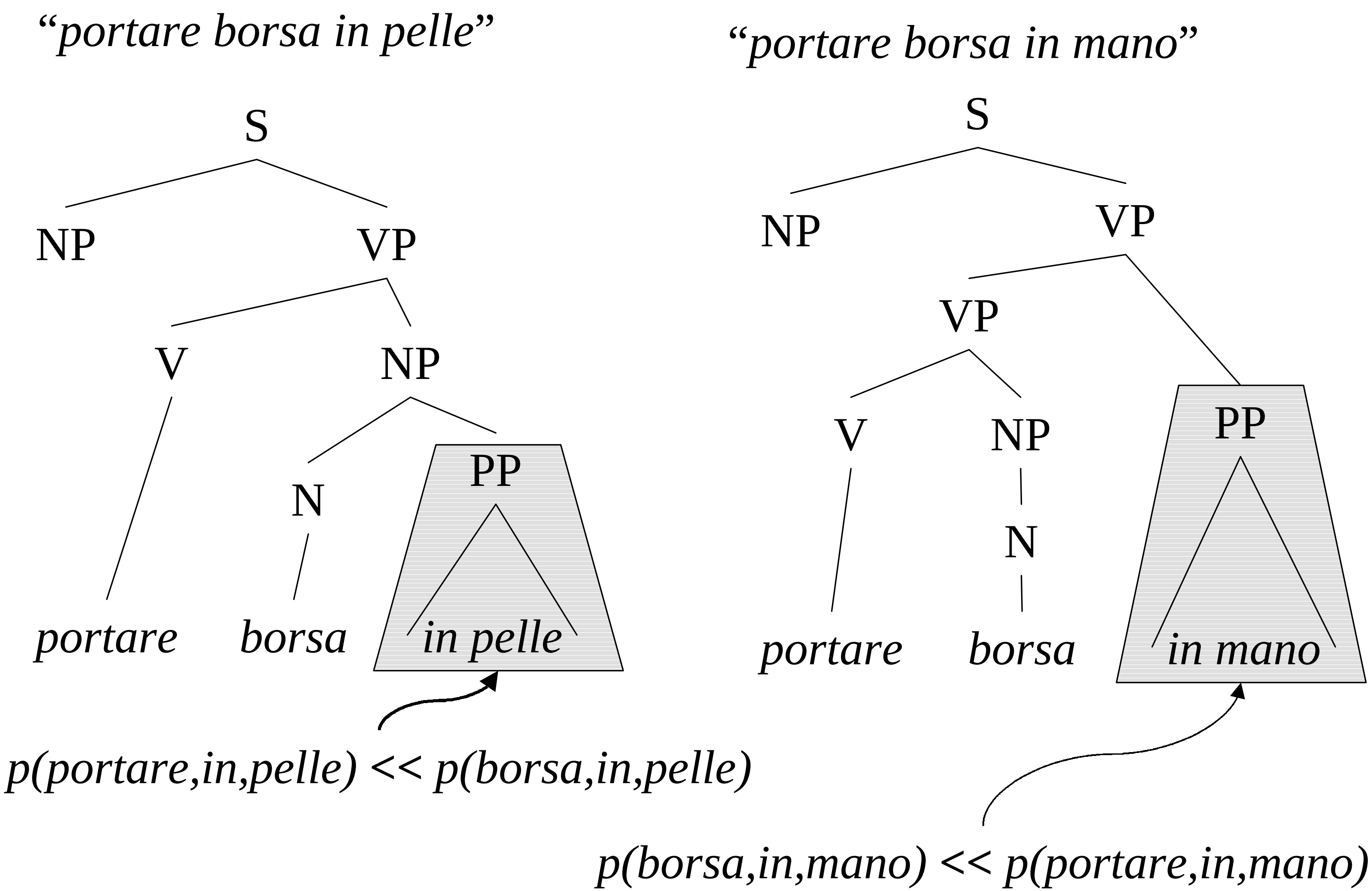
Structural Disambiguation (cont'd)

"*portare borsa in mano*"

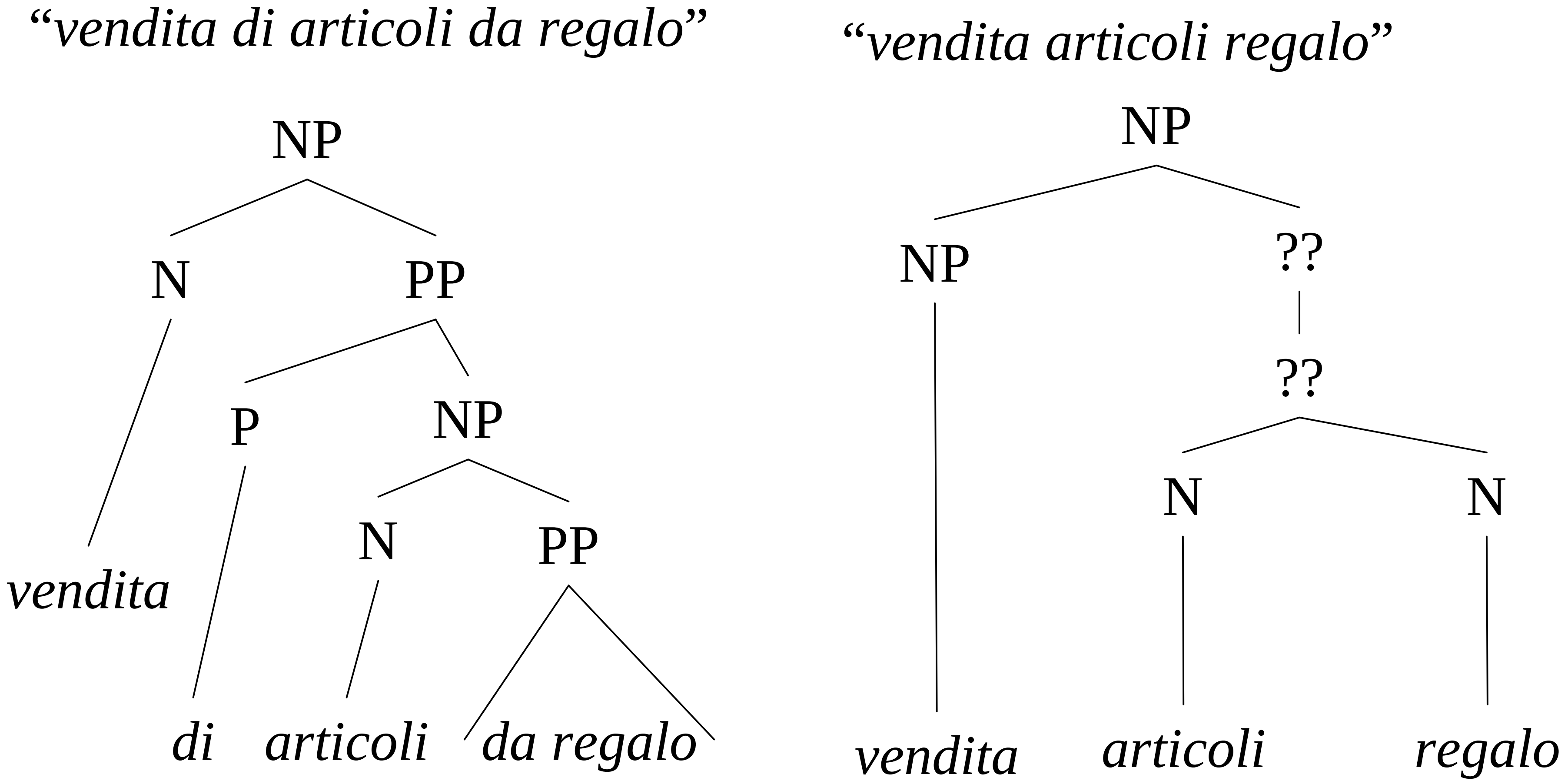


Derivation Trees for a second structurally ambiguous sentence.

