Natural Language Processing ECS763P

Statistical Parsing, Ch. 14 of J&F

February 27th 2017

Why go statistical?

The idea behind statistical anything is that

- Sometimes, we <u>cannot have full knowledge</u> about something.
- With <u>enough knowledge</u>, we can figure its <u>probability</u> (and this is better than saying nothing about it).

- One crucial use of statistical parsing is to solve the problem of ambiguity in parsing: we compute the probability of each interpretation and choose the most possible one.

Applications

- Statistical parsers deal better with applications of parsing: Question-Answering, translation, summarisation.
- Statistical parsers are applied to language modelling: when guessing what the next word is, they provide you with the most possible grammatical constructions.
- Experiments from psycholinguistics suggest that humans use some kind of statistical grammar when communicating and doing other language tasks.

Outline

- Probabilistic Context Free Grammars (PCFG's): the most commonly used probabilistic grammar.
- How they learn the probabilities: by being trained on an annotated Treebank
- A basic statistical parsing algorithm (PCKY).
- Problems with PCFG's.
- If time: evaluation measure for parsers (Parseval)
- Psychological results on human parsing

PCFG

- Probabilistic Context Free Grammars (PCFG's) were first introduced by Booth in 1969.

A tuple of 4 parameters: (N, Σ, R, S)

N	a set of non-terminal symbols
Σ	a set of terminal symbols (disjoint from N)
S	a designated start symbol
R	a set of production rules of the form $A \rightarrow \beta\,[p]$

α a non-terminal

β in $(\Sigma \cup N)^*$

p the probability $P(\beta|A)$

PCFG

- A PCFG is obtained from a CFG by augmenting each rule with a probability $A \rightarrow \beta\, [p]$.
- What is this probability?
- The probability that the given non-terminal A will be expanded to the string β .
- In other words, the conditional probability of the string β given the non-terminal A.
- Other notations: $P(A \rightarrow \beta)$ $P(A \rightarrow \beta | A)$ $P(RHS|LHS)$

- All possible expansions of a non-terminal add up to 1:

$$\Sigma_\beta P(A \rightarrow \beta) = 1$$

Example

S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vt	NP	0.2
VP	$\rightarrow$	VP	PP	0.7
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4

Application to Ambiguity

- A PCFG assigns a probability to each parse T tree of a given sentence S.
- This probability is used to disambiguate T:
  - A probability is assigned to each tree T.
  - The tree with a higher probability is considered to be the more likely one.
- Probability of a T given S is computed as follows:

$$P(T, S) = \Pi_{i=1}^{n} \; P(RHS_i | LHS_i)$$

- The tree with highest probability is chosen as output.

$$P(S) = argmax_T \, P(T)$$

P(T) and P(T,S)

P(T,S) is both the joint probability of a tree and a sentence and the probability of the tree P(T):

Apply the definition of joint probability

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

Since $P(S|T) = 1$

Explanation: T contains all words of S.

Application to Ambiguity

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

Let's read this in words:

Take the product of of the probabilities of

$$RHS_i|LHS_i$$

These are the <u>rules of the parse tree</u>:

Rules that are used to expand each of the non-terminal nodes of the tree.

