جامعة المنوفية

MENOUFIA UNIVERSITY
FACULTY OF COMPUTERS AND INFORMATION

Fourth Year (Second Semester)
CS Dept., (CS 436)

Natural Language Processing NLP

Lecture Eight

Dr. Hamdy M. Mousa

PROBABILISTIC MODELS OF PRONUNCIATION AND SPELLING

Language Modeling

- We can model the word prediction task as the ability to assess the **conditional probability** of a word given the previous words in the sequence

P(wn|w1, w2 … wn-1)

- We'll call a statistical model that can assess this a Language Model

Language Modeling

- How might we go about calculating such a conditional probability?
  - One way is to use the definition of conditional probabilities and look for counts. So to get
  - **P(the | its water is so transparent that)**
- By definition that's

$$\frac{P(\text{its water is so transparent that the})}{P(\text{its water is so transparent that})}$$

- We can get each of those from counts in a large corpus.

Easy Estimate

- How to estimate?

  P(the | its water is so transparent that)

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{\text{Count(its water is so transparent that the)}}{\text{Count(its water is so transparent that)}}$$

The Chain Rule

$$
\begin{aligned}
P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_1^{k-1})
\end{aligned}
$$

P(its water was so transparent)=

P(its)*

P(water | its)*

P(was | its water)*

P(so | its water was)*

P(transparent | its water was so)

Language Modeling

- Unfortunately, for most sequences and for most text collections we won’t get good estimates from this method.
- What we’re likely to get is **0. Or worse 0/0**.
- Clearly, we’ll have to be a little more clever.
- Let’s use
  - the chain rule of probability
  - a particularly useful independence assumption.

Independence Assumption

- This particular kind of independence assumption is called a **Markov assumption**
- So for each component in the product replace with the approximation (assuming a prefix of N)

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

- Bigram version

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$$

Problem

- Let's assume we're using N-grams
- How can we assign a probability to a sequence where one of the component n-grams has a value of zero
- Assume all the words are known and have been seen
  - Go to a lower order n-gram
  - Back off from bigrams to unigrams
  - Replace the zero with something else

Example

Training set:

… denied the allegations

… denied the reports

… denied the claims

… denied the request

Test set:

… denied the offer

… denied the loan

P(offer | denied the) = 0

Add-One (Laplace)

- Make the zero counts 1.
- Rationale: They're just events you haven't seen yet.
- If you had seen them, chances are you would only have seen them once… so make the count equal to 1.

Add-one Smoothing

- For unigrams:
  - Add 1 to every word (type) count
  - Normalize by N (tokens) /(N (tokens) +V (types))
  - Smoothed count (adjusted for additions to N) is:

$$\left(c_i + 1\right)\frac{N}{N+V}$$

  - Normalize by N to get the new unigram probability:

$$p_i^* = \frac{c_i + 1}{N+V}$$

- For bigrams:
  - Add 1 to every bigram $c(w_{n-1}\ w_n) + 1$
  - Increase unigram count by vocabulary size $c(w_{n-1}) + V$

BERP Bigram Counts

Berkeley Restaurant Corpus

	I	Want	To	Eat	Chinese	Food	lunch
I	8	1087	0	13	0	0	0
Want	3	0	786	0	6	8	6
To	3	0	10	860	3	0	12
Eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
Food	19	0	17	0	0	0	0
Lunch	4	0	0	0	0	1	0

Bigram Probabilities: Use Unigram Count

- Normalization: divide bigram count by unigram count of first word.

I	Want	To	Eat	Chinese	Food	Lunch
3437	1215	3256	938	213	1506	459

- Computing the probability of I I
  - P(I|I) = C(I I)/C(I) = 8 / 3437 = .0023
- A bigram grammar is an VxV matrix of probabilities, where **V** is the vocabulary size

Learning a Bigram Grammar

- The formula

  $P(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$

  is used for bigram “parameter estimation”

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

Training and Testing

- Probabilities come from a training corpus, which is used to design the model.
  - overly narrow corpus: probabilities don't generalize
  - overly general corpus: probabilities don't reflect task or domain
- A separate test corpus is used to evaluate the model, typically using standard metrics
  - held out test set
  - cross validation
  - evaluation differences should be statistically significant

Add-one Smoothing

Add 1 to every N-gram count

- $P(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$

- $P(w_n|w_{n-1}) = [C(w_{n-1}w_n) + 1] / [C(w_{n-1}) + V]$

Add-one Smoothed Bigrams

Assume a vocabulary V=1500

$$P(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$$

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

$$P'(w_n|w_{n-1}) = [C(w_{n-1}w_n)+1]/[C(w_{n-1})+V]$$

	I	want	to	eat	Chinese	food	lunch
I	9	1088	1	14	1	1	1
want	4	1	787	1	7	9	7
to	4	1	11	861	4	1	13
eat	1	1	3	1	20	3	53
Chinese	3	1	1	1	1	121	2
food	20	1	18	1	1	1	1
lunch	5	1	1	1	1	2	1

	I	want	to	eat	Chinese	food	lunch
I	.0018	.22	.00020	.0028	.00020	.00020	.00020
want	.0014	.00035	.28	.00035	.0025	.0032	.0025
to	.00082	.00021	.0023	.18	.00082	.00021	.0027
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032
lunch	.0024	.00048	.00048	.00048	.00048	.00096	.00048

Smoothing Changes

- To reconstruct the count matrix so we can see how much a smoothing algorithm has changed the original counts.

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

Smoothing Changes

– Discount: ratio of new counts to old (e.g. add-one smoothing changes the BERP count (to|want) from 786 to 331 (d_c=.42) and p(to|want) from .65 to .28)
  - Add-one smoothing thinks we are extremely likely to see unseen word, rather than words we've seen.