Struc

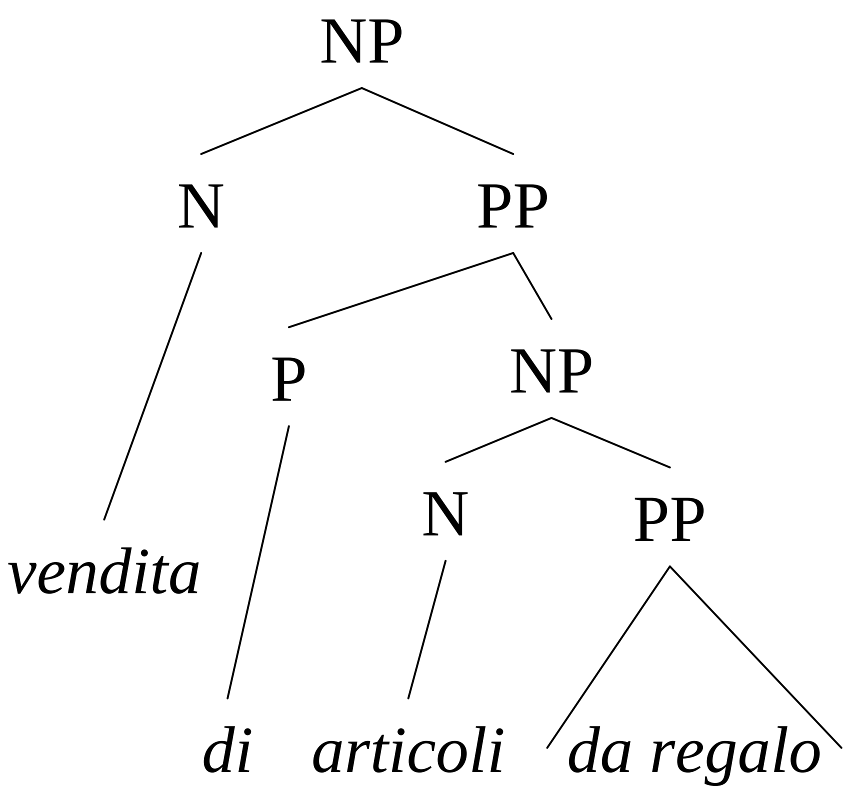


Disambiguation of structural ambiguity

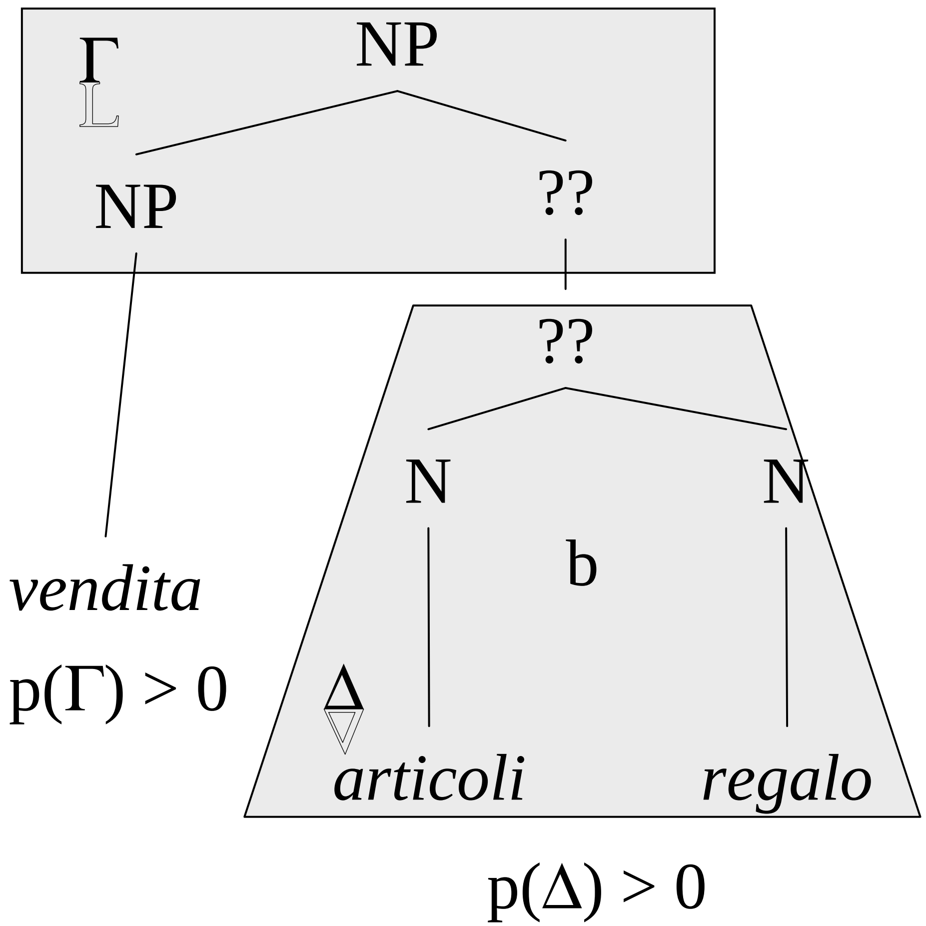


An example of ungrammatical but meaningful sentence.

“*vendita di articoli da regalo*”



“*vendita articoli regalo*”



Probability and Language Modeling

- Aims
  - to extend existing models with predictive and disambiguation capabilities
  - to offer theoretically well founded inductive methods
  - to develop (not-so) quantitative models of linguistic phenomena

- Methods and Resources:
  - Methematical theories (e.g. Markov models)
  - Systematic testing/evaluation frameworks
  - Extended repositories of language use instances
  - Traditional linguistic resources (e.g. "models" like dictionaries)

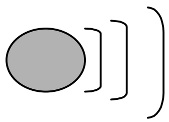
Probability and the Empiricist Renaissance (2)

- Differences
  - amount of knowledge available *a priori*
  - target: *competence* vs. *performance*
  - methods: *deduction* vs. *induction*
- The role of probability in NLP is also related to:
  - difficulties in categorial statements in language study (e.g. grammaticality or syntactic categorization)
  - the cognitive nature of language understanding
  - the role of uncertainty