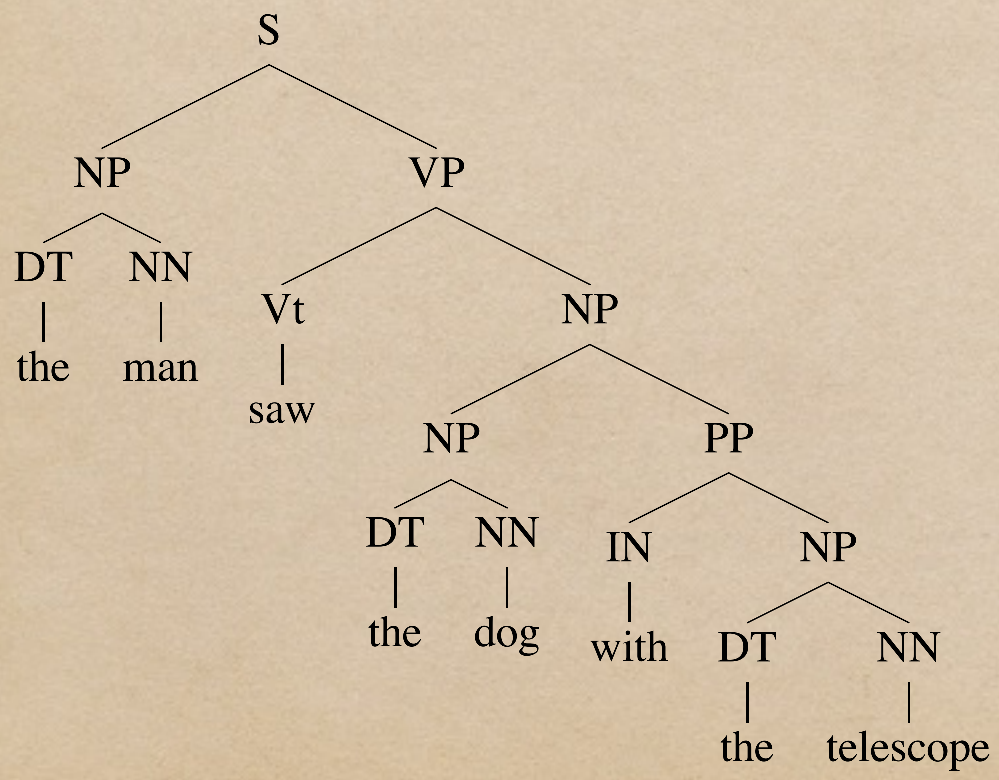
We assume there are n of them. So i is indexing over all of these rules.

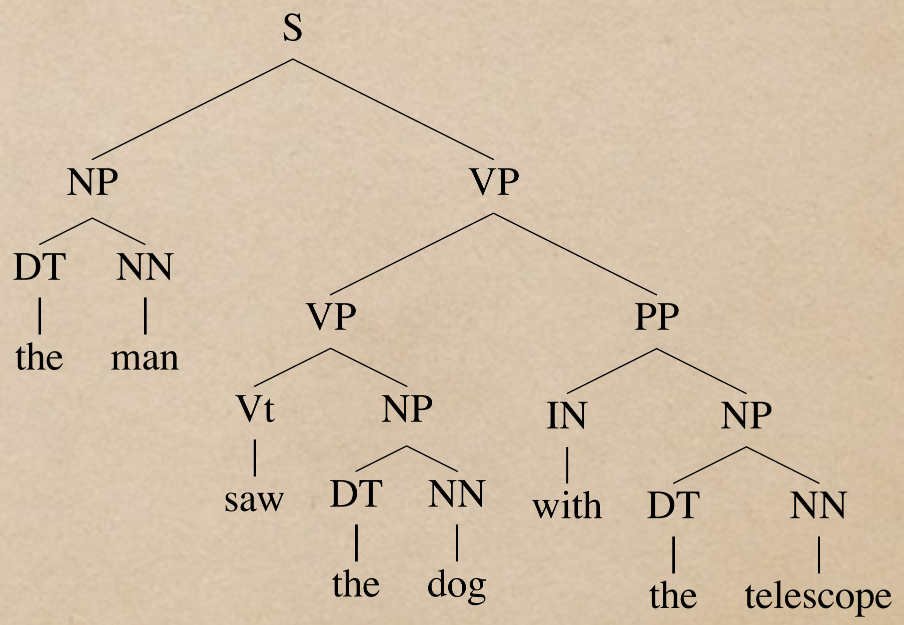
Example

Consider the two parses of "**the man saw the dog with the telescope**". Compute the probability of each parse tree.