– Problem: add one smoothing changes counts drastically:
  - too much weight given to unseen n-grams
  - in practice, unsmoothed bigrams often work better!

Other smoothing methods

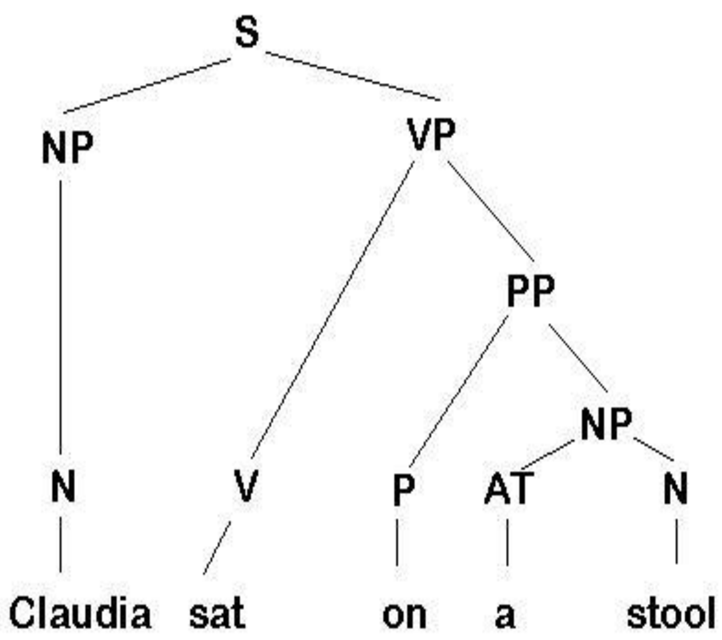
- Witten-Bell
- Good-Turing Discounting
- Backoff methods

PROBABILISTIC CONTEXT-FREE GRAMMARS

CFG

Parsing

- bottom up where you start with the input sentence and try to reach the start symbol
- Top down, **you** start with the starting symbol and try to reach the input sentence by applying the appropriate rules. Left recursion is a problem. (A → Aa)



Advantage bottom up:

“What is the weather forecast for this afternoon?”

A lot of parsing algorithms available from computer science

Problem: people don’t follow the rules of grammar strictly, especially in spoken language. Creating a grammar that covers all this constructions is unfeasible.

Probabilistic CFG

The weighted finite state automaton/transducer (or probabilistic FSA/FST)

A mixture between formal language and probabilistic models is the PCFG

If there are m rules for left-hand side non terminal node

$$\boxed{A: A \rightarrow \alpha_1, A \rightarrow \alpha_2 ..., A \rightarrow \alpha_m}$$

Then probability of these rules is

$$\boxed{P(A \rightarrow \alpha_j \mid G) = C(A \rightarrow \alpha_j) / \sum_{i=1}^{m} C(A \rightarrow \alpha_i)}$$

Where C denotes the number of times each rule is used.

PCFGs

- PCFG = CFG + rule probabilities
  - Help solve ambiguity
  - <u>Definition:</u>
    - N (S, VP, NP, …)
    - T (words)
    - R (set of rules of type A → β)
    - S (start symbol)
    - P (probabilities, one for each rule)
- Should sum to 1 for each element of N
- <u>Example:</u>
  - P(S → NP VP | S) = 0.8
  - P(S → VP | S) = 0.2

PCFGs

- The probability of some parse of a sentence is then the **product of the probabilities of all the rules** used.
- The probability of a sentence **s**, e.g. any particular string of words, is the sum of the probabilities of all its parses $t_1, t_2, \ldots, t_n$.

$$p(s) = \sum_j p(t_j) p(s|t_j)$$

PCFGs

- We want the parse tree T which is most likely given the sentence S.

$$\hat{T}(S) = \underset{T \in \tau(S)}{\operatorname{argmax}} \frac{P(T,S)}{P(S)}$$

- P(S) will be a constant for each tree,

$$\hat{T}(S) = \underset{T \in \tau(S)}{\operatorname{argmax}} P(T,S)$$

- So, P(T, S) = P(T),

$$\hat{T}(S) = \underset{T \in \tau(S)}{\operatorname{argmax}} P(T)$$

PCFGs

- The probability of an **unambiguous sentence** is P(T, S) = P(T) or just the probability of the single parse tree for that sentence.
- The probability of an **ambiguous sentence** is the sum of the probabilities of all the parse trees for the sentence:

$$
\begin{aligned}
P(S) &= \sum_{T \in \tau(S)} P(T,S) \\
&= \sum_{T \in \tau(S)} P(T)
\end{aligned}
$$

- An additional useful feature of PCFGs for language modeling is that they can assign a probability to **substrings of a sentence**.