Probability and Language Modeling

- Signals are abstracted via symbols not known in advance
- Emitted signals belong to an alphabet A

ne



…, 8, 7, 6, 5, 4, 3, 2, 1

…, w_{i8}, w_{i7}, w_{i6}, w_{i5}, w_{i4}, w_{i3}, w_{i2}, w_{i1}

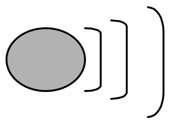
Probability and Language Modeling

- A random variable X can be introduced so that
  - It assumes values w_i in the alfabet A
  - Probability is used to describe the uncertainty on the emitted signal

$$p(X = w_i) \qquad w_i \in A$$

Probability and Language Modeling

- A random variable X can be introduced so that
  - X assumes values in A at each step i, i.e. $X_i = w_j$
  - probability is $p(X_i = w_j)$



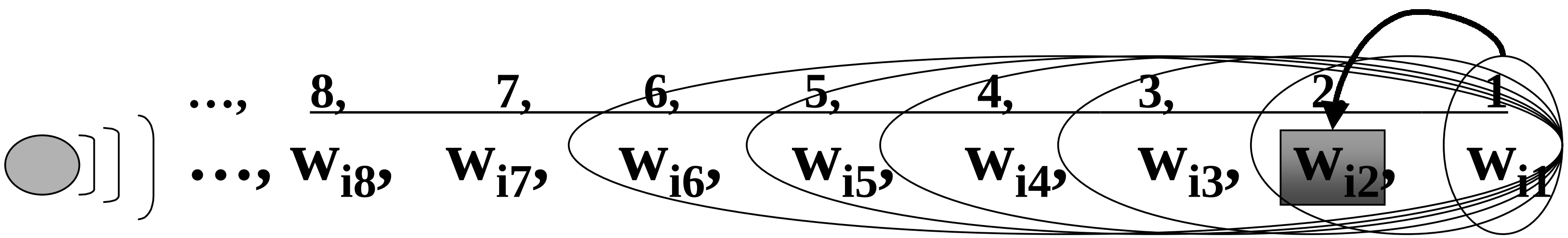
..., 8, 7, 6, 5, 4, 3, 2, 1

..., w_{i8}, w_{i7}, w_{i6}, w_{i5}, w_{i4}, w_{i3}, w_{i2}, w_{i1}

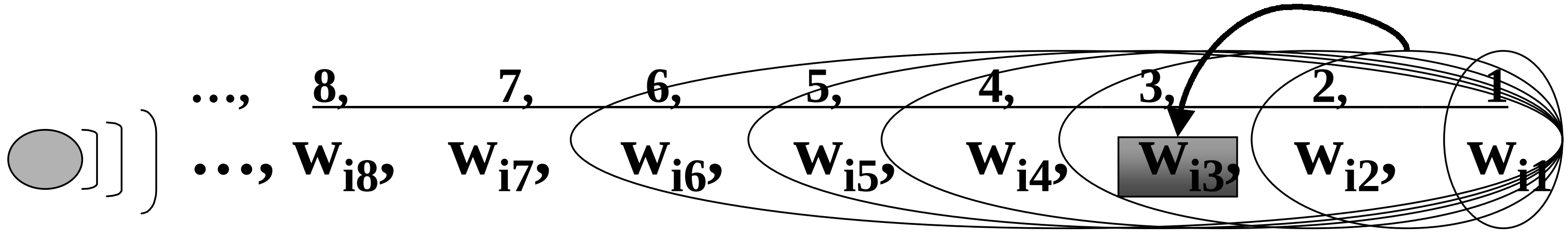
Probability and Language Modeling

- Notice that time points can be represented as **states** of the emitting

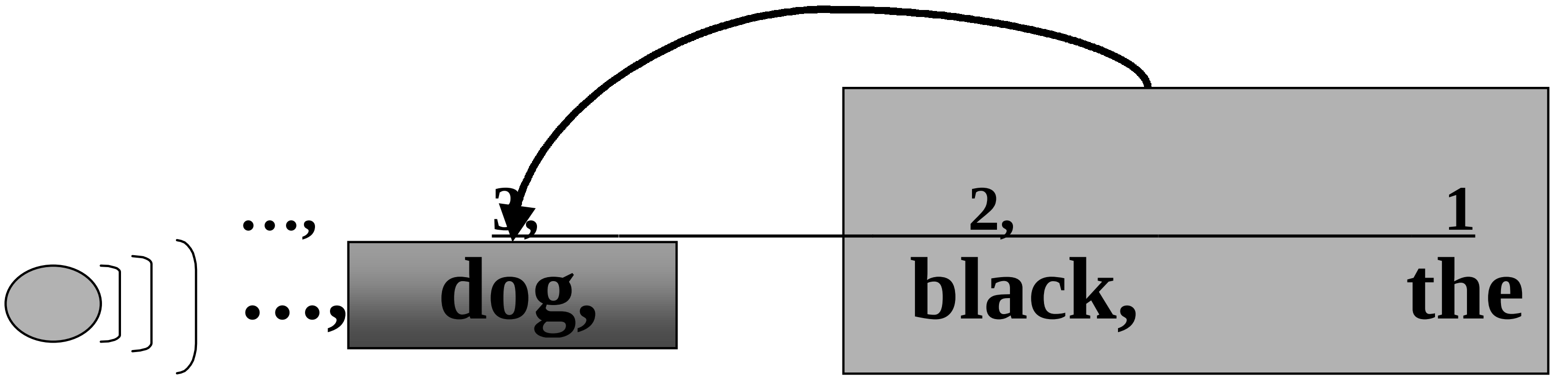
- the



Probability and Language Modeling

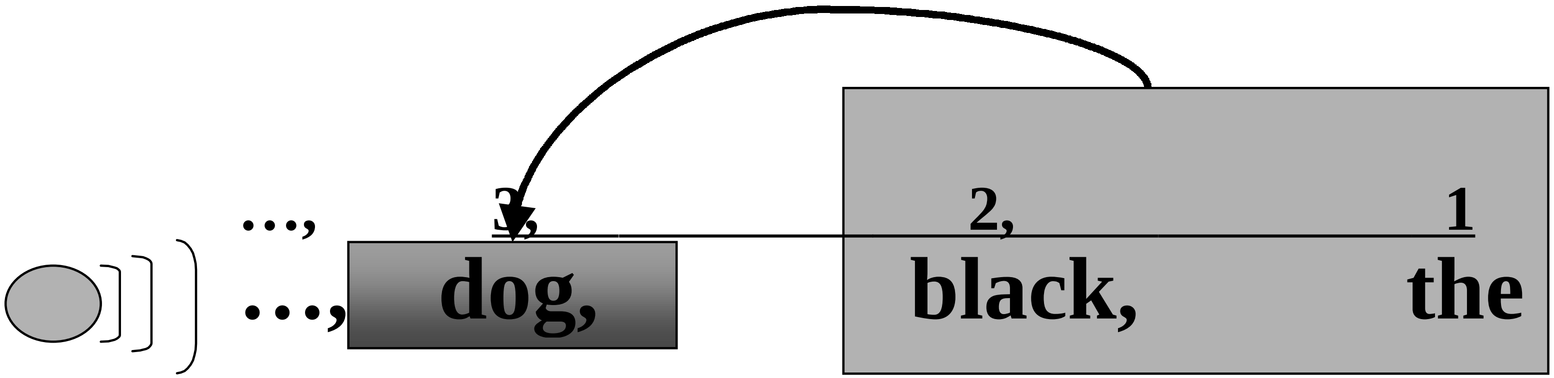


Probability and Language Modeling

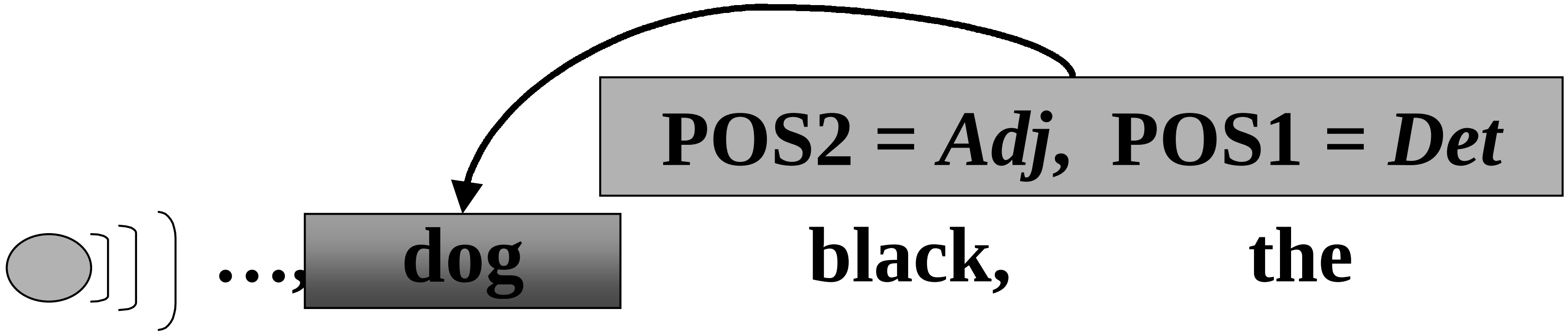


- $p(the, black, dog) = p(dog|the, black) \ldots$

Probability and Language Modeling



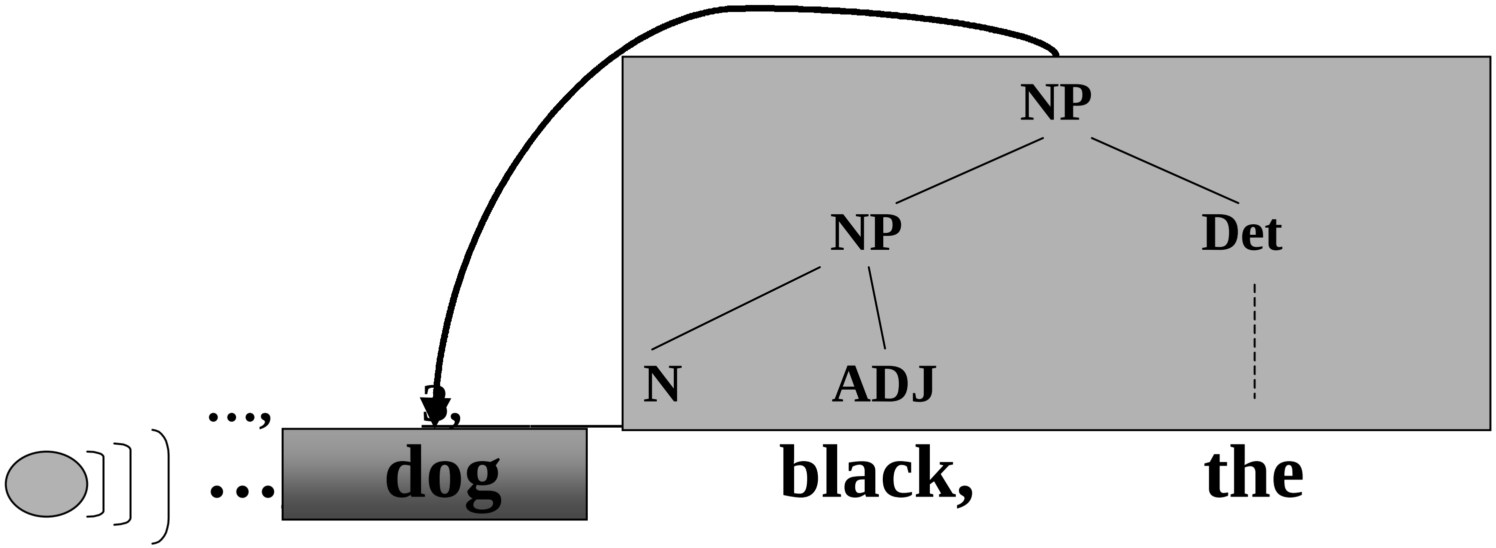
- $p(the, black, dog) = p(dog|the, black)p(black|the)p(the)$



- $p(the_{DT}, black_{ADJ}, dog_N) = p(dog_N | the_{DT}, black_{ADJ}) \ldots$

Probability and Language Modeling

- **What's in a state**
  - preceding parses → stochastic grammars



- $p((the_{Det},(black_{ADJ},dog_N)_{NP})_{NP}) = p(dog_N|((the_{Det}),(black_{ADJ},\_)))\ldots$