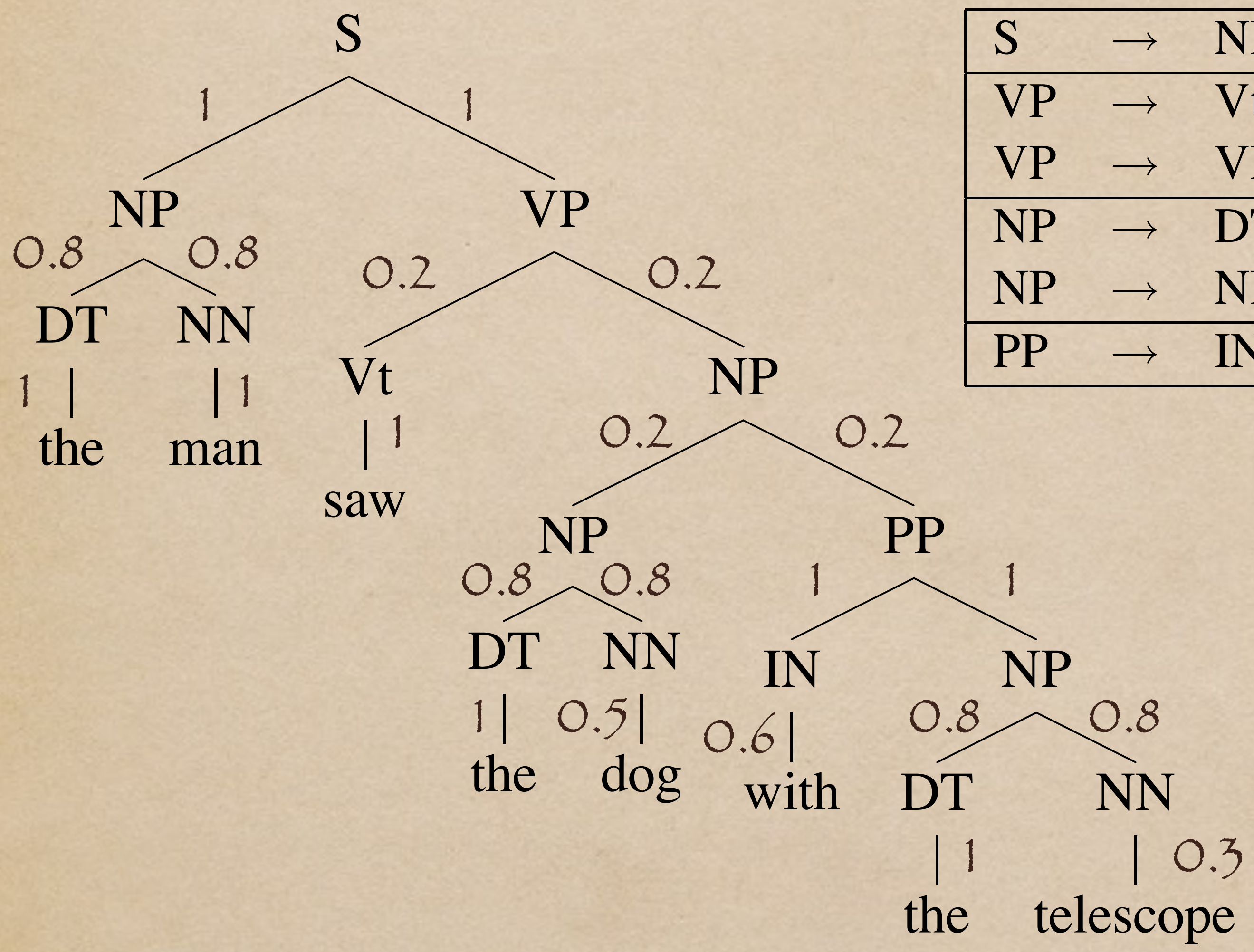
S	→	NP	VP	1.0
VP	→	Vt	NP	0.2
VP	→	VP	PP	0.7
NP	→	DT	NN	0.8
NP	→	NP	PP	0.2
PP	→	IN	NP	1.0

Vi	→	sleeps	1.0
Vt	→	saw	1.0
NN	→	man	0.1
NN	→	woman	0.1
NN	→	telescope	0.3
NN	→	dog	0.5
DT	→	the	1.0
IN	→	with	0.6
IN	→	in	0.4





