Today: Probabilistic Parsing

Goal: Find the most likely parse.

1. Parsing with PCFGs
2. Problems
3. Probabilistic lexicalized CFGs

Slide CS474–1

CFG's

A context free grammar consists of:

1. a set of non-terminal symbols N
2. a set of terminal symbols Σ (disjoint from N)
3. a set of productions, P, each of the form $A \rightarrow \alpha$, where A is a non-terminal and α is a string of symbols from the infinite set of strings $(\Sigma \cup N)$
4. a designated start symbol S

Slide CS474–2

Probabilistic CFGs

Augments each rule in P with a conditional probability:

$A \rightarrow \beta\ [p]$

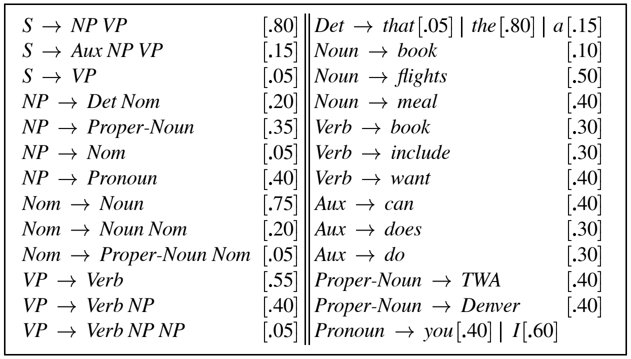
where p is the probability that the non-terminal A will be expanded to the sequence β. Often referred to as

$P(A \rightarrow \beta)$ or

$P(A \rightarrow \beta | A)$.

Slide CS474–3

Example



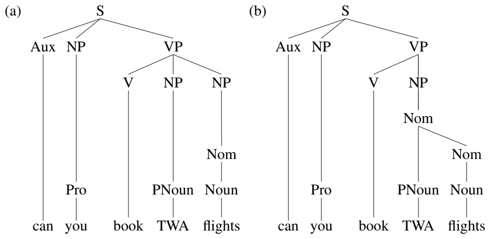
S → *NP VP*	[.80]	*Det* → *that* [.05] | *the* [.80] | *a* [.15]	
S → *Aux NP VP*	[.15]	*Noun* → *book*	[.10]
S → *VP*	[.05]	*Noun* → *flights*	[.50]
NP → *Det Nom*	[.20]	*Noun* → *meal*	[.40]
NP → *Proper-Noun*	[.35]	*Verb* → *book*	[.30]
NP → *Nom*	[.05]	*Verb* → *include*	[.30]
NP → *Pronoun*	[.40]	*Verb* → *want*	[.40]
Nom → *Noun*	[.75]	*Aux* → *can*	[.40]
Nom → *Noun Nom*	[.20]	*Aux* → *does*	[.30]
Nom → *Proper-Noun Nom*	[.05]	*Aux* → *do*	[.30]
VP → *Verb*	[.55]	*Proper-Noun* → *TWA*	[.40]
VP → *Verb NP*	[.40]	*Proper-Noun* → *Denver*	[.40]
VP → *Verb NP NP*	[.05]	*Pronoun* → *you* [.40] | *I* [.60]	

Slide CS474–4

Why are PCFGs useful?

- Assigns a probability to each parse tree T
- Useful in **disambiguation**
  - Choose the most likely parse
  - Computing the probability of a parse
    If we make independence assumptions, $\mathrm{P(T)} = \prod_{n \in T} \mathrm{p(r(n))}$.
- Useful in **language modeling** tasks

Example



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

Where does the grammar come from?

1. developed manually
2. from a **treebank**

Where do the probabilities come from?

1. from a **treebank**:

$$P(\alpha \to \beta | \alpha) = Count(\alpha \to \beta)/Count(\alpha)$$

2. use EM (forward-backward algorithm, inside-outside algorithm)

Parsing with PCFGs

Produce the most likely parse for a given sentence:

$$\hat{T}(S) = argmax_{T \in \tau(S)} P(T)$$

where $\tau(S)$ is the set of possible parse trees for S.

- Augment the Earley algorithm to compute the probability of each of its parses.

  When adding an entry E of category C to the chart using rule i with n subconstituents, $E_1, \ldots, E_n$:

  $P(E) = P(rule\ i \mid C) * P(E_1) * \ldots * P(E_n)$
- probabilistic CKY (Cocke-Kasami-Younger) algorithm

Slide CS474–9

Problems with PCFGs

Do not model *structural dependencies.*

Often the choice of how a non-terminal expands depends on the location of the node in the parse tree.

E.g. Strong tendency in English for the syntactic subject of a spoken sentence to be a pronoun.

- 91% of declarative sentences in the Switchboard corpus are pronouns (vs. lexical).
- In contrast, 34% of direct objects in Switchboard are pronouns.

Slide CS474–10

Problems with PCFGs

Do not adequately model *lexical dependencies.*

Moscow sent more than 100,000 soldiers into Afghanistan...

PP can attach to either the NP or the VP:
NP → NP PP or VP → V NP PP?

Attachment choice depends (in part) on the verb: *send* subcategorizes for a destination (e.g. expressed via a PP that begins with *into* or *to* or ...).

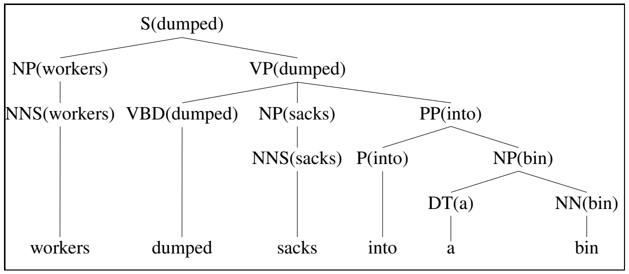
Slide CS474–11

Probabilistic lexicalized CFGs

- Each non-terminal is associated with its head.
- Each PCFG rule needs to be augmented to identify one rhs constituent to be the head daughter.
- Headword for a node in the parse tree is set to the headword of its head daughter.

Slide CS474–12

Example



Slide CS474–13

Probabilistic lexicalized CFGs

View a lexicalized (P)CFG as a simple (P)CFG with a lot more rules.

VP(dumped) → VBD(dumped) NP(sacks) PP(into) [3x10^{-10}]
VP(dumped) → VBD(dumped) NP(cats) PP(into) [8x10^{-10}]
VP(dumped) → VBD(dumped) NP(sacks) PP(above) [1x10^{-12}]
...

Problem?

Slide CS474–14

Incorporating lexical dependency information

Incorporates lexical dependency information by:

1. relating the heads of phrases to the heads of their constituents;
2. including syntactic subcategorization information.

Syntactic subcategorization dependencies:

Probability of a rule r of syntactic category n:
p(r(n) | n, h(n)).

Example: probability of expanding VP as VP → VBD NP PP will be p (r | VP, dumped).

Slide CS474–15

Incorporating lexical dependency information

Condition the probability of a node n having a head h on two factors:

1. the syntactic category of the node n
2. the head of the node's mother $h(m(n))$

p(h(n) = word_i | n, h(m(n)))

Slide CS474–16