Probability and Language Modeling (2)

- Expressivity
  - The predictivity of a statistical device can be very good explanatory model of the source information
  - Simpler and Systematic Induction
  - Simpler and Augmented Description (e.g. grammatical preference)
  - Optimized Coverage (better on more important phenomena)
- Integrating Linguistic Description
  - Start with poor assumptions and approximate as much as possible *what is known* (evaluate performance only)
  - *Bias* the statistical model since the beginning and check the results on a *linguistic ground*

Probability and Language Modeling (3)

Performances

- Faster Processing
- Faster Design
- Linguistic Adequacy
  - Acceptance
  - Psychological Plausibility
  - Explanatory power
- Tools for further analysis of Linguistic Data

Markov Models

Suppose $X_1, X_2, ..., X_T$ form a sequence of random variables taking values in a countable set $W = p_1, p_2, ..., p_N$ (State space).

- Limited Horizon Property:
  $P(X_{t+1} = p_k | X_1, ..., X_t) = P(X_{t+1} = k | X_t)$

- Time invariant:
  $P(X_{t+1} = p_k | X_t = p_l) = P(X_2 = p_k | X_1 = p_l) \qquad \forall t (> 1)$

It follows that the sequence of $X_1, X_2, ..., X_T$ is a **Markov chain**.

Representation of a Markov Chain

Matrix Representation:

- A (transition) matrix A:

$$a_{ij} = P(X_{t+1} = p_j | X_t = p_i)$$

Note that $\forall i, j \quad a_{ij} \geq 0$ and $\forall i \quad \sum_j a_{ij} = 1$

- Initial State description (i.e. probabilities of initial states):

$$\pi_i = P(X_1 = p_i)$$

Note that $\sum_{j=1}^{n} \pi_{ij} = 1$.