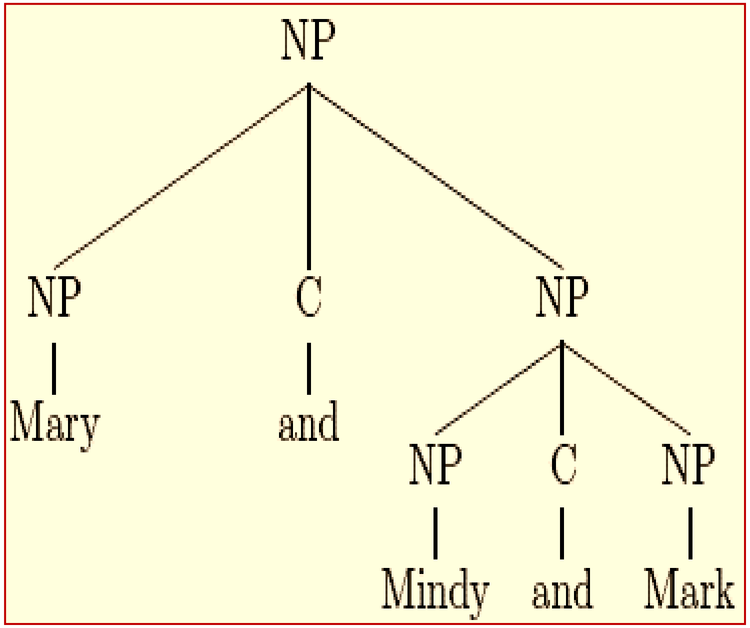
PCFGs

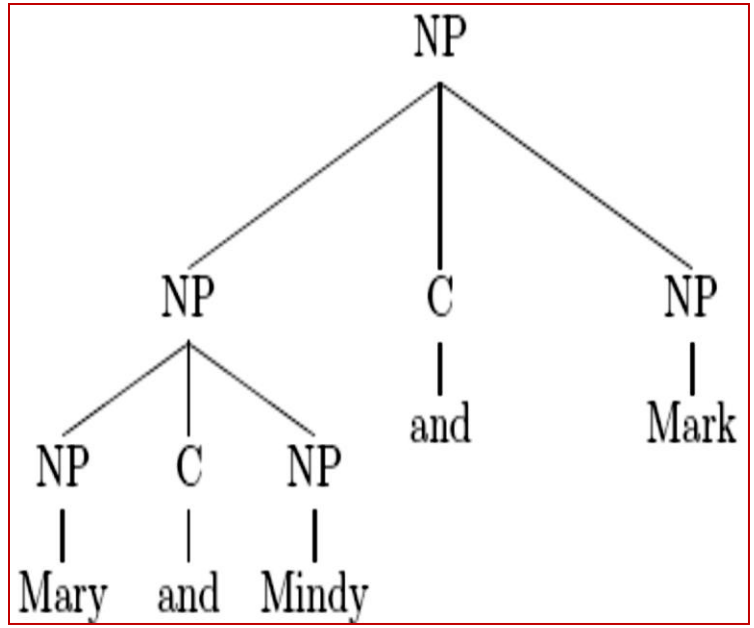
- In the case at hand, there are no structural ambiguities; there is only one possible structure for any acceptable sentence.
- Let's consider example, but one where there are structural ambiguities.

$$
\begin{array}{rcll}
NP & \rightarrow & NP\ C\ NP & .4 \\
NP & \rightarrow & Mary & .3 \\
NP & \rightarrow & Mindy & .2 \\
NP & \rightarrow & Mark & .1 \\
C & \rightarrow & and & 1
\end{array}
$$

<u>Example:</u> **Mary and Mindy and Mark.**

PCFGs





NP	$\rightarrow$	$NP\ C\ NP$	.4
NP	$\rightarrow$	$Mary$	.3
NP	$\rightarrow$	$Mindy$	.2
NP	$\rightarrow$	$Mark$	.1
C	$\rightarrow$	and	1

- Mary and Mindy and Mark. The ambiguity surrounds whether the first two conjuncts are grouped together or the last two. The same rules are used in each parse, so the probability of either one of them is: .3 × .2 ×.1 × 1 × 1 × .4 × .4 = .00096. The overall probability of the string is then .00096 + .00096 = .00192.

Probability of a parse tree in a PCFG

$$P(tree) = \prod P(rule_i)$$

- The more rules are involved, the *less probable a parse*

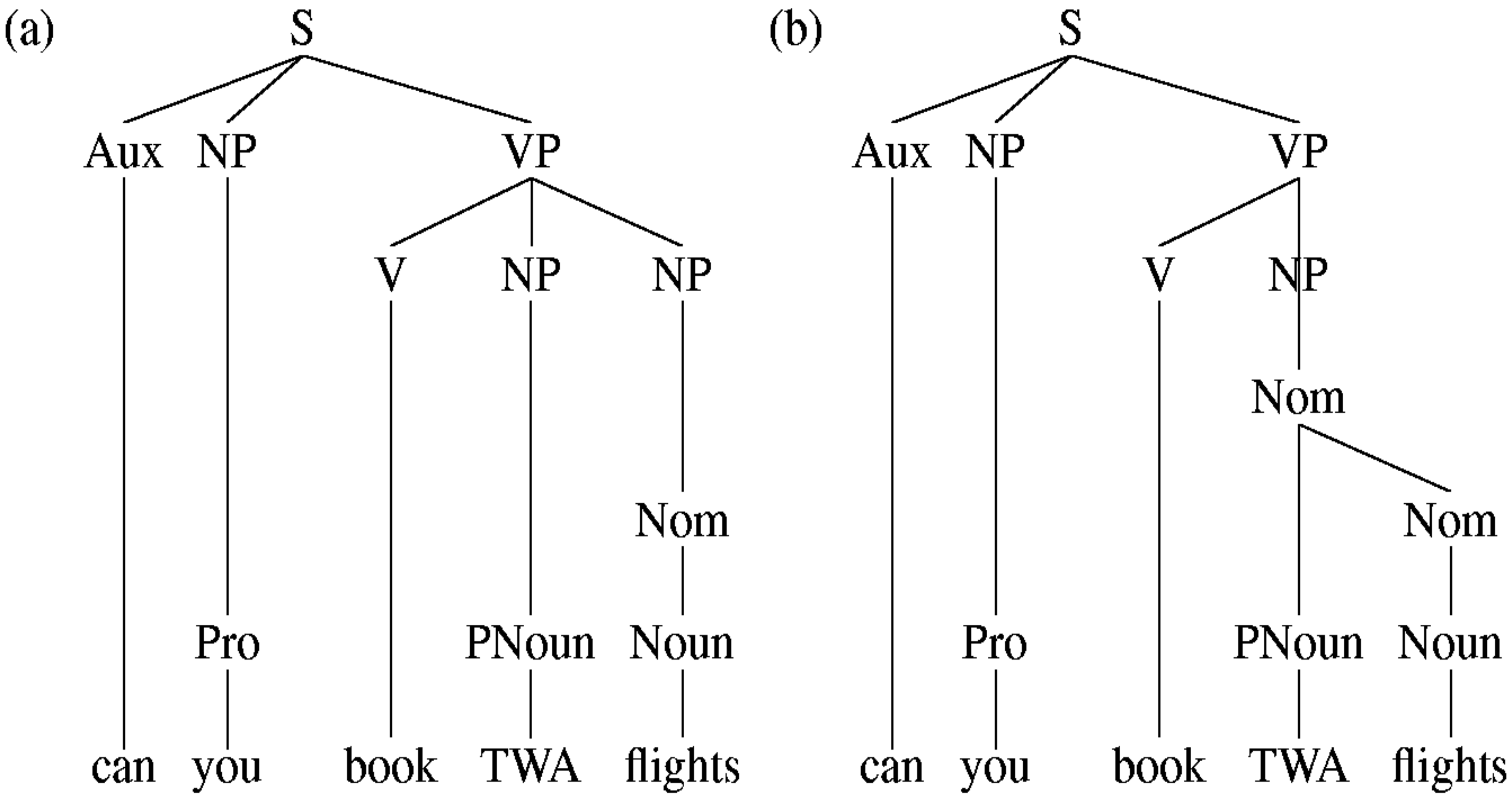
$$Parse(sentence) = \arg\max_i P(tree_i)$$

