Parsing

Probabilistic Context-Free Grammars

Kurt Eberle

k.eberle@lingenio.de

(bases largely on chap. 14 of **Jurafsky/Martin 2019**)

28 Februar 2020

Outline

Definition

Probabilistic CKY Parsing of PCFGs

Problems and Improvement

Outline

Definition

Probabilistic CKY Parsing of PCFGs

Problems and Improvement

Probabilistic Context-Free Grammar - Definition

- PCFG = CFG where the rules of P obtain an additional (conditional) probability:

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$ $[p]$, where A is a non-terminal, β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$, and p is a number between 0 and 1 expressing $P(\beta \mid A)$
S	a designated **start symbol**

- $A \rightarrow \beta$ $[p]$: means $[p] = P(A \rightarrow \beta)$, where this:
  $= P(''analyzedinto''\beta \mid A)$

  where $\Sigma_\beta P(A \rightarrow \beta) = 1$
- PCFG consistent iff the sum of the probs of the sentences of L(PCFG) is 1.

Example

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow$ *that* [.10] | *a* [.30] | *the* [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow$ *book* [.10] | *flight* [.30]
$S \rightarrow VP$	[.05]	| *meal* [.05] | *money* [.05]
$NP \rightarrow Pronoun$	[.35]	| *flight* [.40] | *dinner* [.10]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow$ *book* [.30] | *include* [.30]
$NP \rightarrow Det\ Nominal$	[.20]	| *prefer* [.40]
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow$ *I* [.40] | *she* [.05]
$Nominal \rightarrow Noun$	[.75]	| *me* [.15] | *you* [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow$ *Houston* [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]	| *NWA* [.40]
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow$ *does* [.60] | *can* [.40]
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow$ *from* [.30] | *to* [.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]	| *on* [.20] | *near* [.15]
$VP \rightarrow Verb\ PP$	[.15]	| *through* [.05]
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

- [redacted]

Example

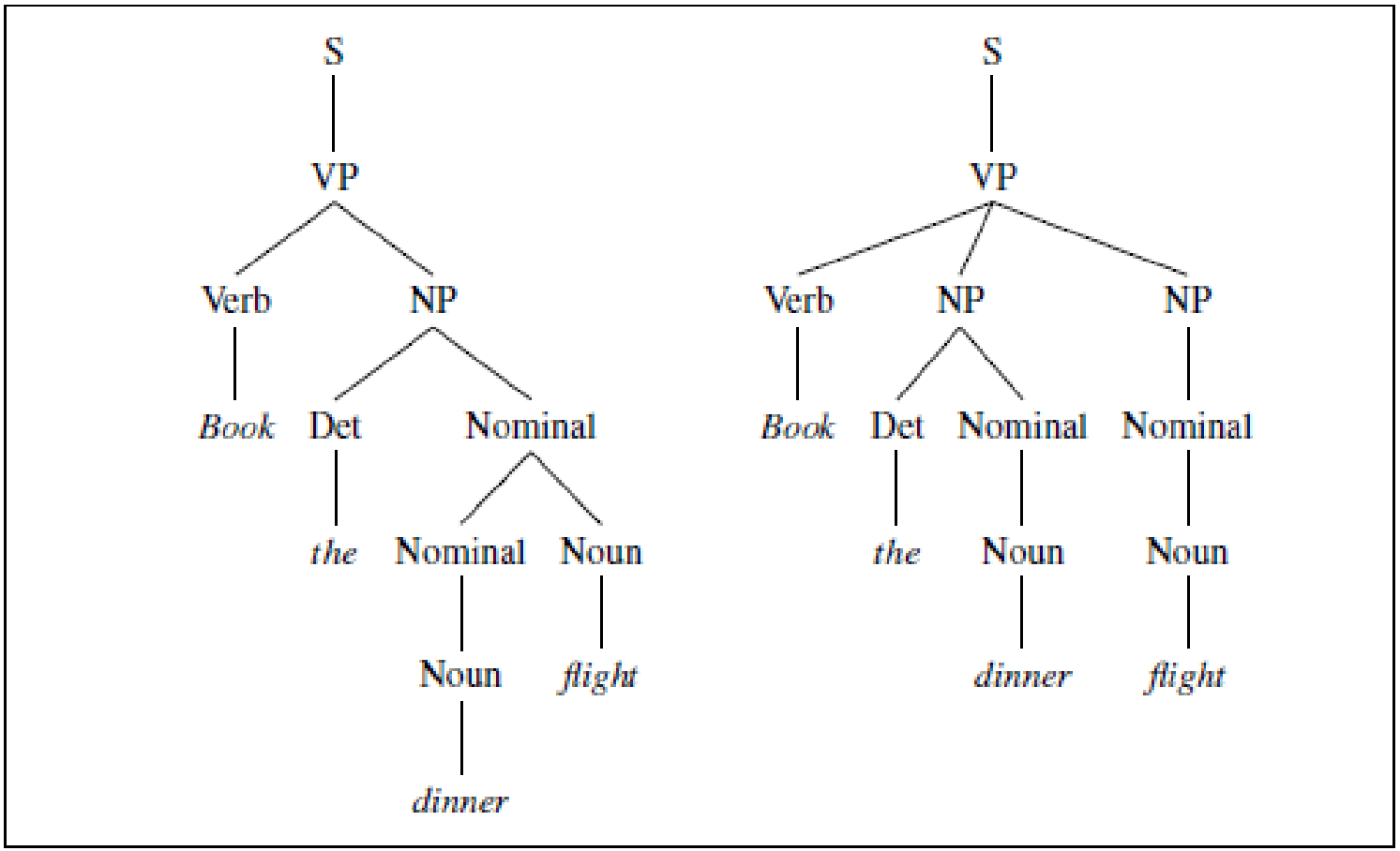
- *Book a flight that serves dinner*
- could be expressed by:

  Book the [*dinner flight*]
- according to the grammar, this string could obtain the following reading also:

  Book [*the dinner*] [*flight*]
  (structurally identical to the borderline case
  Book [*John*] [*flight*] ,
  but not possible her)

Example

- *Book the dinner flight*



Probability of trees

The probability of a particular parse T is defined as the product of the probabilities of all the n rules used to expand each of the n non-terminal nodes in the parse tree T, where each rule i can be expressed as $LHS_i \rightarrow RHS_i$:

$$P(T,S) = \prod_{i=1}^{n} P(RHS_i|LHS_i) \tag{14.2}$$

The resulting probability $P(T,S)$ is both the joint probability of the parse and the sentence and also the probability of the parse $P(T)$. How can this be true? First, by the definition of joint probability:

$$P(T,S) = P(T)P(S|T) \tag{14.3}$$

But since a parse tree includes all the words of the sentence, $P(S|T)$ is 1. Thus,

$$P(T,S) = P(T)P(S|T) = P(T) \tag{14.4}$$

Example

- *Book the dinner flight*

Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

Figure 14.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner' ".

- left tree is better (as should be).

Parsing Task

- Search for tree with highest probability

$$\hat{T}(S) = \underset{T\,s.t.\,S=\text{yield}(T)}{\text{argmax}} P(T|S) \tag{14.5}$$

By definition, the probability $P(T|S)$ can be rewritten as $P(T,S)/P(S)$, thus leading to

$$\hat{T}(S) = \underset{T\,s.t.\,S=\text{yield}(T)}{\text{argmax}} \frac{P(T,S)}{P(S)} \tag{14.6}$$

Since we are maximizing over all parse trees for the same sentence, $P(S)$ will be a constant for each tree, so we can eliminate it:

$$\hat{T}(S) = \underset{T\,s.t.\,S=\text{yield}(T)}{\text{argmax}} P(T,S) \tag{14.7}$$

Furthermore, since we showed above that $P(T,S) = P(T)$, the final equation for choosing the most likely parse neatly simplifies to choosing the parse with the highest probability:

$$\hat{T}(S) = \underset{T\,s.t.\,S=\text{yield}(T)}{\text{argmax}} P(T) \tag{14.8}$$