Representation of a Markov Chain

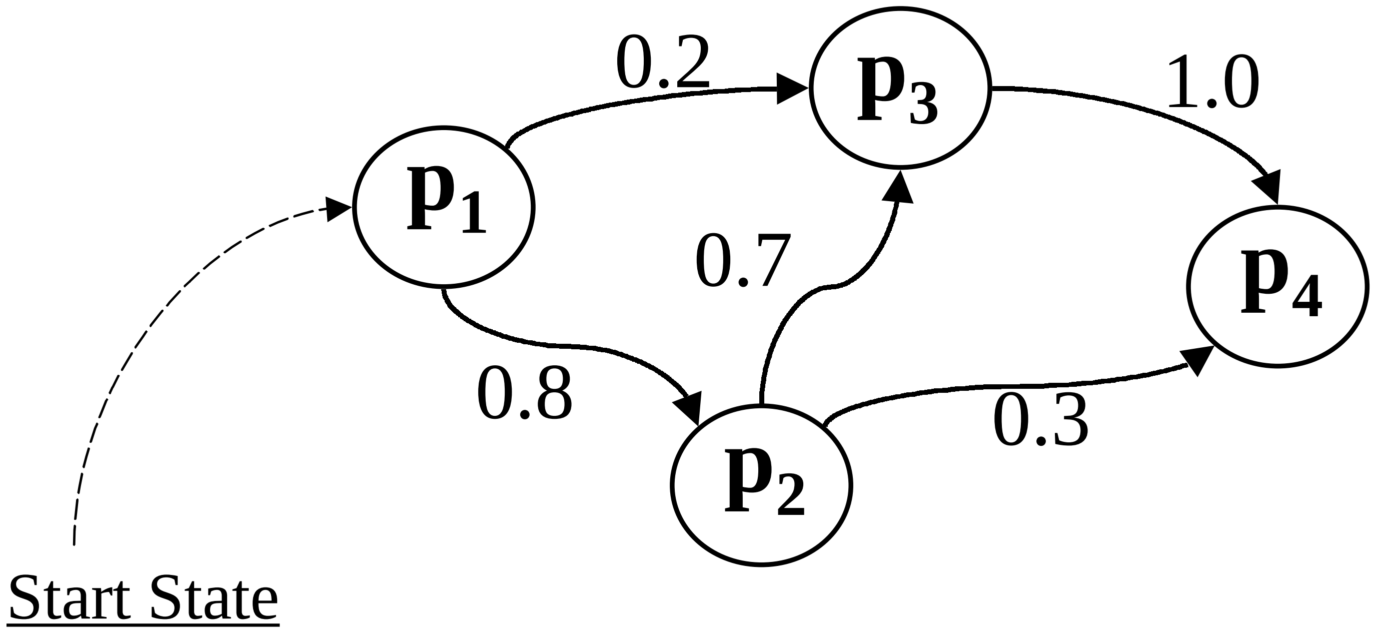
Graphical Representation (i.e. Automata)

- States as nodes with names

- Transitions from states i-th and j-th as arcs labelled by conditional probabilities $P(X_{t+1} = p_j | X_t = p_i)$
  Note that 0 probability arcs are omitted from the graph.

Representation of a Markov Chain

- Gr



A Simple Example of Hidden Markov Model

Crazy Coffee Machine

- Two states: Tea Preferring (TP), Coffee Preferring (CP)
- Switch from one state to another randomly
- Simple (or visible) Markov model:
  Iff the machine output Tea in TP AND $Coffee$ in CP

What we need is a description of the random event of switching from one state to another. More formally we need for each time step n and couple of states p_i and p_j to determine following conditional probabilities:

$$P(X_{n+1} = p_j | X_n = p_i)$$

where p_t is one of the two states *TP*, *CP*.

A Simple Example of Hidden Markov Model

Crazy Coffee Machine

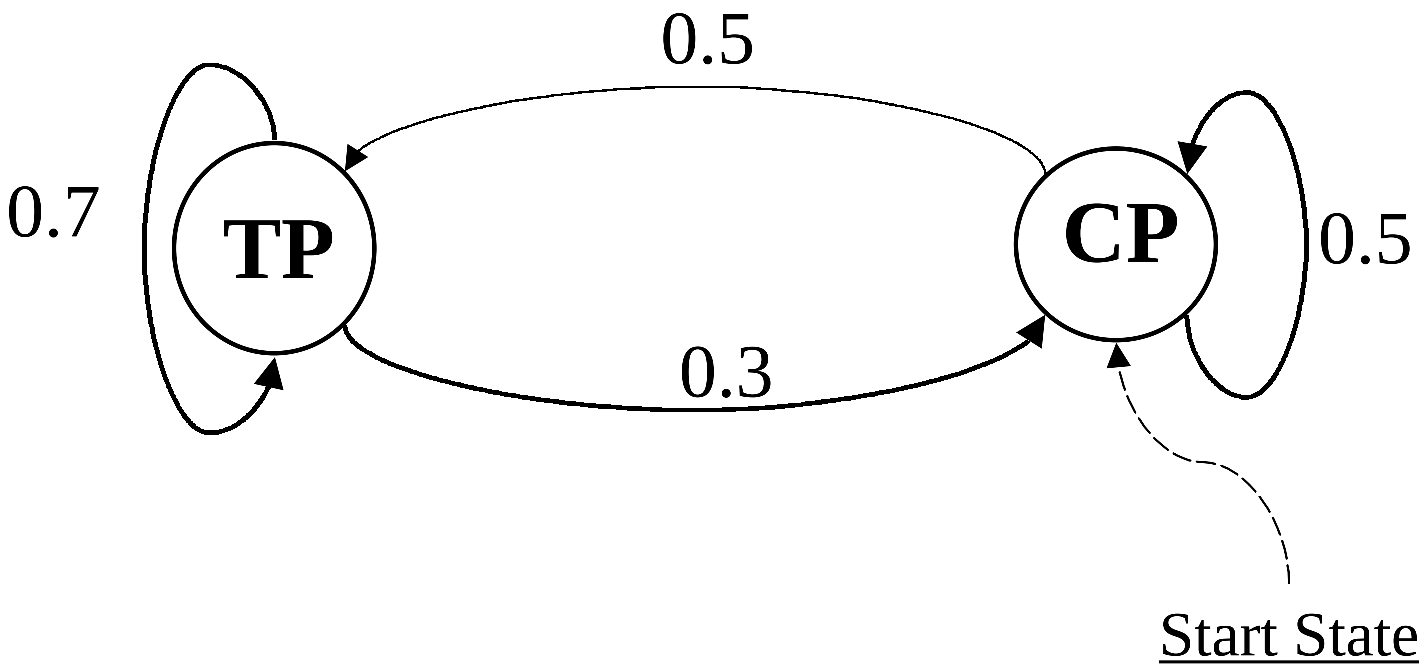
Assume, for example, the following state transition model:

	TP	CP
TP	0.70	0.30
CP	0.50	0.50

and let CP be the starting state (i.e. $\pi_{CP} = 1$, $\pi_{TP} = 0$).