Example



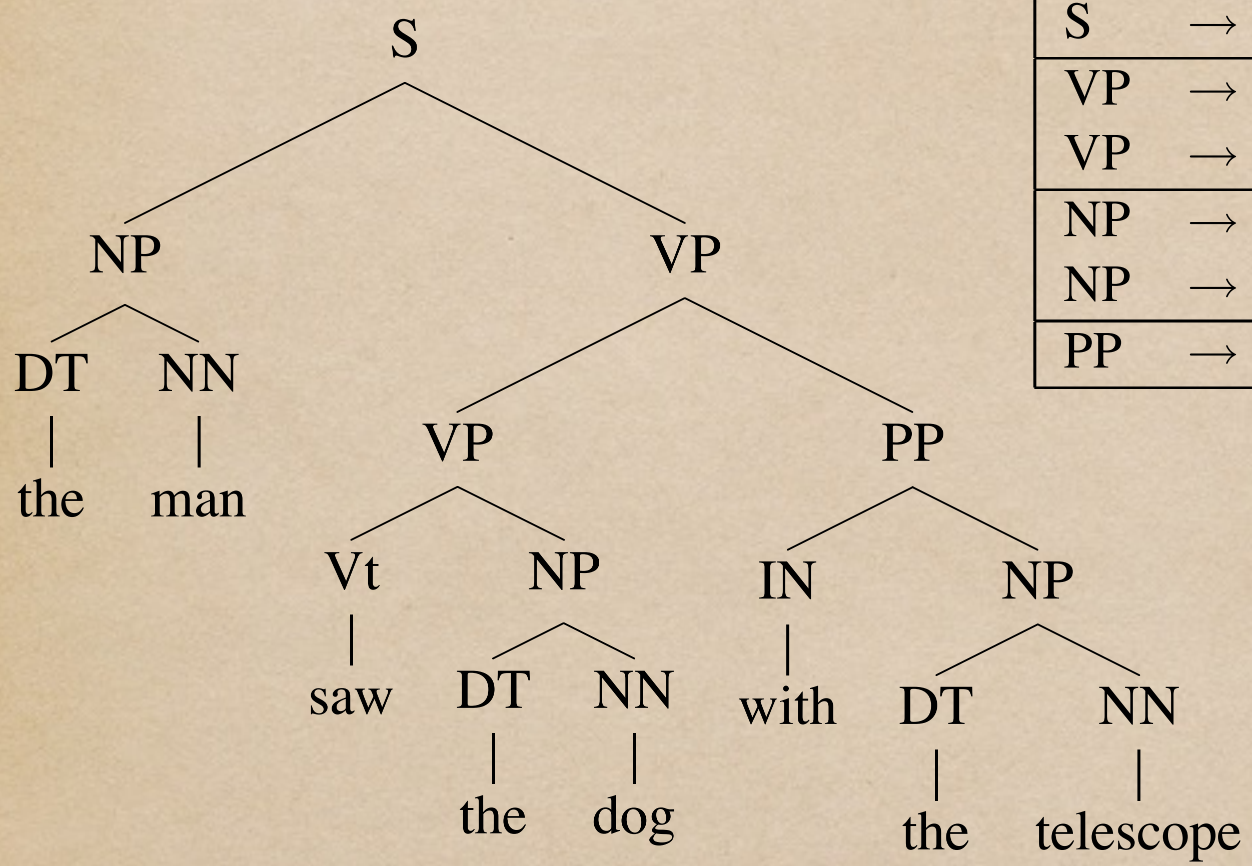
S	→	NP	VP	1.0
VP	→	Vt	NP	0.2
VP	→	VP	PP	0.7
NP	→	DT	NN	0.8
NP	→	NP	PP	0.2
PP	→	IN	NP	1.0

Vi	→	sleeps	1.0
Vt	→	saw	1.0
NN	→	man	0.1
NN	→	woman	0.1
NN	→	telescope	0.3
NN	→	dog	0.5
DT	→	the	1.0
IN	→	with	0.6
IN	→	in	0.4

$$P(T,S) = 1\times0.8\times1\times1\times0.2\times1\times0.2\times0.8\times1\times1\times0.5\times0.6\times0.8\times1\times0.3 = 0.0018$$

Avoid the temptation of multiplying the probability decorating each branch.

