Last Classes: Chart parsing and chunking

Today:

1. The Earley Algorithm
2. Probabilistic Parsing

Efficient Parsing

n = sentence length
Time complexity for naive algorithm: exponential in n
Time complexity for bottom-up chart parser: $\bigcirc(n^3)$

Options for improving efficiency:

1. Don't do twice what you can do once.
2. Don't represent distinctions that you don't need.

   Fall leaves fall and spring leaves spring.
3. Don't do once what you can avoid altogether.

   The can holds the water. ("can": AUX, V, N)

Earley Algorithm: Top-Down Chart Parser

For all S rules of the form $S \rightarrow X_1 \ldots X_k$, add a (top-down) edge from 1 to 1 labeled: $S \rightarrow \circ X_1 \ldots X_k$.

Do until there is no input left:

1. If the agenda is empty, look up word categories for next word, add to agenda.
2. Select a constituent from the agenda: constituent C from p_1 to p_2.
3. Using the (bottom-up) edge extension algorithm, combine C with every active edge on the chart (adding C to chart as well). Add any new constituents to the agenda.
4. For any active edges created in Step 3, add them to the chart using the top-down edge introduction algorithm.

Top-down edge introduction.

To add an edge $S \rightarrow C_1 \ldots \circ C_i \ldots C_n$ ending at position j:

For each rule in the grammar of form $C_i \rightarrow X_1 \ldots X_k$,

recursively add the new edge $C_i \rightarrow \circ X_1 \ldots X_k$ from j to j.

Grammar and Lexicon

Grammar	Lexicon
1. S → NP VP	the: ART
2. NP → ART ADJ N	large: ADJ
3. NP → ART N	can: N, AUX, V
4. NP → ADJ N	hold: N, V
5. VP → AUX VP	water: N, V
6. VP → V NP	

Sentence: $_1$ The $_2$ large $_3$ can $_4$ can $_5$ hold $_6$ water $_7$

Probabilistic Parsing

Goal: Find the most likely parse.

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

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

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

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

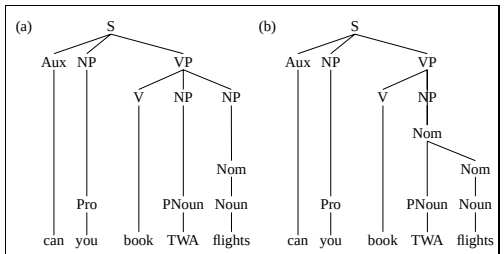
Example

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

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



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 do the probabilities come from?

1. from a **treebank**:

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

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 CYK (Cocke-Younger-Kasami) algorithm

Slide CS474–13

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 CS474–14

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 CS474–15