Potential Use:

- Which is the probability at time step 3 to be in state TP
- Which is the probability at time step n to be in state TP
- Which is the probability of the following sequence in output $(Coffee, Tea, Coffee)$


0.5
0.7
TP
CP
0.5
0.3
Start State

Crazy Coffee Machine

Solutions:

- $P(X_3 = TP) =$ (given by (CP, CP, TP) *and* (CP, TP, TP))

  $$= P(X_1 = CP) * P(X_2 = CP|X_1 = CP) * P(X_3 = TP|X_1 = CP, X_2 = CP) +$$
  $$+ \quad P(X_1 = CP) * P(X_2 = TP|X_1 = CP) * P(X_3 = TP|X_1 = CP, X_2 = TP) =$$

  $$= P(CP)P(CP|CP)P(TP|CP, CP) + P(CP)P(TP|CP)P(TP|CP, TP) =$$
  $$= P(CP)P(CP|CP)P(TP|CP) + P(CP)P(TP|CP)P(TP|TP) =$$
  $$= 1 * 0.50 * 0.50 + 1 * 0.50 * 0.70 = 0.25 + 0.35 = 0.60$$

- In the general case,
  $P(X_n = TP) =$
  $\sum_{CP, p_2, p_3, \ldots, TP} P(X_1 = CP)P(X_2 = p_2|X_1 = CP)P(X_3 = p_3|X_1 = CP, X_2 = p_2) * \ldots *$
  $P(X_n = TP|X_1 = CP, X_2 = p_2, \ldots, X_{n-1} = p_{n-1}) =$
  $= \sum_{CP, p_2, p_3, \ldots, TP} P(CP)P(p_2|CP)P(p_3|p_2) * \ldots * P(TP|p_{n-1}) =$
  $= \sum_{CP, p_2, p_3, \ldots, TP} P(CP) * \prod_{t=1}^{n-1} P(p_{t+1}|p_t) =$
  $= \sum_{p_1, \ldots, p_n} P(p_1) * \prod_{t=1}^{n-1} P(p_{t+1}|p_t)$

- $P(Cof, Tea, Cof) =$
  $= P(Cof) * P(Tea|Cof) * P(Cof|Tea) = 1 * 0.5 * 0.3 = 0.15$

A Simple Example of Hidden Markov Model (2)

Crazy Coffee Machine

- **Hidden** Markov model: If the machine output *Tea*, *Coffee* or *Capuccino* independently from CP and TP.

What we need is a description of the random event of output(ting) a drink.

Crazy Coffee Machine

A description of the random event of output(ting) a drink.

More formally we need (for each time step n and for each kind of output $O = \{Tea, Cof, Cap\}$), the following conditional probabilities:

$$P(O_n = k | X_n = p_i, X_{n+1} = p_j)$$

where k is one of the values *Tea*, *Coffee* or *Capuccino*.
This matrix is called the **output matrix** of the machine (or of its Hidden markov Model).

A Simple Example of Hidden Markov Model (2)

Crazy Coffee Machine

Given the following output probability for the machine

	Tea	Coffee	Capuccino
TP	0.8	0.2	0.0
CP	0.15	0.65	0.2

and let CP be the starting state (i.e. $\pi_{CP} = 1$, $\pi_{TP} = 0$).

- Find the following probabilities of output from the machine
  1. $(Cappuccino, Coffee)$ given that the state sequence is (CP, TP, TP)
  2. $(Tea, Coffee)$ for any state sequence
  3. a generic output $O = (o_1, ..., o_n)$ for *any* state sequence

A Simple Example of Hidden Markov Model (2)

Solution for the problem 1

- For the given state sequence $X = (CP, TP, TP)$
  $P(O_1 = Cap, O_2 = Cof, X_1 = CP, X_2 = TP, X_3 = TP) =$
  $P(O_1 = Cap, O_2 = Cof | X_1 = CP, X_2 = TP, X_3 = TP) P(X_1 = CP, X_2 = TP, X_3 = TP)) =$
  $P(Cap, Cof | CP, TP, TP) P(CP, TP, TP))$

  Now:
  $P(Cap, Cof | CP, TP, TP)$ is the probability of output Cap, Cof *during* transitions from CP to TP and TP to TP

  and

  $P(CP, TP, TP)$ is the probability of the transition chain.

  Therefore,
  $= P(Cap|CP, TP) P(Cof|TP, TP) =$(in our simplified model)
  $= P(Cap|CP) P(Cof|TP) = 0.2 * 0.2 = 0.04$

A Simple Example of Hidden Markov Model (2)

Solutions for the problem 2

In general, for any sequence of three states $X = (X_1, X_2, X_3)$
$P(Tea, Cof|X_1, X_2, X_3) =$
$P(Tea, Cof) =$ (as sequences are a partition for the sample space)
$= \sum_{X_1,X_2,X_3} P(Tea, Cof|X_1, X_2, X_3)P(X_1, X_2, X_3)$

where

$P(Tea, Cof|X_1, X_2, X_3) = P(Tea|X_1, X_2)P(Cof|X_2, X_3) =$
(for the simplified model of the coffee machine)

$= P(Tea|X_1)P(Cof|X_2)$
and (for the Markov constraint) $P(X_1, X_2, X_3) = P(X_1)P(X_2|X_1)P(X_3|X_2)$

The simplified model is concerned with only the following transition chains
(CP, CP, CP)
(CP, TP, CP)
(CP, CP, TP)
(CP, TP, TP)

so that the following probability is given

$$
\begin{array}{rll}
P(Tea, Cof) & = & \\
& P(Tea|CP)P(Cof|CP)P(CP)P(CP|CP)P(CP|CP)+ & \text{states: } (CP, CP, CP)) \\
& P(Tea|CP)P(Cof|TP)P(CP)P(TP|CP)P(CP|TP)+ & \text{states: } (CP, TP, CP)) \\
& P(Tea|CP)P(Cof|CP)P(CP)P(CP|CP)P(TP|CP)+ & \text{states: } (CP, CP, TP)) \\
& P(Tea|CP)P(Cof|TP)P(CP)P(TP|CP)P(TP|TP) = & \text{states: } (CP, TP, TP))
\end{array}
$$

$= 0.15 * 0.65 * 1 * 0.5 * 0.5+$

$+ \quad 0.15 * 0.2 * 1 * 0.5 * 0.3+$

$+ \quad 0.15 * 0.65 * 1 * 0.5 * 0.5+$

$+ \quad 0.15 * 0.2 * 1.0 * 0.5 * 0.7 =$

$= 0.024375 + 0.0045 + 0.024375 + 0.0105 =$

$= 0.06375$

A Simple Example of Hidden Markov Model (2)

Solution to the problem 3 (*decoding*)

In the general case, a sequence of n symbols $O = (o_1, ..., o_n)$ out from any sequence of $n + 1$ transitions $X = (p_1, ..., p_{n+1})$
can be predicted by the following probability:

$$
\begin{aligned}
P(O) &= \textstyle\sum_{p_1,...,p_{n+1}} P(O|X)P(X) = \\
&= \textstyle\sum_{p_1,...,p_{n+1}} P(CP) \prod_{t=1}^{n} P(O_t|p_t, p_{t+1})P(p_{t+1}|p_t)
\end{aligned}
$$

Modeling linguistic tasks of Stochastic Processes

Outline

- Available mathematical frameworks
- Questions for Stochastic models
- Modeling POS tagging via HMM

Mathematical Methods for HMM

The complexity of training and decoding can be limited by the use of optimization techniques

- Parameter estimation via entropy minimization (EM)
- Tagging and Decoding via dynamic programming ($O(n)$)

A relevant issue is the availablity of source data so that supervised training can (or cannot) be applied

HMM and POS tagging

Given a sequence of morphemes $w_1, ..., w_n$ with ambiguous syntactic descriptions (i.e.part-of-speech) tag, derive the sequence of n POS tags $t_1, ..., t_n$ that maximizes the following probability:

$$P(w_1, ..., w_n, t_1, ..., t_n)$$

that is
$(t_1, ..., t_n) = argmax_{pos_1, ..., pos_n} P(w_1, ..., w_n, pos_1, ..., pos_n)$

Note that this is equivalent to the following:

$(t_1, ..., t_n) = argmax_{pos_1, ..., pos_n} P(pos_1, ..., pos_n | w_1, ..., w_n)$

as: $\frac{P(w_1, ..., w_n, pos_1, ..., pos_n)}{P(w_1, ..., w_n)} = P(pos_1, ..., pos_n | w_1, ..., w_n)$

and $P(w_1, ..., w_n)$ is the same for all the sequencies $(pos_1, ..., pos_n)$.