Outline

Definition

Probabilistic CKY Parsing of PCFGs

Problems and Improvement

Probabilistic CKY Parsing of PCFGs

-

```
function PROBABILISTIC-CKY(words, grammar) returns most probable parse
                                                and its probability
  for j ← from 1 to LENGTH(words) do
    for all { A | A → words[j] ∈ grammar }
      table[j − 1, j, A] ← P(A → words[j])
    for i ← from j − 2 downto 0 do
      for k ← i + 1 to j − 1 do
        for all { A | A → BC ∈ grammar,
                      and table[i,k,B] > 0 and table[k,j,C] > 0 }
          if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
            table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
            back[i,j,A] ← {k,B,C}
    return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

Outline

Definition

Probabilistic CKY Parsing of PCFGs

Problems and Improvement

Problems and Improvement

- Lexical dependencies and others are not taken into account

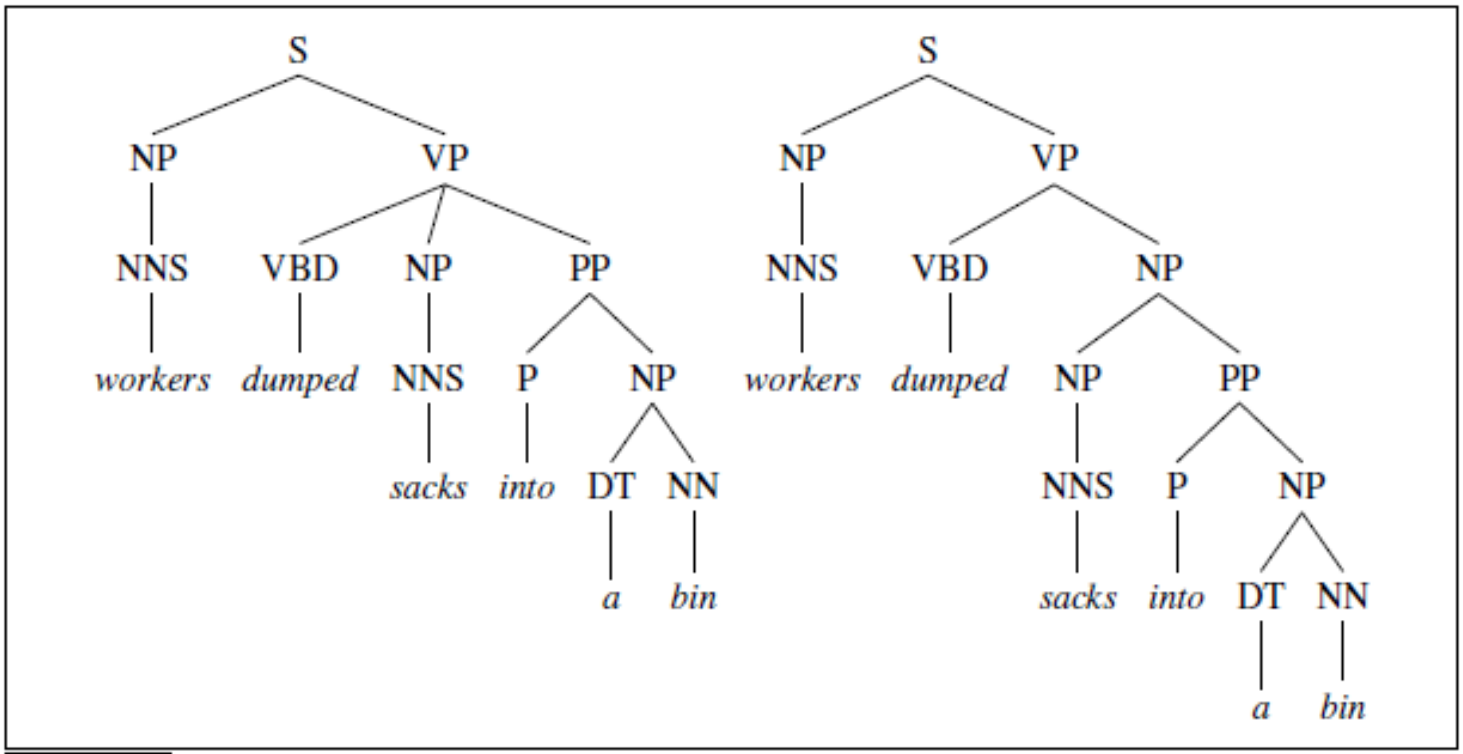


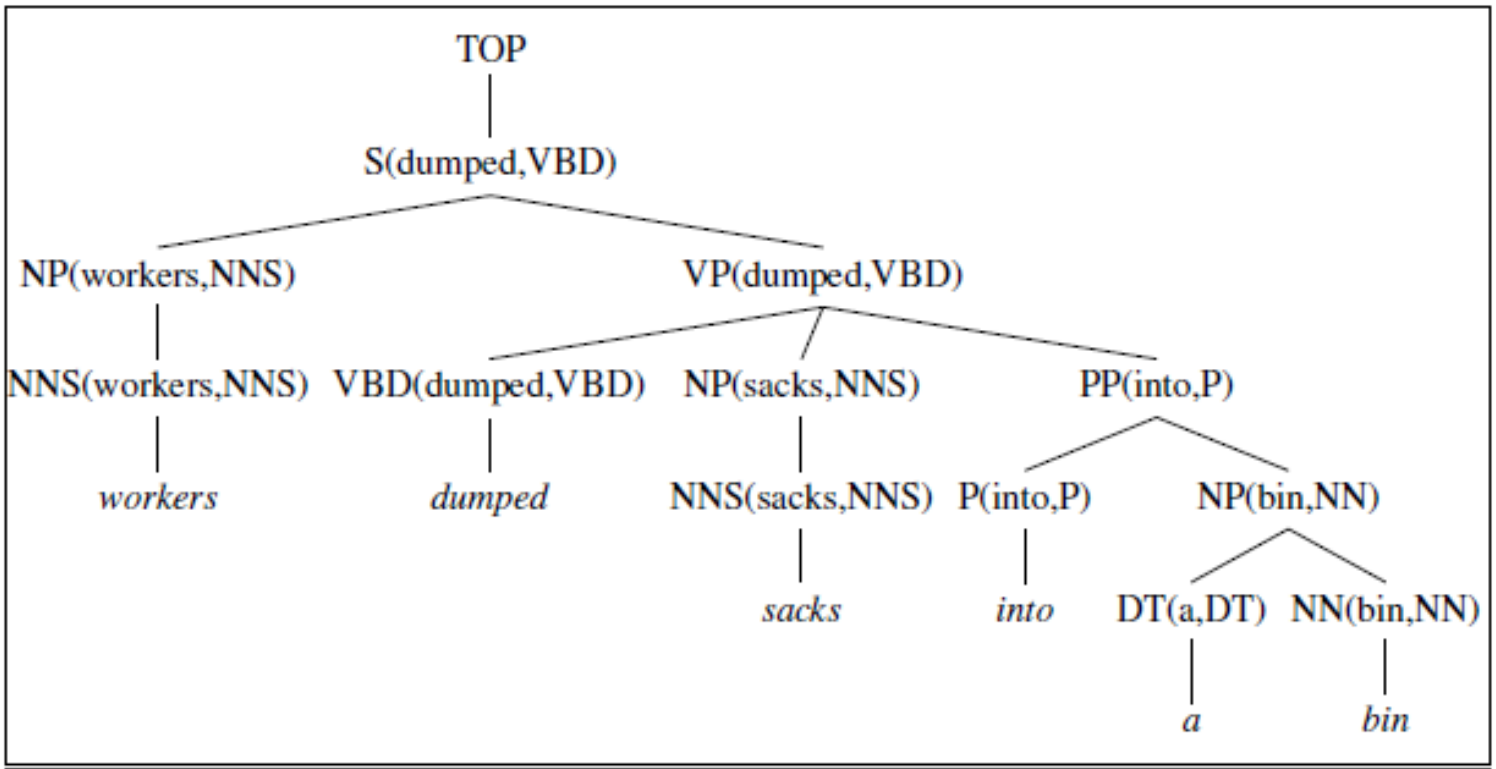
Figure 14.5 Two possible parse trees for a **prepositional phrase attachment ambiguity**. The left parse is the sensible one, in which "into a bin" describes the resulting location of the sacks. In the right incorrect parse, the sacks to be dumped are the ones which are already "into a bin", whatever that might mean.

Improvement

- Tag nodes with information about their lexical heads
- Refine constituent structure
  (several types of S, NP, etc. or simply add class information from context:
  for instance, annotate node by parent type:
  NP is in S (a subject), *NP is in VP*(a object), etc.)

$\rightarrow$ *lexicalized and unlexicalized PCFG*

Lexical context information



Internal Rules

TOP	→	S(dumped,VBD)	
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)