PCFG as a language model

$$P(\text{parse}) = \prod P(\text{rule}_i)$$
$$P(\text{sentence}) = \Sigma\, P(\text{parse}_k)$$
$$P(\text{text}) = \Sigma\, P(\text{parse}_l)$$

- **Problems:**
  - disadvantage for embedded structures purely syntactic difference
  - context-free: node expansion is local decision inside the tree
- **solution: Lexicalized Parsing**

PCFG


(a)
S
Aux NP VP
V NP NP
Nom
Pro PNoun Noun
can you book TWA flights
(b)
S
Aux NP VP
V NP
Nom
Nom
Pro PNoun Noun
can you book TWA flights

PCFG

Rules			P	Rules			P
S	$\rightarrow$	Aux NP VP	.15	S	$\rightarrow$	Aux NP VP	.15
NP	$\rightarrow$	Pro	.40	NP	$\rightarrow$	Pro	.40
VP	$\rightarrow$	V NP NP	.05	VP	$\rightarrow$	V NP	.40
NP	$\rightarrow$	Nom	.05	NP	$\rightarrow$	Nom	.05
NP	$\rightarrow$	PNoun	.35	Nom	$\rightarrow$	PNoun Nom	.05
Nom	$\rightarrow$	Noun	.75	Nom	$\rightarrow$	Noun	.75
Aux	$\rightarrow$	Can	.40	Aux	$\rightarrow$	Can	.40
NP	$\rightarrow$	Pro	.40	NP	$\rightarrow$	Pro	.40
Pro	$\rightarrow$	you	.40	Pro	$\rightarrow$	you	.40
Verb	$\rightarrow$	book	.30	Verb	$\rightarrow$	book	.30
PNoun	$\rightarrow$	TWA	.40	Pnoun	$\rightarrow$	TWA	.40
Noun	$\rightarrow$	flights	.50	Noun	$\rightarrow$	flights	.50

$$\begin{aligned} P(T_l) &= .15 * .40 * .05 * .05 * .35 * .75 * .40 * .40 * .40 \\ &\quad * .30 * .40 * .50 \\ &= 1.5 \times 10^{-6} \end{aligned}$$

$$\begin{aligned} P(T_r) &= .15 * .40 * .40 * .05 * .05 * .75 * .40 * .40 * .40 \\ &\quad * .30 * .40 * .50 \\ &= 1.7 \times 10^{-6} \end{aligned}$$

PCFG Probability

- We can see that the right tree has a higher probability.
- Thus this parse would correctly be chosen by a disambiguation algorithm which selects the parse with the highest PCFG probability.

- Two parse trees for an ambiguous sentence. **The meaning**
  - Parse (a): 'Can you book flights on behalf of TWA?',
  - parse (b): 'Can you book flights which are run by TWA'.

Parsing as Dynamic Programming

- Given a problem, systematically fill a table of solutions to sub-problems: this is called memorization.
  - Once solutions to all sub-problems have been accumulated, solve the overall problem by composing them.
  - For parsing, the sub-problems are analyses of sub-strings and correspond to constituents that have been found.
  - Sub-trees are stored in a chart (aka well-formed substring table), which is a record of all the substructures that have ever been built during the parse.
- **Solves re-parsing problem:** sub-trees are looked up, not re-parsed-->
- **Solves ambiguity problem:** chart implicitly stores all parses-->

CYK Algorithm

- CYK (Cocke, Younger, Kasami) is an algorithm for recognizing and recording constituents in the chart.
- Assumes that the grammar is in Chomsky Normal Form:
  - if it is ε-free & rules all have form A → BC or A → w.
- Conversion to CNF can be done automatically.