HMM and POS tagging

How to map a POS tagging problem into a HMM.

The following problem

$$(t_1, ..., t_n) = argmax_{pos_1,...,pos_n} P(pos_1, ..., pos_n | w_1, ..., w_n)$$

can be also written (Bayes law) as:

$$(t_1, ..., t_n) = argmax_{pos_1,...,pos_n} P(w_1, ..., w_n | pos_1, ..., pos_n) P(pos_1, ..., pos_n)$$

HMM and POS tagging

The HMM Model of POS tagging:

- **HMM States are mapped into POS tags** (t_i), so that $P(t_1, ..., t_n) = P(t_1)P(t_2|t_1)...P(t_n|t_{n-1})$
- **HMM Output symbols are words**, so that $P(w_1, ..., w_n|t_1, ..., t_n) \quad = \quad \prod_{i=1}^{n} P(w_i|t_i)$
- Transitions represent moves from one word to another

Note that *the Markov assumption is used*

- to model probability of a tag in position i (i.e. t_i) only by means of the preceeding part-of-speech (i.e. t_{i-1})
- to model probabilities of words (i.e. w_i) based only on the tag (t_i) appearing in that position (i).

HMM and POS tagging

The final equation is thus:

$$(t_1, ..., t_n) = argmax_{t_1,...,t_n} P(t_1, ..., t_n | w_1, ..., w_n) = \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

Fundamental Questions for HMM in POS tagging

1. Given a sample of the output sequences and a space of possible models how to find out the best model, that is the model that best explains the data:
   how to estimate parameters?

2. Given a model and an observable output sequence O (i.e. words), how to determine the sequence of states $(t_1, ..., t_n)$ such that it is the best explanation of the observation O:
   Statistical Inference

3. Given a model what is the probability of an output sequence, O:
   Decoding Problem.

Fundamental Questions for HMM in POS tagging

- 1. Baum-Welch (or Forward-Backward algorithm), that is a special case of Expectation Maximization estimation.
  Weakly supervised or even unsupervised.
  Problems: Local minima can be reached when initial data are poor.

- 2. Viterbi Algorithm for evaluating $P(W|O)$.
  Linear in the sequence length.

- 3. Not relevant for POS tagging, where $(w_1, ..., w_n)$ are always known.
  Trellis and dynamic programming technique.

HMM and POS tagging

Advantages for adopting HMM in POS tagging

- An elegant and well founded theory
- Training algorithms:
  - Estimation via EM (Baum-Welch)
  - Unsupervised (or possibly weakly supervised)
- Fast Inference algorithms: Viterbi algorithm
  Linear wrt the sequence length ($O(n)$)
- Sound methods for comparing different models and estimations
  (e.g. cross-entropy)

HMM and POS tagging: Parameter Estimation

Supervised methods in tagged data sets:

- Output probs: $P(w_i|p^j) = \frac{C(w_i, p^j)}{C(p^j)}$
- Transition probs: $P(p^i|p^j) = \frac{C(p^i \text{ follows } p^j)}{C(p^j)}$
- Smoothing: $P(w_i|p^j) = \frac{C(w_i, p^j)+1}{C(p^j)+K^i}$

HMM and POS tagging: Parameter Estimation

Unsupervised (few tagged data available):

- With a dictionary: $P(w_i|p^j)$ are early estimated from D, while $P(p^i|p^j)$ are randomly assigned

- With equivalence classes u_L, (Kupiec92):
$$P(w^i|p^L) = \frac{\frac{1}{|L|}C(u^L)}{\sum_{u_{L'}} \frac{C(u^{L'})}{|L'|}}$$
For example, if $L =$\{noun, verb\} then $u_L = \{cross, drive, \}$

Other Approaches to POS tagging

- Church (1988):
  $\prod_{i=n}^{3} P(w_i|t_i)P(t_{i-2}|t_{i-1}, t_i)$ (backward)
  Estimation from tagged corpus (Brown)
  No HMM training
  Perfromances: $> 95\%$

- De Rose (1988):
  $\prod_{i=1}^{n} P(w_i|t_i)P(t_{i-1}|t_i)$ (forward)
  Estimation from tagged corpus (Brown)
  No HMM training Performance: 95%

- Merialdo et al.,(1992), ML estimation vs. Viterbi training
  Propose an incremental approach: small tagging and then Viterbi training

- $\prod_{i=1}^{n} P(w_i|t_i)P(t_{i+1}|t_i, w_i)$???

Statistical Parsing

Outline:

- Parsing and Statistical modeling
- Some Parsing Models
  - Probabistic Context-Free Grammars
  - Left-Corner Grammars (LCG)
  - Dependency-based assumptions
- Systems

Parsing and Statistical modeling

Main Issues in statistical modeling of grammatical recognition:

- Modelling the structure vs. the language
- How context can be taken into account
- Modelling the structure vs. the derivation
- Which Parsing vs. the Model

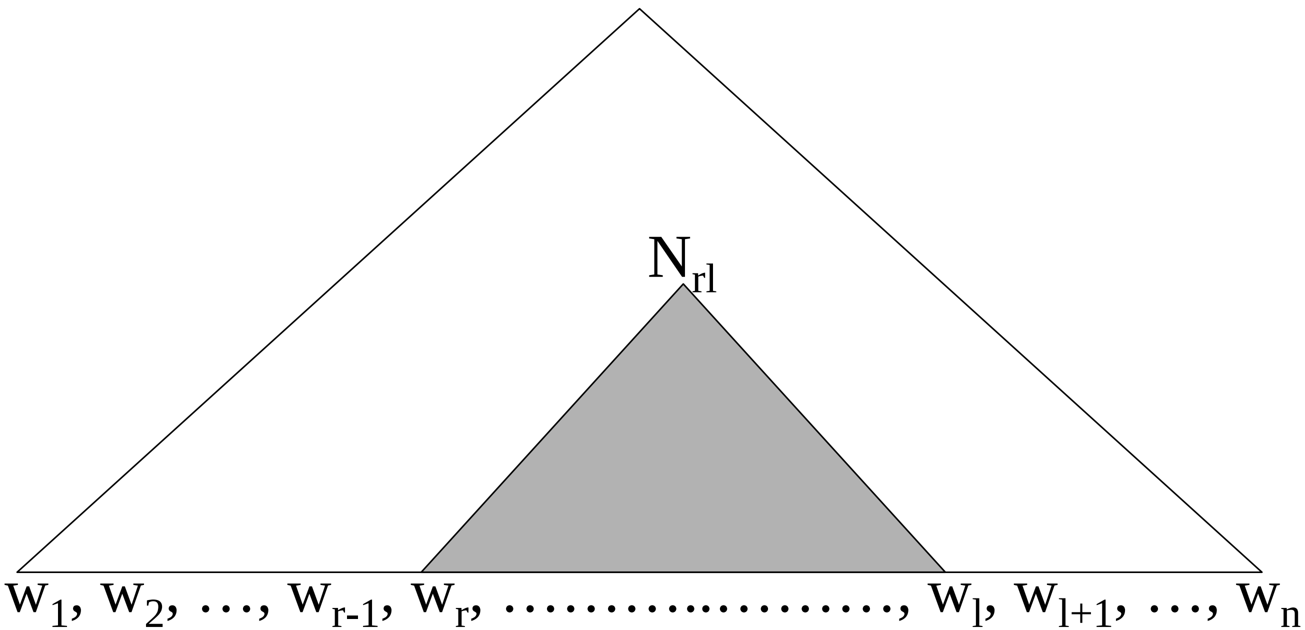
Probabilistic Context-Free Grammars, PCFG