Example

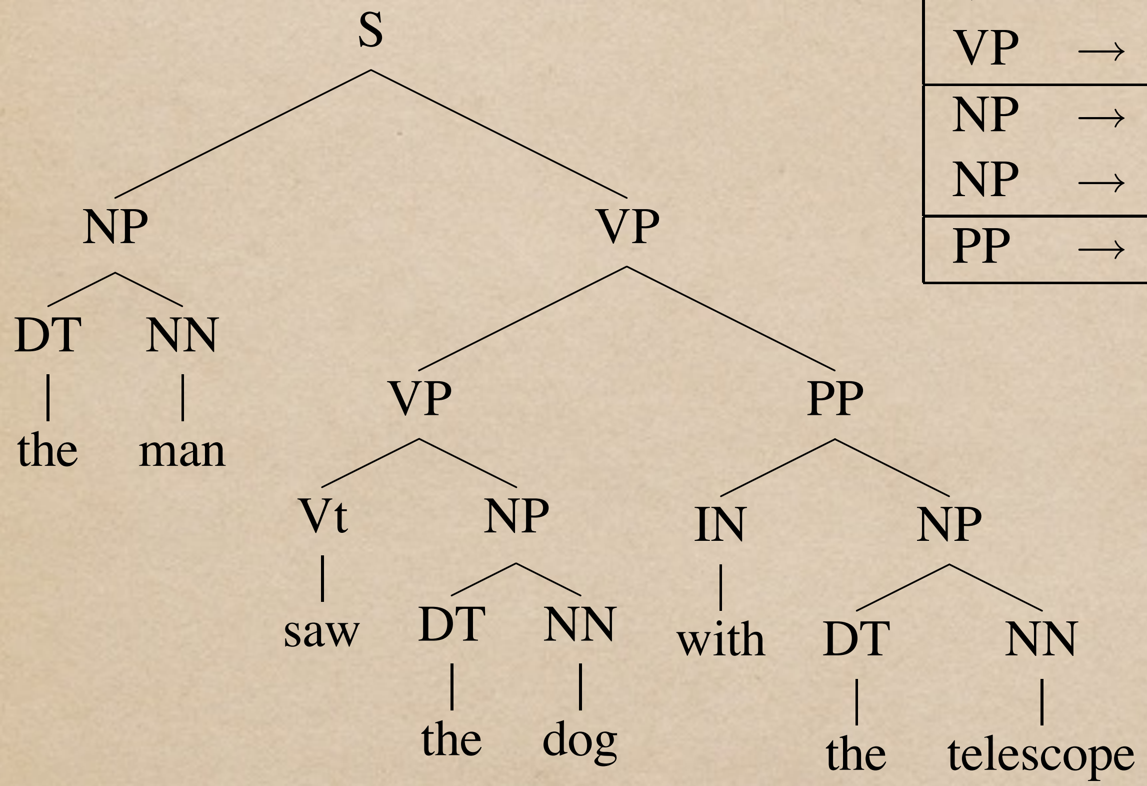


S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vt	NP	0.2
VP	$\rightarrow$	VP	PP	0.7
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4

$$P(T, S) = 0.8 \times 0.7 \times 0.2 \times 0.8 \times 0.5 \times 0.6 \times 0.8 \times 0.3 = 0.0064$$

Example



S	→	NP	VP	1.0
VP	→	Vt	NP	0.8
VP	→	VP	PP	0.2
NP	→	DT	NN	0.8
NP	→	NP	PP	0.2
PP	→	IN	NP	1.0

Vi	→	sleeps	1.0
Vt	→	saw	1.0
NN	→	man	0.1
NN	→	woman	0.1
NN	→	telescope	0.3
NN	→	dog	0.5
DT	→	the	1.0
IN	→	with	0.6
IN	→	in	0.4

$$P(T,S) = 1\times0.8\times1\times1\times0.8\times1\times0.2\times0.8\times1\times1\times0.5\times0.6\times0.8\times1\times0.3 = 0.0073$$

Learning the probabilities

1- Use a treebank:
The probability of of each expansion of a non-terminal is obtained by counting the number of times that expansion occurs in the corpus, then normalise:

$$P(\alpha \to \beta|\alpha) = \frac{Count(\alpha \to \beta)}{\Sigma_y Count(\alpha \to y)} = \frac{Count(\alpha \to \beta)}{\Sigma_y Count(\alpha)}$$

Learning the probabilities

2- Parse your non-annotated corpus.

- For non-ambiguous sentences: increment a counter for every rule in the parse, then normalise.
- For ambiguous sentences: work incrementally
  - assign an equal prob. to rules of the parser
  - parse a sentence and compute a prob for it.
  - use these to weight the counts and re-estimate the probs, until convergence.

  - Inside-Out algorithm: a generalisation of our hybrid topdown-bottomup parsing algorithm and a special case of Expectation Maximisation.

Parsing Algorithm: CKY

Step (1): Turn your grammar into Chomsky normal form:
each rules must expand to either of the following:

- two non-terminals
- a single terminal

$$A \rightarrow B\ C \qquad A \rightarrow a$$

$$A, B, C, \in N \qquad a \in \Sigma$$

Nothing is lost:

Thm. a CFG and its ChNF accept the same language.

Since the tree is now binary, it can be denoted by a
2 dimensional matrix.
More specifically, the upper triangle of an (n+1)x(n+1) matrix.