1. INF VP → to VP	⇛	1. INF VP → TO VP TO → to
2. NP → Pronoun		2. NP → I | he| she
3. S → Aux NP VP		3. S → X1 VP X1 → Aux NP

Probabilistic CYK Algorithm

- The parsing problem for PCFGs is to produce the most-likely parse for a given sentence, i.e. to compute

$$\hat{T}(S) = \operatorname*{argmax}_{T \in \tau(S)} P(T)$$

The Probabilistic CYK algorithm

function CKY-PARSE(*words*, *grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      $table[i,j] \leftarrow table[i,j] \cup$
$$\{A \mid A \rightarrow BC \in grammar, \; B \in table[i,k], \; C \in table[k,j]\}$$

- Fill the upper-triangular matrix a column at a time, working from left to right.
- Fill each column from bottom to top.

Probabilistic CKY

- CKY can be modified for PCFG parsing by including in each cell a probability for each non-terminal.

- Cell[*i*,*j*] must retain the *most probable* derivation of each constituent (non-terminal) covering words *i* +1 through *j* together with its associated probability.

- When transforming the grammar to CNF, must set production probabilities to preserve the probability of derivations.

Construct a Triangular Table

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
w_1	w_2	w_3	w_4	w_5

w1	w2	w3	w4	w5
$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	X1, 5
	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$X_{2,5}$
		$X_{3,3}$	$X_{3,4}$	$X_{3,5}$
			$X_{4,4}$	$X_{4,5}$
				$X_{5,5}$

Table for string ‘**w**’ that has length 5

Example CYK Algorithm

- Show the CYK Algorithm with the following example:
  - CNF grammar **G**
    - S → AB | BC
    - A → BA | a
    - B → CC | b
    - C → AB | a
  - **w** is baaba
  - Question Is **baaba** in L(G)?

Constructing The Triangular Table

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

{B}	**{A, C}**	**{A, C}**	**{B}**	**{A, C}**
b	**a**	**a**	**b**	**a**

Calculating the Bottom ROW

Constructing The Triangular Table

- $\mathbf{X_{1,2} = (X_{i,i}, X_{i+1,j}) = (X_{1,1}, X_{2,2})}$
- → {B}{A,C} = {BA, BC}
- Steps:
  - Look for production rules to generate BA or BC
  - There are two: S and A
  - $\mathbf{X_{1,2} = \{S, A\}}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

S $\rightarrow$ AB | BC
A $\rightarrow$ BA | a
B $\rightarrow$ CC | b
C $\rightarrow$ AB | a

{S, A}				
{B}	{A, C}	{A, C}	{B}	{A, C}
b	**a**	**a**	**b**	**a**

Constructing The Triangular Table

- $\mathbf{X_{2,3} = (X_{i,i}, X_{i+1,j}) = (X_{2,2}, X_{3,3})}$
- $\rightarrow$ {A, C}{A,C} = {AA, AC, CA, CC}
- Steps:
  - Look for production rules to generate {AA, AC, CA, CC}
  - There is one: B
  - $\mathbf{X_{2,3} = \{B\}}$

S $\rightarrow$ AB | BC
A $\rightarrow$ BA | a
B $\rightarrow$ CC | b
C $\rightarrow$ AB | a

Constructing The Triangular Table

- S → AB | BC
- A → BA | a
- B → CC | b
- C → AB | a

{S, A}	{B}			
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Constructing The Triangular Table

- $\mathbf{X_{3,4} = (X_{i,i}, X_{i+1,j}) = (X_{3,3}\ , X_{4,4})}$
- ➔ {A, C}{B} = {AB, CB}
- Steps:
  - Look for production rules to generate {AB, CB}
  - There are two: S and C
  - $\mathbf{X_{3,4} = \{S, C\}}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

- S → AB | BC
- A → BA | a
- B → CC | b
- C → AB | a

{S, A}	**{B}**	**{S, C}**		
{B}	**{A, C}**	**{A, C}**	**{B}**	**{A, C}**
b	**a**	**a**	**b**	**a**

Constructing The Triangular Table

- $\mathbf{X_{4,5} = (X_{i,i}, X_{i+1,j}) = (X_{4,4}\ , X_{5,5})}$
- $\rightarrow$ {B}{A, C} = {BA, BC}
- Steps:
  - Look for production rules to generate {BA, BC}
  - There are two: S and A
  - $\mathbf{X_{4,5} = \{S, A\}}$

S $\rightarrow$ AB | BC
A $\rightarrow$ BA | a
B $\rightarrow$ CC | b
C $\rightarrow$ AB | a

Constructing The Triangular Table

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Constructing The Triangular Table

- $\mathbf{X_{1,3} = (X_{i,i}, X_{i+1,j})\ (X_{i,i+1}, X_{i+2,j})}$
  $\mathbf{= (X_{1,1}, X_{2,3}), (X_{1,2}, X_{3,3})}$
- ➔ {B}{B} **U** {S, A}{A, C}= {BB, SA, SC, AA, AC}
- Steps:
  - Look for production rules to generate {BB, SA, SC, AA, AC}
  - There are NONE: S and A
  - $\mathbf{X_{1,3} = \emptyset}$
  - **no elements in this set (empty set)**

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Ø				
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

Constructing The Triangular Table

- $\mathbf{X_{2,4} = (X_{i,i}, X_{i+1,j})\ (X_{i,i+1}, X_{i+2,j})}$
  $\mathbf{= (X_{2,2}, X_{3,4}),\ (X_{2,3}, X_{4,4})}$
- ➔ {A, C}{S, C} **U** {B}{B}= {AS, AC, CS, CC, BB}

Steps:

- Look for production rules to generate {AS, AC, CS, CC, BB}
- There is one: B
- $\mathbf{X_{2,4} = \{B\}}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Ø	{B}			
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	**a**	**a**	**b**	**a**

Constructing The Triangular Table

- $\mathbf{X_{3,5} = (X_{i,i}, X_{i+1,j})\ (X_{i,i+1}, X_{i+2,j})}$
  $\mathbf{= (X_{3,3}, X_{4,5}),\ (X_{3,4}, X_{5,5})}$
- → {A,C}{S,A} **U** {S,C}{A,C}
  = {AS, AA, CS, CA, SA, SC, CA, CC}

Steps:

- Look for production rules to generate {AS, AA, CS, CA, SA, SC, CA, CC}
- There is one: B
- $\mathbf{X_{3,5} = \{B\}}$

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Constructing The Triangular Table

Ø	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	**a**	**a**	**b**	**a**

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

Final Triangular Table

{S, A, C}	← $X_{1, 5}$			
Ø	{S, A, C}			
Ø	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	**a**	**a**	**b**	**a**

5	SAC	SAC	B	SA	AC
4	–	B	SC	B	
3	–	B	AC		
2	SA	AC			
1	B				
	1	2	3	4	5

- Table for string ‘**w**’ that has length 5
- The algorithm populates the triangular table

CYK Algorithm for Parsing CFG

IDEA: For each substring of a given input *x*, find all variables which can derive the substring.

- Once these have been found, telling which variables generate *x* becomes a simple matter of looking at the grammar, since it's in Chomsky normal form

CYK Example

- S → NP VP
- VP → V NP
- NP → NP PP
- VP → VP PP
- PP → P NP
- NP → Ahmad | Ali | Hail
- V → called
- P → from

Example: Ahmad called Ali from Hail

CYK Example

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

$_0$ **Ahmad** $_1$ **called** $_2$ **Ali** $_3$ **from** $_4$ **Hail** $_5$

end at / start at	1:	2:	3:	4:	5:
0:	Ahmad	Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		called	called Ali	called Ali from	called Ali from Hail
2:			Ali	Ali from	Ali from Hail
3:				from	From Hail
4:					Hail

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP
NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ **Ahmad** $_1$ **called** $_2$ **Ali** $_3$ **from** $_4$ **Hail** $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

0 Ahmad 1 called 2 Ali 3 from 4 Hail 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

0 **Ahmad** 1 **called** 2 **Ali** 3 **from** 4 **Hail** 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	Ali from Hail
3:				**P** (From)	From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

$_0$ **Ahmad** $_1$ **called** $_2$ **Ali** $_3$ **from** $_4$ **Hail** $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

0 **Ahmad** 1 **called** 2 **Ali** 3 **from** 4 **Hail** 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X** Ahmad called	**S** Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

0 Ahmad 1 called 2 Ali 3 from 4 Hail 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X**	**S** Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	**X** called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

0 **Ahmad** 1 **called** 2 **Ali** 3 **from** 4 **Hail** 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP
NP → Ahmad | Ali | Hail V → called P → from

0 **Ahmad** 1 **called** 2 **Ali** 3 **from** 4 **Hail** 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	**X**	**S** Ahmad called Ali	**X** Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	**X** called Ali from	called Ali from Hail
2:			**NP** (Ali)	**X** Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP　VP → V NP　NP → NP PP　VP → VP PP　PP → P NP

NP → Ahmad | Ali | Hail　V → called　P → from

$_0$ **Ahmad** $_1$ **called** $_2$ **Ali** $_3$ **from** $_4$ **Hail** $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP_1** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

$_0$ **Ahmad** $_1$ **called** $_2$ **Ali** $_3$ **from** $_4$ **Hail** $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	$\mathbf{VP_2}$ $\mathbf{VP_1}$ called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

0 Ahmad 1 called 2 Ali 3 from 4 Hail 5

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	**S** Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP_2** **VP_1** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP VP → V NP NP → NP PP VP → VP PP PP → P NP

NP → Ahmad | Ali | Hail V → called P → from

$_0$ **Ahmad** $_1$ **called** $_2$ **Ali** $_3$ **from** $_4$ **Hail** $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	**S$_1$** Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	**VP$_2$** **VP$_1$** called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP
NP → Ahmad | Ali | Hail   V → called   P → from

$_0$ Ahmad $_1$ called $_2$ Ali $_3$ from $_4$ Hail $_5$

end at / start at	1:	2:	3:	4:	5:
0:	**NP** (Ahmad)	X	**S** Ahmad called Ali	X Ahmad called Ali from	S_1 S_2 Ahmad called Ali from Hail
1:		**V** (Called)	**VP** called Ali	X called Ali from	VP_2 VP_1 called Ali from Hail
2:			**NP** (Ali)	X Ali from	**NP** Ali from Hail
3:				**P** (From)	**PP** From Hail
4:					**NP** (Hail)