Basic Steps:

- Start from a CF grammar
- Attach probability to the rules
- Tune (i.e. adjust them) against training data (e.g. a treebank)

Probabilistic Context-Free Grammars, PCFG

- **Syntactic sugar**
  - $N_{rl} \rightarrow \Gamma \quad \Rightarrow \quad P(N_{rl} \rightarrow \Gamma)$
  - $P(s) = \sum_t P(t)$, where t are parse trees for s



Probabilistic Context-Free Grammars, PCFG

Main assumptions

- Total probability of each continuation, i.e.
  $\forall i \qquad \sum_j P(N^i \to \Gamma^j) = 1$

- *Context* Independence

  - *Place Invariance*, $\forall k \quad P(N^j_{k(k+const)} \to \Gamma)$ is the same

  - *Context-free*ness, $\forall r, l \quad P(N^j_{rl} \to \Gamma)$ is **independent** from $w_1, .., w_{r-1}$ and $w_{l+1}, ..., w_n$

- Derivation (or *Anchestor*) Independence, i.e.
  $\forall k \quad P(N^j_{rl} \to \Gamma)$ is **independent** from any anchestor node outside N^j_{rl}

PCFG

Main Questions:

- How to develop the best model from observable data (Training)
- Which is the best parse for a sentence
- Which training data?

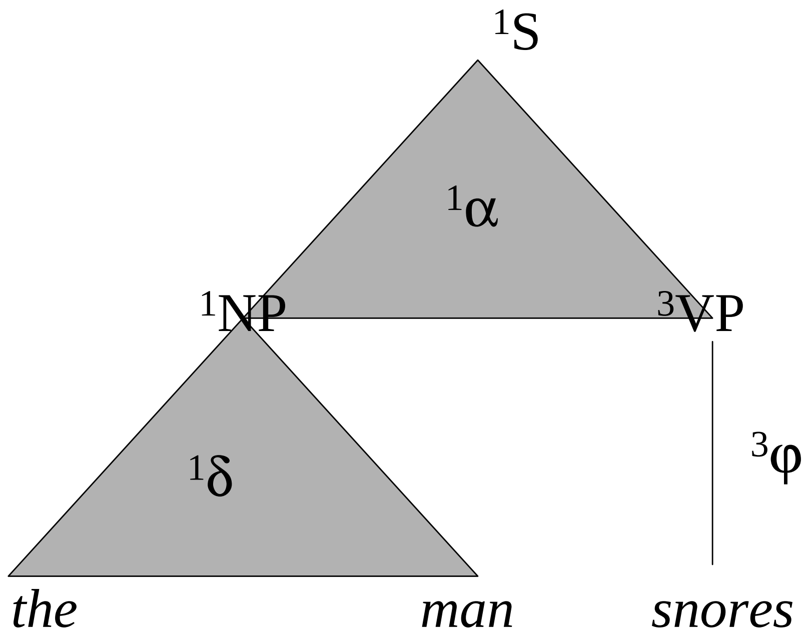
Probabilistic Context-Free Grammars, PCFG

- **Probabilities of trees**
  - $P(t) = P(^1S_{13} \rightarrow ^1NP^3_{12}VP_{33}, ^1NP_{12} \rightarrow the\ man, ^3VP_{33} \rightarrow snores) =$

    $= P(^1\alpha)P(^1\delta|^1\alpha)P(^3\varphi|^1\alpha, ^1\delta) =$

    $= P(^1\alpha)P(^1\delta)P(^3\varphi) =$

    $= P(S$ -



PCFG Training

Supervised case: Treebanks are used.

- Simple ML estimation by counting subtrees
- Basic problems are low counts for some structures and local maxima
- When the target grammatical model is different (e.g. dependency based), rewriting is required before counting

PCFG Training

Unsupervised: Given a grammar G, and a set of sentences (parsed but not validated)

- The best model is the one maximizing the probability of useful structures in observable data set
- Parameters are adjusted via EM algorithms
- $Ranking$ data according to small training data (Pereira and Schabes 1992)

Approaches different from PCFG

- Lexicalized extensions (e.g. LCFG or PLTAG)
- Derivation-based approaches (Left Corner probabilistic parsing): same structure may receive different probability assignments

Statistical Lexicalized Parsing

In (Collins 96) a dependency-based statistical parser is proposed:

- It adopts chunks (Abney,1991) for complex NPs
- Probabilities are assigned to head-modifier dependencies, $(R, < h_j, t_j >, < h_k, t_k >)$
- Dependency relationships are considered independent
- Supervised training is allowed from the treebank

Results

Performances of different Probabilistic Parsers
(Sentences with $\leq$ 40 words)

	%LR	%LP	CB	% 0 CB
Charniak (1996)	80.4	78.8		
Collins (1996)	85.8	86.3	1.14	59.9
Charniak (1997)	87.5	87.4	1.00	62.1
Collins (1997)	88.1	88.6	0.91	66.5

Further Issues in statistical grammatical recognition

- Syntactic Disambiguation
  - PP-attachment
  - Subcategorization frames Induction and Use

- Semantic Disambiguation

- Word Clustering for better estimation (e.g. data sparseness)

Disambiguation of Prepositional Phrase attachment

Purely probabilstic model (Hindle and Rooths, 1991, 1993)

- VP NP PP sequences (i.e. *verb* and *noun* attachment)

- *bi*-gram probabilties $P(prep|noun)$ and $P(prep|verb)$ are compared (via χ^2 test)

- Smoothing low counts is applied for sparse data problems

Disambiguation of Prepositional Phrase attachment

An hybrid model (Basili et al, 1993, 1995):

- Multiple attachment sites in a dependency framework
- Semantic classes for words are available
- semantic *tri*-gram probabilities, i.e. $P(prep, \Gamma|noun)$ and $P(prep, \Gamma|verb)$, are compared
- Smoothing is not required as Γ are classes and low counts are less effective

Bibliographic References

- HMM models
- Corpus-driven POS tagging
- PCFGs
- Statistical Parsing
- Corpus-driven Lexical and Grammatical Acquisition

$\Rightarrow$ see the *handsout*!