Parsing Algorithm: CKY

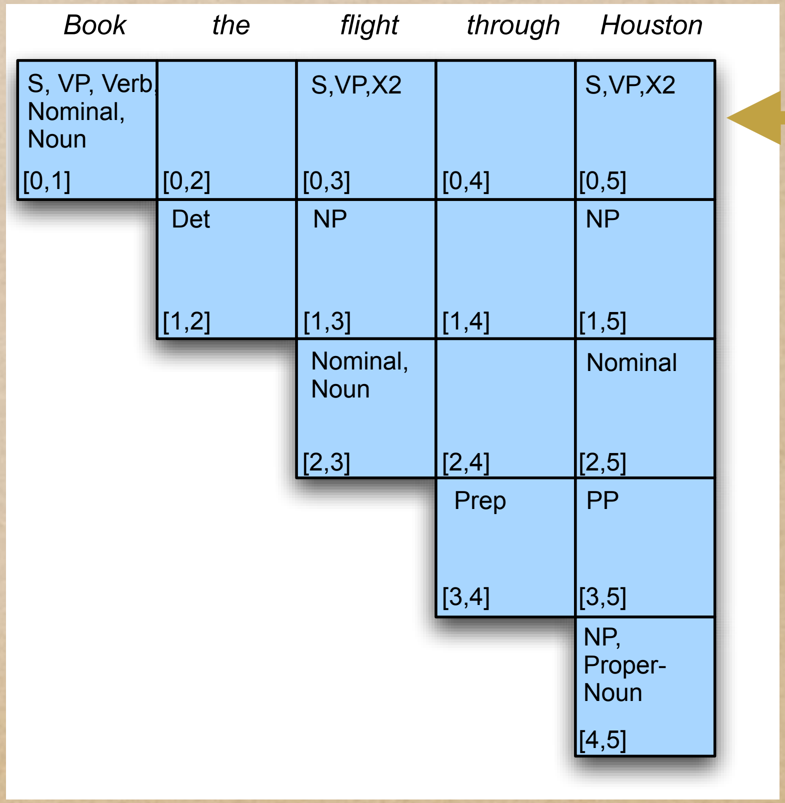
Secondary Steps: Fill in that matrix.

1- Index your input sentence by inserting a number before and after each word:

0 Book 1 that 2 flight 3 through 4 Houston 5

2- Each cell (i,j) of matrix can be filled in with a set of non terminals representing all constituents from position i to j.

The cell representing the entire input is [0,n].

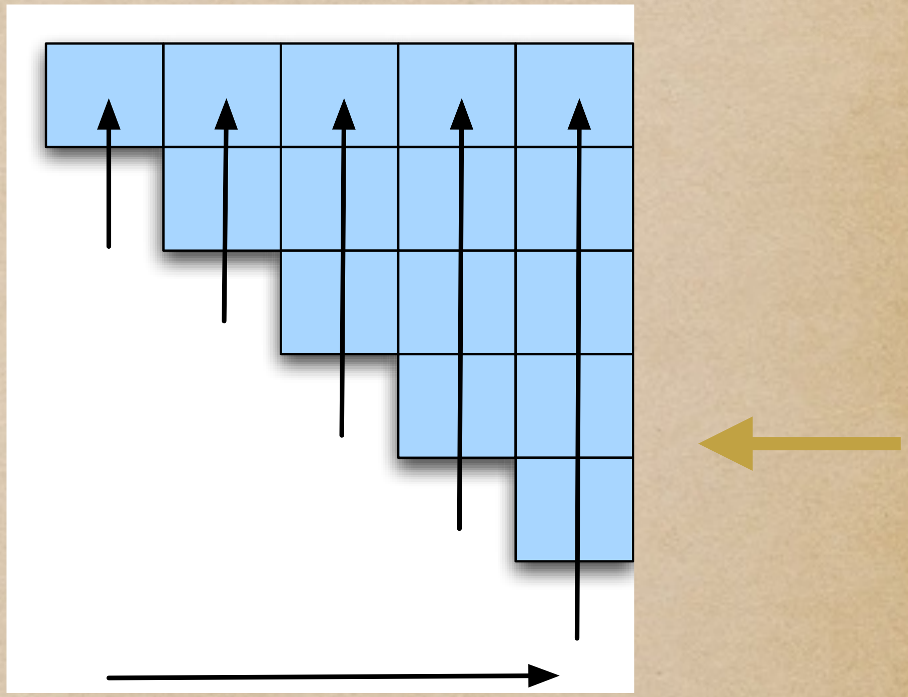


Book	the	flight	through	Houston
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper-Noun [4,5]

Parsing Algorithm: CKY

The matrix is filled in a bottom-up fashion:

- fill the matrix a column at a time from the left.
- fill each column from bottom to top.