S → NP VP   VP → V NP   NP → NP PP   VP → VP PP   PP → P NP

NP → Ahmad | Ali | Hail   V → called   P → from

CYK Example

S → NP VP

VP → V NP

NP → NP PP

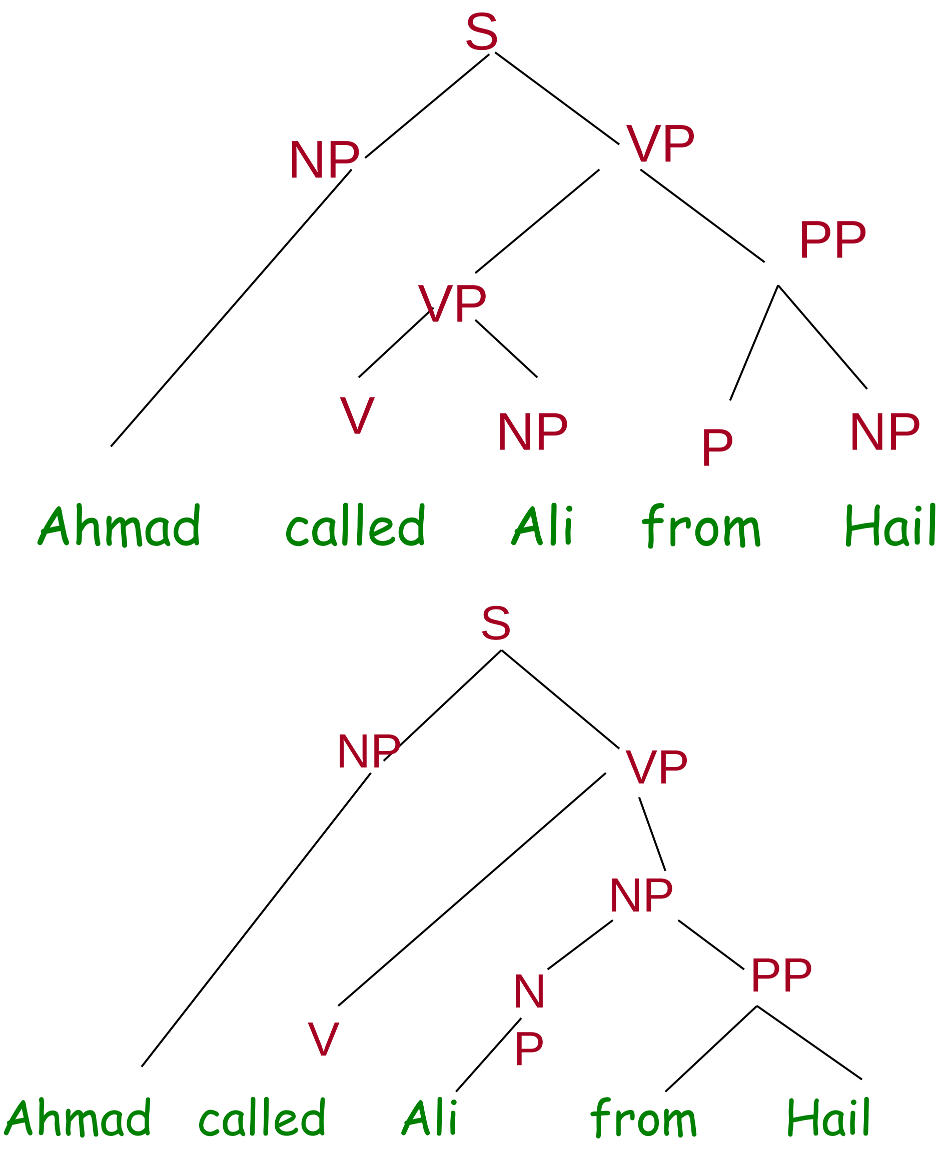
VP → VP PP

PP → P NP

NP → Ahmad | Ali | Hail

V → called

P → from



CYK Example

Same Example: We might see it in different format

				NP
			P	Hail
		NP	from	
	V	Ali		
NP	called			
Ahmad				

S → NP VP

VP → V NP

NP → NP PP

VP → VP PP

PP → P NP

NP → Ahmad | Ali | Hail

V → called

P → from

CYK Example

S_1 S_2	VP_1 VP_2	NP	PP	NP
X	X	X	P	Hail
S	VP	NP	from	
X	V	Ali		
NP	called			
Ahmad				

Simple PCFG

Grammar	Prob	
S → NP VP	0.8	
S → Aux NP VP	0.1	+ 1.0
S → VP	0.1	
NP → Pronoun	0.2	
NP → Proper-Noun	0.2	+ 1.0
NP → Det Nominal	0.6	
Nominal → Noun	0.3	
Nominal → Nominal Noun	0.2	+ 1.0
Nominal → Nominal PP	0.5	
VP → Verb	0.2	
VP → Verb NP	0.5	+ 1.0
VP → VP PP	0.3	
PP → Prep NP	1.0	

Lexicon

Det → the | a | that | this
0.6 0.2 0.1 0.1

Noun → book | flight | meal | money
0.1 0.5 0.2 0.2

Verb → book | include | prefer
0.5 0.2 0.3

Pronoun → I | he | she | me
0.5 0.1 0.1 0.3

Proper-Noun → Houston | NWA
0.8 0.2

Aux → does
1.0

Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2

Probabilistic Grammar Conversion

Original Grammar

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Chomsky Normal Form

S → NP VP 0.8
S → X1 VP 0.1
X1 → Aux NP 1.0
S → book | include | prefer
0.01 0.004 0.006
S → Verb NP 0.05
S → VP PP 0.03
NP → I | he | she | me
0.1 0.02 0.02 0.06
NP → Houston | NWA
0.16 .04
NP → Det Nominal 0.6
Nominal → book | flight | meal | money
0.03 0.15 0.06 0.06
Nominal → Nominal Noun 0.2
Nominal → Nominal PP 0.5
VP → book | include | prefer
0.1 0.04 0.06
VP → Verb NP 0.5
VP → VP PP 0.3
PP → Prep NP 1.0
Det → the | a | that | this
0.6 0.2 0.1 0.1
Noun → book | flight | meal | money
0.1 0.5 0.2 0.2
Prep → from | to | on | near | through
0.25 0.25 0.1 0.2 0.2
Proper-Noun → Houston | NWA
0.8 0.2

