Parsing: Earley & PCFGs

Ling 571
Deep Processing Techniques for NLP
January 20, 2016

Roadmap

- Earley wrap-up:
  - Scanner
  - Completer
- Probabilistic Context-free Grammars (PCFGs)
  - Motivation: Ambiguity
  - Approach:
    - Definition
    - Disambiguation
    - Parsing
    - Evaluation
    - Enhancements

3 Main Sub-Routines of Earley Algorithm

- **Predictor**: Adds predictions into the chart.
- **Completer**: Moves the dot to the right when new constituents are found.
- **Scanner**: Reads the input words and enters states representing those words into the chart.

Predictor

- Intuition: create new state for top-down prediction of new phrase.
- Applied when non part-of-speech non-terminals are to the right of a dot: **S → • VP [0,0]**
- Adds new states to *current* chart
  - One new state for each expansion of the non-terminal in the grammar
    VP → • V [0,0]
    VP → • V NP [0,0]
- Formally:
  S_j: A → α · B β, [i,j]
  S_j: B → · γ, [j,j]

Chart[0]

S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor

Note that given a grammar, these entries are the same for all inputs; they can be pre-loaded.

Scanner

- Intuition: Create new states for rules matching part of speech of next word.
- Applicable when part of speech is to the right of a dot: VP → • V NP [0,0] 'Book...'
- Looks at current word in input
- If match, adds state(s) to *next* chart
  V → Book •[0,1]
- Formally:
  S_j: A → α • B β, [i,j], B in POS(words[j])
- S_{j+1}: B → words[j]•, [j,j+1]

Completer

- Intuition: parser has finished a new phrase, so must find and advance all states that were waiting for this
- Applied when dot has reached right end of rule
  NP → Det Nom • [1,3]
- Find all states w/dot at 1 and expecting an NP:
  - VP → V • NP [0,1]
- Adds new (completed) state(s) to *current* chart :
  - VP → V NP • [0,3]
- Formally: S_k: B → δ ·, [j,k]
  S_k: A → α B · β, [i,k],
  where: S_j: A → α · B β, [i,j].

Chart[1]

S12	*Verb* → *book* •	[0,1]	Scanner
S13	*VP* → *Verb* •	[0,1]	Completer
S14	*VP* → *Verb* • *NP*	[0,1]	Completer
S15	*VP* → *Verb* • *NP PP*	[0,1]	Completer
S16	*VP* → *Verb* • *PP*	[0,1]	Completer
S17	*S* → *VP* •	[0,1]	Completer
S18	*VP* → *VP* • *PP*	[0,1]	Completer
S19	*NP* → • *Pronoun*	[1,1]	Predictor
S20	*NP* → • *Proper-Noun*	[1,1]	Predictor
S21	*NP* → • *Det Nominal*	[1,1]	Predictor
S22	*PP* → • *Prep NP*	[1,1]	Predictor

Charts[2] and [3]

S23	$Det \rightarrow that \bullet$	[1,2]	Scanner
S24	$NP \rightarrow Det \bullet Nominal$	[1,2]	Completer
S25	$Nominal \rightarrow \bullet Noun$	[2,2]	Predictor
S26	$Nominal \rightarrow \bullet Nominal\ Noun$	[2,2]	Predictor
S27	$Nominal \rightarrow \bullet Nominal\ PP$	[2,2]	Predictor
S28	$Noun \rightarrow flight \bullet$	[2,3]	Scanner
S29	$Nominal \rightarrow Noun \bullet$	[2,3]	Completer
S30	$NP \rightarrow Det\ Nominal \bullet$	[1,3]	Completer
S31	$Nominal \rightarrow Nominal \bullet Noun$	[2,3]	Completer
S32	$Nominal \rightarrow Nominal \bullet PP$	[2,3]	Completer
S33	$VP \rightarrow Verb\ NP \bullet$	[0,3]	Completer
S34	$VP \rightarrow Verb\ NP \bullet PP$	[0,3]	Completer
S35	$PP \rightarrow \bullet Prep\ NP$	[3,3]	Predictor
S36	$S \rightarrow VP \bullet$	[0,3]	Completer

How do we retrieve the parses at the end?

- Augment the Completer to add pointers to prior states it advances as a field in the current state
  - i.e. what state did we advance here?
  - Read the pointers back from the final state

- What about ambiguity?
- CKY/Earley can represent it
- Can’t resolve it

Probabilistic Parsing

- Provides strategy for solving disambiguation problem
  - Compute the probability of all analyses
  - Select the most probable

- Employed in language modeling for speech recognition
  - N-gram grammars predict words, constrain search
  - Also, constrain generation, translation

PCFGs

- Probabilistic Context-free Grammars
  - Augmentation of CFGs

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**

PCFGs

- Augment each production with probability that LHS will be expanded as RHS
  - P(A→B) or P(A→B|A), P(RHS|LHS)
  - Sum over all possible expansions is 1

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

- A PCFG is consistent if sum of probabilities of all sentences in language is 1.
  - Recursive rules often yield inconsistent grammars