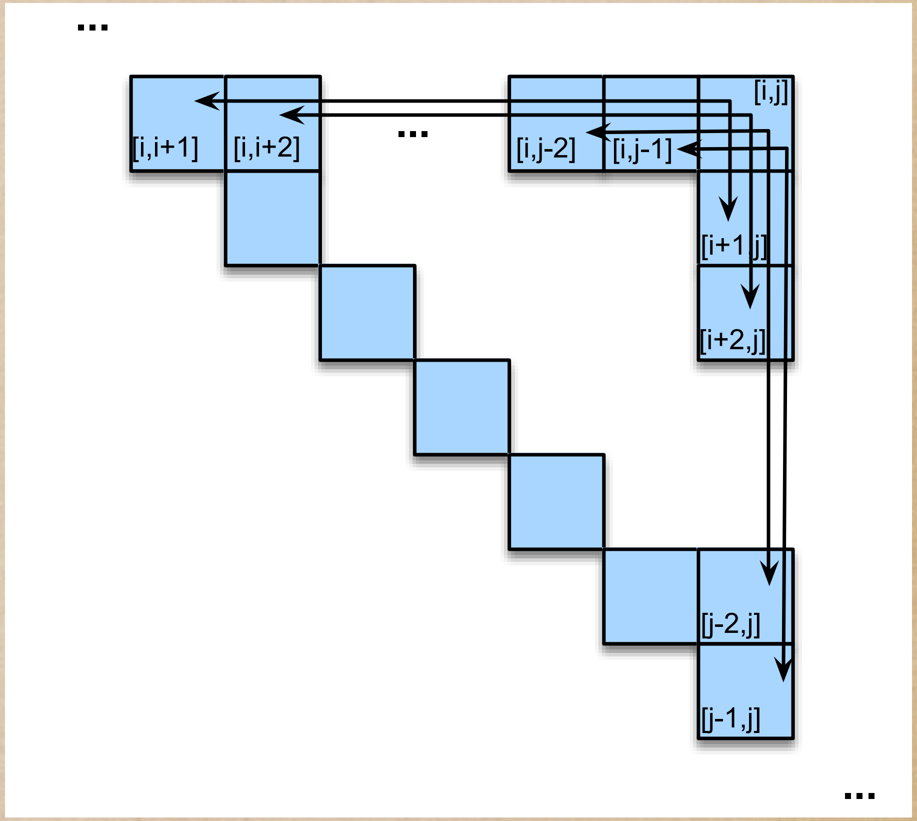
Parsing Algorithm: CKY

Last Step: Find an S cell in the table.

In the presence of ambiguity, we want to return all possible parses of a sentence.

- make each non-terminal point to cells from which it was derived.

Now all parses are returned by choosing an S from cell [0,n] and retrieving all of its component constituents from the matrix, by following the pointers.



Probabilistic CKY

Assign a 3rd dimension to each cell of matrix.
This dimension corresponds to each non-terminals that can be put in the original cell.
The value of the 3rd dimension cells contain a probability of that non-terminal.

So we have an $(n+1) \times (n+1) \times k$ matrix.

A cell $[i,j,A]$ therein is the probability of the constituent A that is between positions i to j of the input.