NP(workers,NNS)	→	NNS(workers,NNS)	
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS) PP(into,P)
PP(into,P)	→	P(into,P)	NP(bin,NN)
NP(bin,NN)	→	DT(a,DT)	NN(bin,NN)

Lexical Rules

NNS(workers,NNS)	→	workers
VBD(dumped,VBD)	→	dumped
NNS(sacks,NNS)	→	sacks
P(into,P)	→	into
DT(a,DT)	→	a
NN(bin,NN)	→	bin

Figure 14.10 A lexicalized tree, including head tags, for a WSJ sentence, adapted from Collins (1999). Below we show the PCFG rules needed for this parse tree, internal rules on the left, and lexical rules on the right.

Non-lexical context information

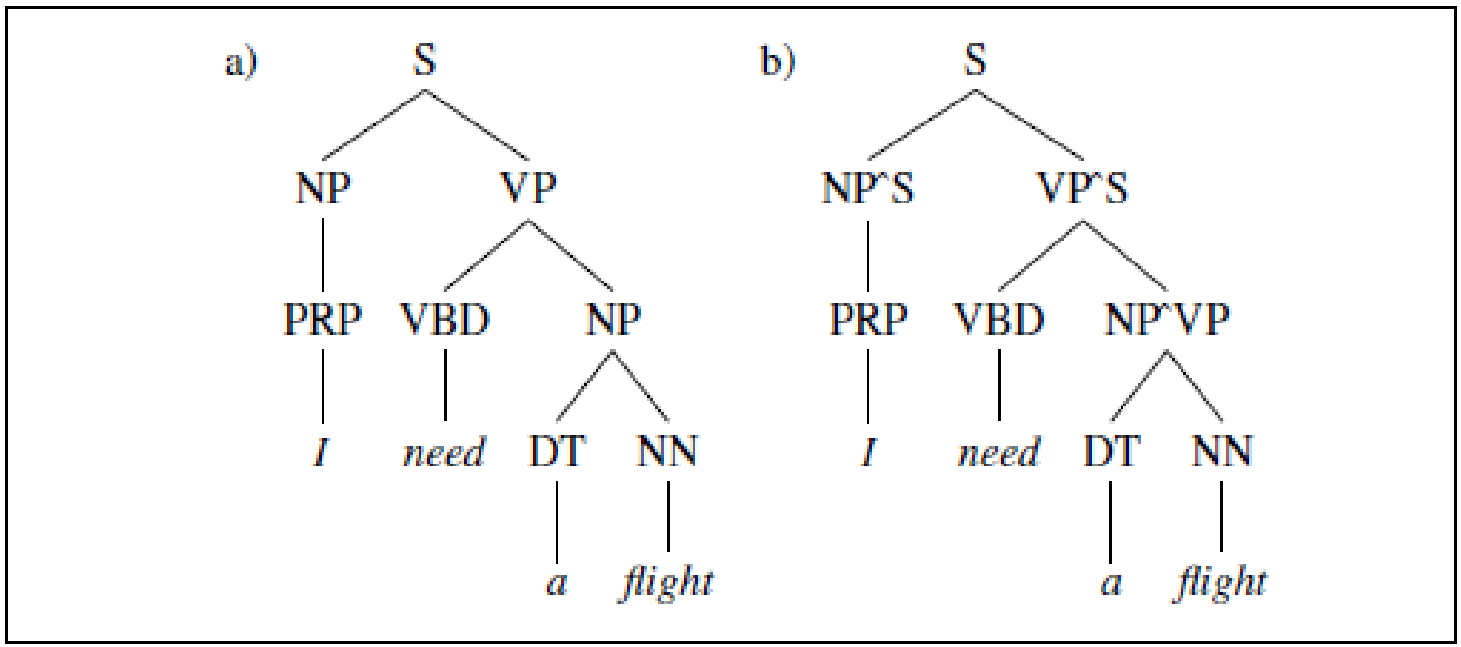


Figure 14.8 A standard PCFG parse tree (a) and one which has **parent annotation** on the nodes which aren't pre-terminal (b). All the non-terminal nodes (except the pre-terminal part-of-speech nodes) in parse (b) have been annotated with the identity of their parent.