Example PCFG

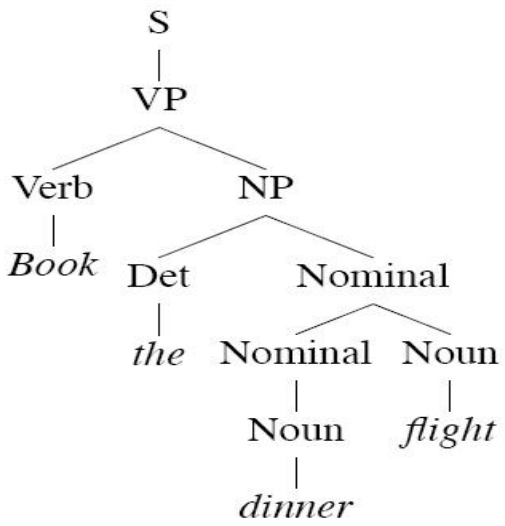
Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that\ [.10] \mid a\ [.30] \mid the\ [.60]$
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book\ [.10] \mid flight\ [.30]$
$S \rightarrow VP$	[.05]	$\mid meal\ [.15] \mid money\ [.05]$
$NP \rightarrow Pronoun$	[.35]	$\mid flights\ [.40]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

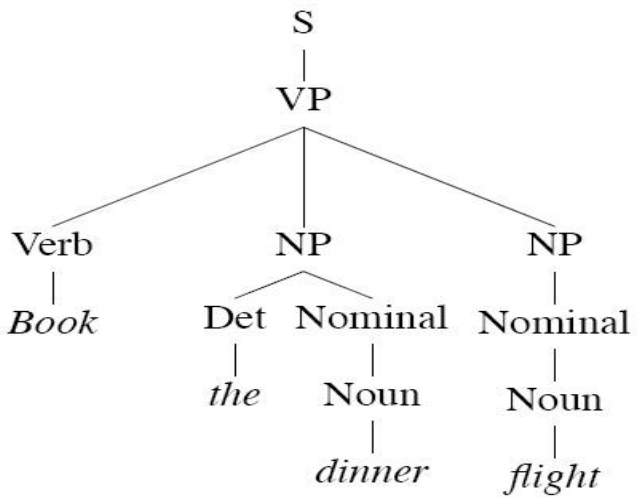
Disambiguation

- A PCFG assigns probability to each parse tree T for input S.
  - Probability of T: product of all rules to derive T

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

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





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

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

P(T,S)=0.05*0.2*0.2*0.2*0.75*
0.3*0.6*0.1*0.4=2.2x10^-6

P(T,S)=0.05*0.1*0.2*0.15*0.75*
0.75*0.3*0.6*0.1*0.4=6.1x10^-7

Parsing Problem for PCFGs

- Select T such that:

$$\hat{T}(S) = \underset{Ts.t, S=yield(T)}{\text{argmax}} \; P(T)$$

  - String of words S is *yield* of parse tree over S
  - Select tree that maximizes probability of parse

- Extend existing algorithms: CKY & Earley
  - Most modern PCFG parsers based on CKY
    - Augmented with probabilities

Probabilistic CKY

- Like regular CKY
  - Assume grammar in Chomsky Normal Form (CNF)
    - Productions:
      - A → B C or A → w
  - Represent input with indices b/t words
    - E.g., $_0$ Book $_1$ that $_2$ flight $_3$ through $_4$ Houston $_5$

- For input string length *n* and non-terminals *V*
  - Cell[i,j,A] in (n+1)x(n+1)xV matrix contains
    - Probability that constituent A spans [i,j]

Probabilistic CKY Algorithm

function PROBABILISTIC-CKY(*words*,*grammar*) **returns** most probable parse
and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{ A \mid A \rightarrow words[j] \in grammar \}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{ A \mid A \rightarrow BC \in grammar,$
        and $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
        if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
          $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
          $back[i,j,A] \leftarrow \{k, B, C\}$
  return BUILD_TREE($back$[1, LENGTH(*words*), S]), $table$[1, LENGTH(*words*), S]

PCKY Grammar Segment

- S → NP VP [0.80]
- NP → Det N [0.30]
- VP → V NP [0.20]
- V → includes [0.05]
- Det → the [0.40]
- Det → a [0.40]
- N → meal [0.01]
- N → flight [0.02]

PCKY Matrix: The flight includes a meal

Det: 0.4 **[0,1]**	**NP: 0.3*0.4*0.02 =.0024** **[0,2]**	**[0,3]**	**[0,4]**	**S: 0.8* 0.000012* 0.0024** **[0,5]**
	N: 0.02 [1,2]	[1,3]	[1,4]	[1,5]
		V: 0.05 [2,3]	[2,4]	VP: 0.2*0.05* 0.0012=0.0 00012 [2,5]
			Det: 0.4 [3,4]	NP: 0.3*0.4*0.01 =0.0012 [3,5]
				N: 0.01 [4,5]

Learning Probabilities

- Simplest way:
  - Treebank of parsed sentences
  - To compute probability of a rule, count:
    - Number of times non-terminal is expanded
    - Number of times non-terminal is expanded by given rule

$$P(\alpha \rightarrow \beta \mid \alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

  - Alternative: Learn probabilities by re-estimating
    - (Later)