Example

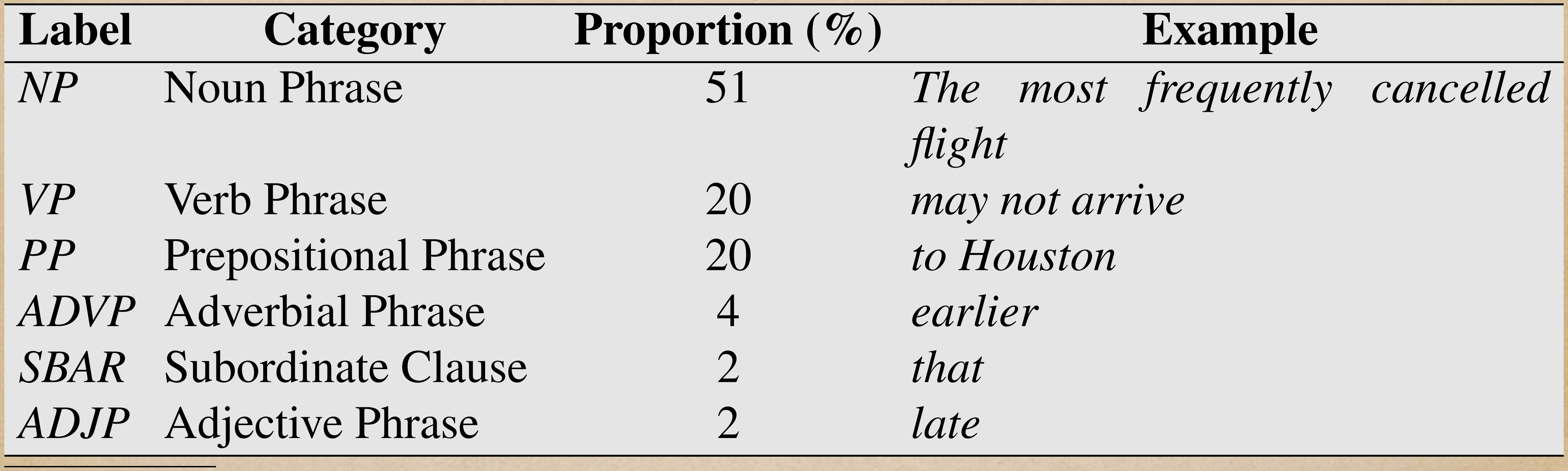
PCFG

S-> NP VP	0.8	Det -> the	0.4
NP -> Det	0.3	Det -> a	0.4
VP -> V NP	0.2	N -> meal	0.01
V ->	0.05	N -> flight	0.02

The	flight	includes	a	meal
Det .4 [0,1]	NP .3*.4 * .2			
	N .2 [1,2]			
		V 0.05 [2,3]		

First steps of a PCKY parse.

Some real statistics



Label	**Category**	**Proportion (%)**	**Example**
NP	Noun Phrase	51	*The most frequently cancelled flight*
VP	Verb Phrase	20	*may not arrive*
PP	Prepositional Phrase	20	*to Houston*
ADVP	Adverbial Phrase	4	*earlier*
SBAR	Subordinate Clause	2	*that*
ADJP	Adjective Phrase	2	*late*

Most frequent major categories of Pen treebank, according to a CONLL shared task.

Problems with PCFGs

1- Independence Assumption

PCFG's assume rules are applied independent of each other:

"**the probability of a group of independent events is their multiplication**."

Not correct,
as shown by this real statistics:

	pronoun	non pronoun
Subject	91%	9%
Object	34%%	66%%

Solution:

Conditionalise the probabilities on whether the non-terminal is in subject or object position.

Technique: **parent annotation**

Problems with PCFGs

2- Lack of sensitivity to lexical facts

Recall the problem of the ambiguity caused by PP attachment. This problem can be solved if we had more information about the word the pp was modifying. For example:

in "**workers dump sacks into bins**", the PP "**into bins**" is modifying the VP; the expansion rule is:

VP->V NP PP

in "**fishermen caught tons of herring**", the PP "**of herring**" is modifying the NP "**herring**", the rules to use are:

VP->V NP, NP -> NP PP

Solution: lexicalized parsers: Collins 1999, Charniak 1997.

Evaluating Parsers

Parse Eval:

A metric that measures how much the constituents in the hypothesis parse tree look like the constituents in a gold standard parse tree.

Gold standards: hand annotated corpora, like Pen tree.

$$\text{Recall} = \frac{\text{No. correct constituents in the hypothesis parse of s}}{\text{No. constituents in the gold standard parse of s}}$$

$$\text{Precision} = \frac{\text{No. correct constituents in the hypothesis parse of s}}{\text{No. of total constituents in the hypothesis parse of s}}$$

Mixture of the two: F-Score

Human Parsing

How do people deal with ambiguity? Do they use probabilities?

Some evidence for:

Word probability: if the compound of words is less probable, humans' reading time increases:

More time: One way to **avoid confusion** is to make changes …
Less time: One way to **avoid discovery** is to make changes …

Parse probability: if the grammatical structure is less probable, humans' reading time increases:

garden-path sentences:

The complex houses married and single students.

People first parse "complex houses" as an adj-noun phrase.

They then have to go back to correct, some times not managing.

The End of Syntax!

See you in two weeks for :

Week 11: Logical and distributional semantics.

Week 12: Neural Nets.