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None			
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

S → NP VP 0.8
S → Aux NP VP 0.1
S → VP 0.1
NP → Pronoun 0.2
NP → Proper-Noun 0.2
NP → Det Nominal 0.6
Nominal → Noun 0.3
Nominal → Nominal Noun 0.2
Nominal → Nominal PP 0.5
VP → Verb 0.2
VP → Verb NP 0.5
VP → VP PP 0.3
PP → Prep NP 1.0

S → NP VP 0.8
S → X1 VP 0.1
X1 → Aux NP 1.0
S → book | include | prefer
0.01 0.004 0.006
S → Verb NP 0.05
S → VP PP 0.03
NP → I | he | she | me
0.1 0.02 0.02 0.06
NP → Houston | NWA
0.16 .04
NP → Det Nominal 0.6
Nominal → book | flight | meal | money
0.03 0.15 0.06 0.06
Nominal → Nominal Noun 0.2
Nominal → Nominal PP 0.5
VP → book | include | prefer
0.1 0.04 0.06
VP → Verb NP 0.5
VP → VP PP 0.3
PP → Prep NP 1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135		
	Det:.6	NP:.6*.6*.15 =.054		
		Nominal:.15 Noun:.5		

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S $\rightarrow$ NP VP	0.8
S $\rightarrow$ Aux NP VP	0.1
S $\rightarrow$ VP	0.1
NP $\rightarrow$ Pronoun	0.2
NP $\rightarrow$ Proper-Noun	0.2
NP $\rightarrow$ Det Nominal	0.6
Nominal $\rightarrow$ Noun	0.3
Nominal $\rightarrow$ Nominal Noun	0.2
Nominal $\rightarrow$ Nominal PP	0.5
VP $\rightarrow$ Verb	0.2
VP $\rightarrow$ Verb NP	0.5
VP $\rightarrow$ VP PP	0.3
PP $\rightarrow$ Prep NP	1.0

S $\rightarrow$ NP VP	0.8
S $\rightarrow$ X1 VP	0.1
X1 $\rightarrow$ Aux NP	1.0
S $\rightarrow$ book | include | prefer 0.01 0.004 0.006	
S $\rightarrow$ Verb NP	0.05
S $\rightarrow$ VP PP	0.03
NP $\rightarrow$ I | he | she | me 0.1 0.02 0.02 0.06	
NP $\rightarrow$ Houston | NWA 0.16 .04	
NP $\rightarrow$ Det Nominal	0.6
Nominal $\rightarrow$ book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal $\rightarrow$ Nominal Noun	0.2
Nominal $\rightarrow$ Nominal PP	0.5
VP $\rightarrow$ book | include | prefer 0.1 0.04 0.06	
VP $\rightarrow$ Verb NP	0.5
VP $\rightarrow$ VP PP	0.3
PP $\rightarrow$ Prep NP	1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.05*.5* .000864 =.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP 0.8
S → Aux NP VP 0.1
S → VP 0.1
NP → Pronoun 0.2
NP → Proper-Noun 0.2
NP → Det Nominal 0.6
Nominal → Noun 0.3
Nominal → Nominal Noun 0.2
Nominal → Nominal PP 0.5
VP → Verb 0.2
VP → Verb NP 0.5
VP → VP PP 0.3
PP → Prep NP 1.0

S → NP VP 0.8
S → X1 VP 0.1
X1 → Aux NP 1.0
S → book | include | prefer
0.01 0.004 0.006
S → Verb NP 0.05
S → VP PP 0.03
NP → I | he | she | me
0.1 0.02 0.02 0.06
NP → Houston | NWA
0.16 .04
NP → Det Nominal 0.6
Nominal → book | flight | meal | money
0.03 0.15 0.06 0.06
Nominal → Nominal Noun 0.2
Nominal → Nominal PP 0.5
VP → book | include | prefer
0.1 0.04 0.06
VP → Verb NP 0.5
VP → VP PP 0.3
PP → Prep NP 1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.03*.0135*.032 =.00001296 S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6*.0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

S → NP VP	0.8
S → Aux NP VP	0.1
S → VP	0.1
NP → Pronoun	0.2
NP → Proper-Noun	0.2
NP → Det Nominal	0.6
Nominal → Noun	0.3
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → Verb	0.2
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

S → NP VP	0.8
S → X1 VP	0.1
X1 → Aux NP	1.0
S → book | include | prefer 0.01 0.004 0.006	
S → Verb NP	0.05
S → VP PP	0.03
NP → I | he | she | me 0.1 0.02 0.02 0.06	
NP → Houston | NWA 0.16 .04	
NP → Det Nominal	0.6
Nominal → book | flight | meal | money 0.03 0.15 0.06 0.06	
Nominal → Nominal Noun	0.2
Nominal → Nominal PP	0.5
VP → book | include | prefer 0.1 0.04 0.06	
VP → Verb NP	0.5
VP → VP PP	0.3
PP → Prep NP	1.0

Probabilistic CKY Parser

Book	the	flight	through	Houston
S :.01, VP:.1, Verb:.5 Nominal:.03 Noun:.1	None	S:.05*.5*.054 =.00135 VP:.5*.5*.054 =.0135	None	S:.0000216
	Det:.6	NP:.6*.6*.15 =.054	None	NP:.6*.6* .0024 =.000864
		Nominal:.15 Noun:.5	None	Nominal: .5*.15*.032 =.0024
			Prep:.2	PP:1.0*.2*.16 =.032
				NP:.16 PropNoun:.8

Pick most probable parse, i.e. take max to combine probabilities of multiple derivations of each constituent in each cell.